Web Genre Classification for Context-Specific Security Warnings

Web-Genre-Klassifizierung für kontextspezifische Sicherheitswarnungen

Master-Thesis von Anne-Christine Karpf aus Aschaffenburg

15. Mai 2014

TECHNISCHE UNIVERSITÄT DARMSTADT

SECUSO
SECURITY · USABILITY · SOCIETY

Web Genre Classification for Context-Specific Security Warnings
Web-Genre-Klassifizierung für kontextspezifische Sicherheitswarnungen

Vorgelegte Master-Thesis von Anne-Christine Karpf aus Aschaffenburg

Prüfer: Prof. Dr. Melanie Volkamer, Prof. Dr. Chris Biemann
Betreuer: Dr. Steffen Bartsch

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den May 15, 2014

(Anne-Christine Karpf)

Contents

Abstract

While internet security is an important issue, browser security warnings are hard to understand without sufficient technical background knowledge. Instead of confronting the user with a technical problem like an *invalid certificate*, one could list the actual consequences for the user, e.g. the potential loss of money when using an online banking website without an encrypted connection. This way, security warnings become more understandable and thus more helpful for the average user. However, in order to predict the consequences, the browser needs to know what genre a website belongs to, e.g. if it is an online banking site, an information site, or something else. This thesis is concerned with automatically recognizing said genre in order to make browser security warnings more understandable for the user.

In order to automatically assign a genre to a web page, a web genre classification approach is applied to the task. We use a web page collection where each web page is labelled with a genre that describes the web page context, like online banking or information site. The web pages were labelled by users during their normal browsing. This means that the collection can contain any web page a user visited and is not a set of web pages carefully selected for the task. This is important because the plan is to later apply the classifier to web pages a user visits while surfing on the internet, which are also arbitrary. A set of features is defined according to which we analyse the web pages. Different data sets are created and different learning algorithms are applied to them. The resulting classifiers are then compared in terms of their accuracy and it is analysed where they work well and what must still be improved. The best classifier on the most interesting data set achieves an accuracy of about 87.39 %. The only genre that the classifier had trouble distinguishing from the others was *Information Site*. In those cases, the user can still be warned if he starts entering data on the web page, so misclassifying a web page as an *Information Site* can be corrected to some extent. Furthermore, our results suggest that this problem can be solved if we identify mislabelled web pages in the training data, label them with the correct genre and retrain the classifier on this data set, which is left for future work. In addition, the resulting classifier needs to be tested in a user study, because this is the only way to tell whether it is good enough for a pratical application.

1 Introduction

While internet security is an important issue, browser security warnings are hard to understand without sufficient technical background knowledge. In security warnings, users are confronted with technical problems like an *invalid certificate* when trying to visit a web page. If they do not know what a certificate is and what can happen if such a certificate is invalid, the warning is useless to them. It is a better solution to list the possible consequences for the user in that particular situation instead of the technical details: If the user wants to visit an unsafe web page in order to do online banking, the security warning can tell them that they may loose money if they do that. To be able to list such consequences in a browser security warning, the browser needs to know what the web page context is, i.e. what type of web page the user is visiting and what they intent to do on that web page.

If we know the web page context, we can improve browser security warnings in two ways: Firstly, the security warnings can be specific to that particular web page context and the consequences that may arise if the web page is not safe. This way, the warning is more easily understood by the user and thus helps them to decide when it is safe to use a web page and when it is not. Secondly, a system knowing the context can use this information to show warnings only if there really is a threat and not just because there may be one. For example, an invalid certificate is a problem if the user wants to do online banking but is irrelevant if the user just wants to look for information. This means warnings using the context information will be less frequent than warnings that do not. Therefore, the user grows less accustomed to seeing the warnings and hence is more likely to take them seriously when they do occur.

The easiest way to get the context information is to ask the user what they intent to do on the web page whenever a security problem occurs. An unencrypted connection, for example, is such a security problem: For an online banking web page, an unencrypted connection is unacceptable. However, the connection is unencrypted for the majority of web pages we visit on the internet to look for information. This implies that the user is asked for their intentions almost every time they visit a web page in order to make sure their intention is not online banking. This will likely cause them to stop using the system because it annoys them, rather than improve their security. Therefore, one must be able to determine the web page context automatically in order to improve security warnings with context information in practice.

There exist approaches that assign genres to web pages automatically, and there exist approaches that ask the user what they intent to do on a web page in order to adapt security warnings accordingly. However, these two things have not yet been combined into one system that can automatically recognize the web page security context encoded by the genre and then adapt the warning to that genre without the need for user interaction. Our solution combines these two approaches in order to improve security warnings with context information in a way that is feasible in practice.

The goal of this thesis is to use web genre classification methods in order to assign genres encoding the context to web pages. Genres that encode the context information that is useful for security warnings have been defined before, so we will use these for our approach. The result of this thesis will be a classifier that can automatically detect the genre of a German web page. As training data, we use web pages labelled with a genre by users while they were surfing on the internet normally. This means that the web pages are an arbitrary collection of what users visit and thus are more respresentative for the task than a set of web pages carefully selected by experts. A set of web page features is developed, including both typical features used in web genre classification and features specific to the security warning application. We train several classifiers on different variants of the data set and evaluate their performance in terms of classifier accuracy and speed in order to determine the best classifier for the security warning application. We do not distinguish fraudulent web pages from trustworthy ones with our classifier. Also, we do not provide an implementation of the best classifier or design the context-specific warnings here. We focus on assigning genres to a web page automatically, where the genres describe the web page function and the intention with which the user visits that web page. Being able to assign such a genre automatically to a web page will help to adapt security warnings to the context and thus allows to provide useful, understable information for the user.

1.1 Background: InUse Project

This thesis is going to be a part of the InUse project. *InUse* is short for ***In****ternet* ***Us****ag****e*** *Support*, which describes the goal of the project: To support internet users in evaluating the trustworthiness of a website or a web shop. This section describes the *InUse* project and where this thesis fits into this project.

As part of the InUse project, three components are developed: The InUse scoring tool installed on the user's computer that analyses the website's trustworthiness when the website is being loaded, the InUse service provider updating this tool and doing part of the security checks, and a security seal for websites that can be verified automatically. The tool on the user's computer is integrated as a browser plugin and automatically checks indicators for a website's trustworthiness whenever the website is being loaded. Without the tool, these indicators would have to be checked manually, which

is tedious and can only be done by experts who know what to look for. As part of the tool that checks the website's trustworthiness, the website's context is analysed to detect if the user is visiting, for example, an online banking website, where security measures like an encrypted connection are very important, or an information site, where less security measures are necessary. (cf. [VBBK11], p. 1-7)

In order to analyse the context, a concept needs to be developed on how this can be done. The context analysis will be based on text recognition methods that need to adapted if necessary. The project focusses on German websites. Apart from the text analysis, other criteria specific to this application need to be determined, like the presence or absence of a password field on the website. Based on the criteria, the website will be assigned to a sensitivity class. The sensitivity classes also need to include a default class in case the website's context cannot be clearly determined. (cf. [VBBK11], p. 20)

How the web page context can be detected automatically is the part of the InUse project that this thesis will focus on. It uses genres to encode the context which where defined in a user study that is also part of the InUse project, [SVK12]. This thesis will not provide a complete implementation of the automatic context detection but instead will set the basis for this task by determining a good version of the genre set, features to describe relevant aspects of web pages and by comparing different learning algorithms in order to find out which one produces the best classifier for our purposes.

1.2 Related Work

As pointed out earlier, no previous work has been found that applies web genre classification to security levels of web pages to improve security warnings. However, there exists previous work both on improving security warnings with context information and on web genre classification techniques.

Warnings that ask the user what they intent to do on the web page are described in [SEA+09] and [SVK12]. In [SEA+09], a user study is conducted to see how people react to current SSL certificate warnings and two improved variants. They reported that their improved warnings were understood significantly better by users but could not completely prevent users from continuing to unsafe websites. One of their improved warnings asked the users what kind of website they were trying to reach whenever there was a problem with the website's certificate. Users could select from a list that they are going to visit the website of a bank, a web shop, something else, or that they do not know. If they selected any of the first two options, they were shown a warning alerting them to the high risk and discouraging them from continuing to the website. (cf. [SEA+09], p.1 and p.10)

In [SVK12], a warning framework is proposed that consists of three stages: Users are asked what type of web page they want to visit. Then they are shown a warning for this particular type of web page. And if they do continue to the web page and start to enter sensitive data, another warning is shown. For this framework, possible types of web pages are identified in a user study. The result are seven categories for web pages. They provide contextual information that can be used in security warnings and they can be mapped to risk levels. Since the user has to choose the type of web page, the number of categories was limited to seven in order to not provide the user with too many options to choose from. (cf. [SVK12])

These categories are what we are going to use here as genres for web pages. In order to sort web pages automatically into these categories, we apply Web Genre Classification methods to the task. We choose [LLK05] and [zES04] as a basis for our approach, especially because they use a large selection of features and compare how well different feature sets and different classifiers perform.

Web Genre Classification describes the classification of web pages by their genre. A genre is not the topic of a web page but rather describes the web page's type or function. Or, as [zES04] puts it: "Genre classification means to discriminate between documents by means of their form, their style, or their targeted audience".

In [LLK05], features for web documents are proposed that are useful to classify web documents according to genre. The goal behind the genre classification is to allow information on the internet to be found more easily, i.e. to enhance search based on a web document's topic by adding the possibility to search for a certain style, i.e. a genre, as well. The proposed features for this task include classic text features as used before in text classification problems, but they also include features extracted from the HTML code and the URL, which are specific to web documents. Several sets of features are proposed and evaluated. The genres used for this approach consist of two major categories, textual ones and non-textual ones. Non-textual ones are, for example, *Personal homepages*, *Image collections* and *Input pages*. Textual ones are, for example, *Research reports*, *FAQs*, and *Product specifications*. (cf. [LLK05], p. 1267) A data set of 1224 web documents in Korean was used which were annotated with a genre label by hand. (cf. [LLK05], p. 1266-1267) From the feature sets used in the classification, HTML tag features and most frequently used punctuaction marks perfom best when used on their own. Also, using the text from the document's body proved to be useful in most cases. (cf. [LLK05], p. 1272) When combining all features, an accuracy of about 74 % was achieved, which was improved to 75.7 % by applying feature selection algorithms on the feature set and only using the best combination of feature sets. (cf. [LLK05], p. 1276) Overall, they found that a combination of feature sets with "URL, HTML tags, token information, most frequently used function words, most frequently used punctuation marks and chunks" ([LLK05], p. 1276) performed best. They report

that their features can also be used in other languages than Korean. We use some of the features they described in their approach, especially those that were found to perform well.

In [zES04], web genre classification is also applied to improve search results for web documents. They evaluate the performance of different features and classifiers, and report that about 70 % of the web pages are assigned to the correct genre by the resulting classifier. (cf. [zES04], p. 256) When selecting the features, they focus on features that can be calculated quickly so that results of the genre classification can be used to improve search engine results. The genres they use are defined by a user study with the goal of identifying useful genres for the search engine application. (cf. [zES04], p. 258) The result were eight genres, *Help*, *Article*, *Discussion*, *Shop*, *Portrayal (non-priv.)*, *Portrayal (priv.)*, *Link Collection*, and *Download*. (cf. [zES04], p. 261) They use features consisting of terms from the document, linguistic features and text statistics. (cf. [zES04], p. 261-262) Additionally, features based on the HTML tags of the web page are calculated. (cf. [zES04], p. 263) Features are grouped according to how many effort it takes to compute them. (cf. [zES04], p. 264) As classifiers, MLP neural networks and Support Vector Machines were used, of which the latter performed better. (cf. [zES04], p. 266) They also suggest to use classifiers specific to a target group of users, which could be used when it is detected that the user falls into one of the target groups. (cf. [zES04], p. 267)

1.3 Sample Scenarios

To illustrate what the automatic genre detection is supposed to do and what it cannot do, three sample scenarios will be described. These sample scenarios include the kind of warning that is shown to the user, which is not directly part of this thesis but is something to keep in mind for our approach.

A user visits the website of his bank in order to do a financial transaction. The system recognizes the site as belonging to the context *Online Banking*, which strongly indicates that the user is going to log in to that site and do financial transactions. Since the website has a valid extended validation certificate providing strong encryption, the user is shown no warning as his login credentials are transmitted to his bank safely.

Now assume the user follows a link in an email he got, claiming that he needs to log in to his bank account for some reason. The link leads the user to a website that looks exactly like the one of his bank, but actually belongs to someone trying to steal the user's identification credentials and possibly do bank transactions from the user's account. Since the website looks like the bank's website, the system recognizes the site as belonging to the context *Online Banking*. However, data is sent unencrypted to that site. A warning is shown to the user, blocking his use of the fake bank website and providing him with information about the consequences in this situation (disclosure of his financial situation to a stranger, financial loss), and ideally with information on how to avoid such attacks in the future.

In the third example, the user visits the blog of a friend to read the newest articles. There is a login form on the site asking for user name and password, probably for the blog's administrator or for some content that is not meant to be seen by everyone. The connection is unencrypted. While the system detects the login form and the unencrypted connection, it also detects that this is an *Information Site* and assumes the user is just looking for information, so it shows no warning. If the user does decide to log in at some point, the system recognizes that the user just clicked into the password field and warns him that the connection is unencrypted. Since the context is *Information Site*, the warning is less alarming than it would be in the *Online Banking* scenario.

These examples show what our approach is supposed to achieve: When we do warn the user, we can provide them with context-specific and understandable information. And when we do not warn the user, we do this because we know from the context information that there is no threat in that particular context. We do not try to detect fraudulent web pages like phishing attempts, but instead focus on the user's intention: If they visit a web page to do online banking, we want to assign the corresponding genre to that web page, no matter if it is a fraudulent or a trustworthy web page. Whether all necessary security measures are in place for a web page of that genre can then be checked by the InUse scoring tool.

1.4 Reasons for the Approach

This section gives reasons for why we choose to apply a machine learning approach to the task of automatically detecting a web page's genre.

In order to develop a system that can automatically assign a genre to a website, we decided to use a machine learning approach and train a classifier on a set of web pages already labelled with a genre. From these web pages, features are extracted that are used in other web genre classification problems and performed well.

We choose to train a classifier on training data in order to get rules that can be used to sort new web pages into the genres. The machine learning approach was chosen over hand-writing rules to distinguish the genres because, although some genre characteristics are obvious, it is hard to find a general rule of what an algorithm has to look for in order to identify a genre.

Some characteristics for certain genre are obvious, for example looking at input fields on the website. Only when there are input fields on the website the user can enter private data, and usually only when there is a password field the user can enter their login credentials. Hence the absence of any input fields suggests that the genre of that website is *Information Site*. When input fields are present, however, things get more difficult. The presence of a password field can hint at any genre where the user needs to login, which includes things like *Shopping*, *Social Networks*, *Online Banking*, *E-Mail* or *Data Exchange*, so all genres except *Information Site* are possible. Furthermore, it is possible that a website that is primarily used to search for information also has a login form, e.g. a public transport website which, as its primary function, allows users to search for train connections but also offers the possibility to login and buy tickets online. Therefore, the presence of input fields can imply any genre.

To be on the safe side, one could classify the public transport website as *Shopping*. This implies, however, that the user will see a warning every time he searches for a train connection since he visits the website via an unencrypted connection by default. This is not desirable, since the user will be annoyed by the warning and learn to ignore it even in cases where it is actually appropriate.

Another obvious thing to look for are certain keywords: *Warenkorb* (shopping cart), *Versandkosten* (shipping fees) for *Shopping*, *PIN* (personal identification number, i.e. the online banking password), *Online Banking* for *Online Banking*, and so on. However, no such keywords can be found for Information Site, since Information Site is the umbrella term for anything that does not fit into the other genres. Furthermore, the keywords for data exchange are less obvious, mainly because in many cases, it is hard to say what exactly qualifies as data exchange and what does not.

Therefore, we choose the machine learning approach as hand-written rules to distinguish the genres cannot be efficiently determined.

1.5 Definitions

This section describes why we classify web pages and not websites and how those terms and some related ones are used within this thesis.

The goal of the InUse scoring tool is to evaluate a website's trustworthiness when the user visits the website in a browser, which includes identifying the context of the website in order to adapt warnings to that context (cf. [VBBK11], p. 5 and p. 11).

We use genres to encode that context, but we do not assign a genre to a website but rather, to a web page. Since both terms are sometimes used interchangeably, we give a short definition of website and web page here. We also define some other terms needed in that context, namely the terms Uniform Resource Locator (URL), home page and host. After the definitions, we explain why we have chosen to assign the genre to a web page rather than a website.

The term *web page* refers to a single page on the internet. A *website* on the other hand may consist of several web pages. For example, the blog of someone may have several pages: A list of the actual blog posts, an About page with information about the blog and its author, and a Contact page with the author's address. The website consists of all three of these pages and potentially more, while each of the three pages is considered a web page of its own. On can also say that a web page is anything that can be identified by a Uniform Resource Locator (URL).

The Uniform Resource Locator (URL) of a web page is its unique ID that is used to find the web page on the internet, i.e. the address of a web page wich we have to enter in the web browser to get to that web page. It consists of several parts: The first is the protocol that is to be used in order to retrieve the web page, for example http or https. The second part is the web page's host, which can be a domain or an IP address. Optionally, a port number can follow the host. After that, the path name follows, which describes the path to the web page on the host. There may also be query string included at the end, which specifies e.g. search parameters. Also, there can optionally be a text anchor referencing a certain point on one web page. A URL uniquely identifies a web page, i.e. no URL references more than one web page.

All web pages from one website will usually share the same *host*, which is the part of a URL between *http://* or *https://* and the next */* after that. For the URL *https://www.informatik.tu-darmstadt.de/en/students/*, for example, the host is *www.informatik.tu-darmstadt.de*. The host, prefixed with *http://* or *https://* and with a trailing */*, usually leads to the home page of a website. The home page is the first web page a user sees when visiting a website, also called its start page.

Now that the necessary terms are defined, we will explain why we assign the genre to a web page rather than a website. Firstly, a web page is uniquely identified by its URL, which allows us to specify exactly what web page we are referring to and what it consists of, i.e. the HTML code available at that URL. This means we can easily specifiy for a web page what exactly we assign our genre label to. For a website, it is more difficult to specify which web pages belong to that web page. All web pages with the same host could be considered to belong to the website, but they could also belong to different websites, or the website could include more than just the web pages from the host, e.g. when including an external service, like a discussion board or an RSS feed. Also, even if the definition that a website consists of exactly the web pages with that host holds true, it is difficult to find all these web pages in order to build a feature vector from all of them. There usually is no list we can use to find all of the web pages that belong to the website. One could decide to

just use the website's home page or the home page and all the web pages with the same host that it references, but that is a lot more difficult to handle than the single web page already, especially later when we want to classifiy the web page while the user loads it in the browser. Secondly, what the user sees in the browser at any given moment is just a web page and not the full website. Therefore, being able to assign a genre label to a web page is sufficient to protect the user from dangers arising from him visiting that web page. Also, it is still possible to take other web pages from the website into account, like the host or the web pages the currently viewed web page links to while the genre label is still assigned to just the web page. And thirdly, different web pages may actually belong to different genres although they are part of the same website. For example, the contact page and the discussion board within a web page - the former is used to look for information and so would best be described as Information Site, the latter to log in and discuss something with other people, which could be called a Social Network.

Therefore, assigning genres to web pages is easier to work with and also sufficient for our purposes. Also, it is possible that different web pages of a website belong to different genres, so assigning the same label to all of them may not lead to a correct result in the first place. Therefore, the approach described in this thesis focusses on assigning genres to web pages rather than to websites as described in the description of the InUse project (cf. [VBBK11]) but still achieves the proposed goal.

2 Approach

This chapter describes how we are going to proceed in order to develop a sytem that can automatically determine a web page's genre. First, anything required for the machine learning approach is described: The genre set, the training data, and the learning algorithms we are going to use. Then we explain how we are going to train the classifiers and how we evaluate their results.

In order to automatically determine the web page context, we need several things: First of all, we need genres that capture our notion of context, i.e. genres that tell us what kind of web page the user is visiting and what he visits the web page for. Next, we need training data for the learning algorithm, which means we need web pages already labelled with the genres. In order to find out what web pages from a genre have in common, we need criteria according to which we can compare them, the so-called features. By calculating each web page's value for each of the features, we can reduce the web page to a feature vector, i.e. a list of all the feature values. Furthermore, we need learning algorithms we can apply to the training data, i.e. to the set of feature vectors. Each learning algorithm produces a classifier, i.e. a model that allows us to assign a genre to new web pages according to their values in the feature vector. And finally, we need a way to evaluate the performance of our classifier in order to find out if it works well enough for our application.

Apart from these things we need, it is important that both the feature values and the classification result can be computed fast. When the user visits a web page in the browser, the InUse scoring tool must be able to compute the web page's genre and check if all security measures are in place before the user can enter any data, such as their password, on the web page. Therefore, the features must not be complicated ones that require a lot of time to compute, and the classifier must be able to quickly decide on the genre based on the feature values.

It is of course important that the classifier's accuracy is high, i.e. the percentage of web pages the classifier classifies correctly. While this is obvious, we have a reason why this is especially important for our application: Both not warning the user when there is a threat and warning them when there is no threat comes with a cost. In the first case, the cost is obvious: If we choose to not warn the user when there is a threat, we do not protect them from the threat and hence consequences like e.g. the user's online banking account being hacked may occur. For this reason, it may seem like a good idea to always warn the user when in doubt just to make sure, but this also comes with a cost: If we warn the user too often when there is no threat, they will get used to web pages being safe although a warning is shown. As a result, the user will take the warnings less seriously after a while, so they may continue to an unsafe web page and e.g. enter their password, assuming that although a warning was shown, there will be no negative consequences as there have not been any in similar situations before.

Having explained the requirements for our approach, we will now describe the web genres we use, where the labelled web pages come from and what features we use to describe them. We will then explain which learning algorithms we use to train a classifier, and how we evaluate the classifier's performance.

2.1 Web Genres

This section explains where the genres used in this paper come from and why they were chosen for this task.

The web genres used here come from a user study. The study was conducted in order to identify seven categories for web pages. These seven categories are supposed to be used in a warning framework where the user is asked with what intention he visits a website so he can then be shown a warning specific to the category he selected. The limit of seven was chosen in order to not provide too many options to choose from, which may be confusing for the user.

In the study, the participants were asked to sort 67 web pages into different categories. They were not provided a set of categories to choose from but rather had to come up with their own categories. In the first round, the participants could sort into as many categories as they wanted while in the second round, they were restricted to at most seven categories. The categories chosen by the users were then analysed according to how often they were used and how many web pages were sorted into the same category by different users. From these and other criteria, the seven most frequently used and most useful categories for the given task were determined. The categories are *Information Site*, *Shopping*, *Social Network*, *Online Banking*, *E-Mail*, *Data Exchange*, and *Other*. (cf. [SVK12])

2.1.1 Basic Genre Set

This section describes the basic genre set we derived from the categories presented in [SVK12].

Since these categories capture our notion of website context very well, they were chosen as the basis for our task of automatically determining website context. The categories will be the web genres we use in this paper, with one modification: The category *Shopping* is split into two genres, *Shopping 1* and *Shopping 2*. *Shopping 1* is for web shops

Table 2.1.: The names of the eight web genres, an description of the web pages that fit into that genre, and the genre's identifier used by our software.

Genre	Description	Identifier
Information Site	web pages you visit only to search for information and not to log in or to provide other personal data	info
Shopping 1	Web shops where you buy things that will be delivered to you, for example books, clothes or electronic devices	shopping1
Shopping 2	Web shops where you buy things that do not need to be delivered to you, for example a vacation you booked or software you can download	shopping2
Social Network	Websites were you can connect with friends or business contacts. You need to log in for these sites, and you usually create a profile with information about yourself, can look at the profiles of others and send messages to other people.	social
Online Banking	Websites where you log in to do banking transactions, for example the website of your bank or credit card provider. When logged in, you can, for example, see your bank account's balance, transfer money to someone else's account or do other such things.	banking
E-Mail	Websites where you log in to send or receive messages like for example emails	mail
Data Exchange	web pages where you can log in and upload photos, videos, text or other data in order to share it with others or just for yourself to see	data
Other	Websites that do not fit into any of the other genres	other

where you can order things that will be delivered to you. *Shopping 2* is for web shops where you can buy things that do not need to be shipped to you, for example a software you can download or a hotel room you reserve online. This distinction is made because these two types of web shops are different according to German law, so being able to differentiate between the two in the InUse scoring tool is preferable to treating them as one.

Hence we use the following eight genres to describe the website context: *Information Site*, *Shopping 1*, *Shopping 2*, *Social Network*, *Online Banking*, *E-Mail*, *Data Exchange*, and *Other*. *Information Site* covers all web pages you visit only to search for information and not to log in or to provide other personal information. This can be, for example, a search engine, a newspaper website, the blog of a friend, or even a video streaming portal if the user does not need to log in to see the videos. *Shopping 1*, as described above, is for web shops where you buy things that will be delivered to you, for example books, clothes or electronic devices. *Shopping 2* is for web shops where you buy things that do not need to be delivered to you, for example a vacation you book or software you can download. *Social Network* covers web pages were you can connect with friends or business contacts. You need to log in for these sites, and you usually create a profile with information about yourself, can look at the profiles of others and send messages to other people. *Online Banking* is for web pages where you log in to do banking transactions, for example the website of your bank or credit card provider. When logged in, you can, for example, see your bank account's balance, transfer money to someone else's account or do other such things. *E-Mail* covers all web pages where you log in to send or receive messages like for example emails. *Data Exchange* means web pages where you can log in and upload photos, videos, text or other data in order to share it with others or just for yourself to see. *Other* is the genre for all web pages that do not fit into any of the other genres. It is not so much a genre by itself but rather a default option to choose when it is unclear what genre the website belongs to. Table 2.1 gives an overview of the genres, their description and the label we will use in our software to identify a genre.

2.1.2 Genre Mappings

In the previous section, we defined the web genres we are going to use for our classification problem, which are explained in table 2.1. Here, we are going to define two more sets of genres. These two sets result in simplified versions of the original classification problem. If we cannot train a classifier that satisfies our requirements for the original classification, we can use these simplified versions and see if we get a better classifier that way. However, by reducing our classification problem to one with less genres, we loose some details as well. This means the context-specific warnings for those new genres will be less specific as well, because we have less information about the web page than before. However, if we can train a classifier that works well for such a reduced genre set, this is still better than having one using all the information but with a low accuracy. Therefore, we will now first define how such new sets are created from the original genre set, and then specify what the two new genre sets are and what their genre labels mean.

Contrary to the genres defined in 2.1.1, we do not use these two new sets of genres explicitly to tag web pages with them. Rather, we define which of the basic genres maps to which new genre and re-label the web pages in our collection accordingly. Since we do not tag web pages but instead map the original genres to new genres, we call this a genre

mapping. It consists of the set of new genres and the rules according to which the original genre label of a web page can be mapped to the corresponding new genre label. Mathematically speaking, a genre mapping defines an equivalence relation on the orignal genre set and a name for each of the equivalence classes induced by this relation. This implies that there can be more than one original genre in each new genre, but each original genre must be mapped to at most one new genre. Note that we do not use the instances from the genre Other here, i.e. we do not define a rule for how instances from Other map to the new genre labels in each of the two genre mappings; we define the genre mappings based on the genre set including Online Banking, Shopping 1, Shopping 2, Data Exchange, E-Mail, Social Networks, and Information Site.

We define two genre mappings. One maps the genres Shopping 1 and Shopping 2 into one combined genre, Shopping, and leaves the rest of the genres as they are. The other maps genres to risk levels. These risk levels describe how dangerous it is if a user visits an insecure web page of that genre. These two genre mappings will be explained in the following, first the *Shopping combined* mapping, and then the *Risk Level* mapping.

The first genre mapping we define is the *Shopping combined* mapping. It maps the genres Shopping 1 and Shopping 2 into one combined genre, Shopping, and leaves the rest of the genres as they are. We define this genre mapping because Web pages in Shopping 1 will probably be very similar to those from Shopping 2 and vice versa. Their distinction is a legal one rather than one based on the expected site structure. There may even be web pages that belong to both genres, since a web shop can sell things that need to be send to the customers and things the customers can download at the same time. An example for this is a web shop where you can buy music: You may have the option to either buy the music on CD and let it be shipped to you, or you can buy it as an MP3 download, which requires no shipping. For this reason, it may not be possible to differentiate between the two versions of shopping web pages. Therefore, we decided that it is best to use both Shopping 1 and Shopping 2 as genres when tagging web pages, but we define a genre Shopping as well, which contains all web pages labelled as either Shopping 1 or Shopping 2. This genre Shopping can be used in training the classifier instead of using Shopping 1 and Shopping 2 in order to see if this yields better results.

The second genre mapping we define is the *Risk Level* mapping. For the security application, the genres we use can be grouped into different security levels. To which security level a web page belongs depends on how dangerous it is to use a web page of a particular type although it is not secure. Accordingly, the highest security measures need to be in place for websites of the highest security level, e.g. an encrypted connection with an extended validation certificate. We focus here on the direct consequences that can occur when using an insecure web page of a particular genre.

As a first step, we describe the direct consequences arising from using an insecure web page of a particular genre. Based on that, we suggest a risk level each of those genres belongs to, and define our Risk Level genre mapping accordingly.

Logging in to an insecure web page may enable someone to gain access to your account and to use it for their purposes. On a Shopping 1, Shopping 2 or Online Banking web page, this may cause you loss of money because someone can transfer the money from your bank account to their own, or because they can shop at your cost. On E-Mail, Data Exchange and Social Network web pages, someone with access to your account can see your private messages or files, and they can use the account to impersonate you. On an Information Site, you do not log in, so an attacker cannot gain access to your account. They can only get some information, e.g. if you enter something in a search field, but it is unlikely that this is sensitive information that allows them to cause you a loss of money or impersonate you.

We consider the web pages in Shopping 1, Shopping 2 and Online Banking as web pages with a high risk. If something is wrong with any of those web pages, you may loose money. It is possible that you loose all the money in your bank account or even more, e.g. when your credit card is used. This can result in you being unable to pay your rent or buy food, so it has the most drastic consequences.

Web pages in E-Mail, Data Exchange and Social Networks are considered to be of medium risk. If something is wrong there, personal information about you may be disclosed or someone may impersonate you. This is undoubtedly undesirable for you, but it will likely be less harmful than the loss of money.

Information Sites are considered to belong to a low risk level. If something is wrong on web pages where you do not log in, no one can gain access to your account. They can learn something about you, but it is unlikely to be very sensitive information they can use to cause any harm to you.

Accordingly, we define our risk level mapping as following: Shoppping 1, Shopping 2 and Online Banking map to a new genre High. E-Mail, Data Excahnge and Social Networks map to the new genre Medium. Information Site maps to the new genre Low.

Note that one could define a different assignment of the different genres to risk levels when considering not only the direct consequences but also the indirect consequences that may arise from using an insecure web page of a particular genre. An example are E-Mail web pages: You do not directly loose money if someone gains access to your email account. However, you may have used that email account to register for a web shop. Most web shops provide the possibility to send you an email with a link where you can reset your password for that web shop in case you forgot it. Thus, someone who can read your emails can see for which web shops you registered and what your username or customer reference number is for that web shop. They can then visit that web shop's internet site, request that a password reset link be sent to your email account, and then use this link to change the password for your web shop account to something they

Table 2.2.: Genre Mappings. The leftmost column shows the original genre set, while the Shopping combined and Risk Levels columns list the genres used in the respective mapping. The genres in the mappings consist of those genres from the original set that are contained in the rows which the cell with new genre's name spans.

Basic Genres	**Shopping combined**	**Risk Levels**
Online Banking	Online Banking	High
Shopping 1	Shopping	
Shopping 2		
Data Exchange	Data Exchange	Medium
E-Mail	E-Mail	
Social Networks	Social Networks	
Information Site	Information Site	Low

choose. This way, they gain access to your web shop account as well, allowing them to shop at your cost and thus causing you a loss of money. Some web shops offer additional security measures that can prevent this, for example by asking the person to answer a security question correctly before letting them reset the password, provided you selected such a security question and entered the correct answer when creating the account. However, currently many web shops do not use any such method to secure your account. This means if someone has access to your email account, it can have the same consequences as if they gained access to your web shop account directly. Therefore, one could associate E-Mail web pages with a high risk as well.

However, accounting for all of the indirect consequences of visiting an insecure web page of a particular genre would go too far here. There are too many possibilities that we would need to be consider. Therefore, we defined the risk levels as described above, because it is a reasonable assignment when considering only the direct consequences of visiting an insecure web page of that genre. Additionally, we cannot list all possible consequences in a security warning, so it makes sense to focus on the direct consequences: After all, we want to use the genre in a warning to give users an idea of what can go wrong, but not to confuse them with too much information. Additionally, we need to keep in mind that our classifier can only work well if the web pages within one genre have enough similarities, so putting too many web pages with different functions into one genre will not help in training a good classifier.

An overview of the basic genre set and the two genre mappings derived from that can be seen in table 2.2.

2.2 Data

We need web pages labelled with a genre for our approach, and there are different sources for such a web page collection. In this section, we are going to describe the data we use for our approach, where it comes from and how we preprocess it.

2.2.1 Sources for Web Pages Labelled with Genre

This section describes the possible sources for web pages labelled with a genre, and which source we decided to use.

Our first option was to take the websites that were used in [SVK12] to determine the categories we use as genres. The websites were essentially chosen from a list of the most popular websites in Germany, and some websites from that list were replaced by one that is more common in the Rhein-Main region than the original one. As a result of the user study, each of the websites was annotated with one of the categories. However, the websites' URL and HTML code is not available exactly as it was used in the study anymore, and they most likely have changed since the study. Maybe some have changed in function and hence changed their category as well. Furthermore, this data set only consists of 67 web pages. We can increase this by determining a set of similar URLs to those from the study and assuming that they belong to the same category, resulting in a data set of about 1150 web pages. But still, both of these sets only use a website's home page rather than any URL of a web page. And they do only have one category for Shopping web pages and not the distinction between two different ones we want to use. So for these reasons, we decided to use a web page collection that was obtained from a browser add-on, namely the WeKaSa web page collection. It includes about 5000 web pages, both home pages and other pages of a website. Since the data comes from a browser add-on, it is a set of web pages that users actually visited when surfing on the internet, so it is also more representative for what the InUse scoring tool will have to classify than the websites selected for the user study. We will now describe the web page collection we use in more detail.

WeKaSa Web Page Collection

The data we decided to use in our approach comes from a Firefox extension which is called *Website-Kategorie-Sammler*, or *WeKaSa* for short. It asks users every once in a while what category the web page they are currently viewing belongs to. How this dialog looks can be seen in figure 2.1. The users can choose between the eight genres we described earlier. If they select the genre *Other*, they can also enter an alternative genre if they want to. The URL of the web page and the genre the user selected are then sent to the *WeKaSa* server, and also the alternative genre if one was entered.

When the *WeKaSa* server receives a new URL labelled with a genre from a user, it visits this URL immediately and downloads the web page's HTML, along with the HTML of the first 200 web pages it links to. Since web pages on the internet may change radically or even disappear over time, it is important that the data is stored right away so that the HTML of the web page in our database corresponds to what the user saw when he assigned the genre label to that web page. Due to transmission latency, the server may not see exactly what the user saw, but it is safe to assume that in most cases, what the server sees is sufficiently similar to what the user saw seconds before for the genre assignment to remain valid. This way, we collected about 5000 URLs labelled with a genre. They are from web pages the users of *WeKaSa* visited during their normal browsing. They can be any kind of URL, not just the address of the web page's homepage - they can also be URLs of a subpage or contain a query string. More information about WeKaSa, how it works and what kind of data it stores, is avaible in [Kar14].

The distribution of web genres within this data set may give a hint on how the actual distribution of genres on the internet is. However, it has some bias. For one, users were able to select how often they want to be asked for the category of a web page. Therefore, the resulting data will contain more web pages from people who selected to be asked very often than from those people who wanted to be asked less often.

Furthermore, users can decide each time they are asked for a genre if they want to send this data for this web page or not. User feedback revealed that people may choose to send the genre less often on web pages such as online banking and shopping web pages due to privacy concerns. Additionally, if people take extra precautions when visiting online banking sites, e.g. by using a specialised software or a different browser, then they may also never tag an online banking site with WeKaSa. Hence, these web pages may be less frequent in our collection of web pages than they are actually used on the internet. To account for that, we manually tagged a number of online banking web pages. This, of course, introduces a bias of its own. The online banking sites were selected by searching for the term *online banking* on Google and picking the first (approximately 100) links that did indeed lead to an online banking log in form. Also, not everyone may have selected *Other* and typed a text when none of the categories seemed to fit the web page. Some people will simply have clicked Cancel, as it was the faster option. This means there could be more web pages that do not fit into our categories. Some of the web pages labelled as *Other* by users actually do fit into our genres, even if they are not labelled as such. Many entered the topic of the web page, rather than its genre, as an alternative option. Many of those sites are actually information sites.

What this means is that the data collected by WeKaSa is not necessarily a representative collection of the most frequent or most frequently used web pages on the internet.

In addition to the information that the WeKaSa server stored right away, we later downloaded the web page's host. Therefore, the data we will use for task consists of three parts: For each web page, the HTML of that site itself has been downloaded, and the data from the first 200 web pages it links to. These were downloaded at the time when the classification was processed. Additionally, the web page's host was downloaded later.

The Users Who Tagged the Web Pages

This section describes what we know about the people who tagged the web pages we are going to use in order to train our classifier.

Knowing who the people are that tagged the web pages with a genre label is important for evaluating the results of our research. What we know about the users of WeKaSa, who tagged the web pages, allows us to see whether they are a representative sample for the target group of the InUse scoring tool. If they are, the web pages they tagged are likely to as well be a representative sample of what the average user may encounter when visiting web pages. If they are not, the web pages in our dataset may not be so representative either.

How well the classifier trained on the WeKaSa web pages works for the average user may depend on that, so we need to know a bit about our users. In the ideal case, the classifier we train works well for the average user. In the less ideal case, the classifier does not work well. If we know that our data has some bias and we know where it came from, that is one of the things we can go back to, change them, and see if this improves our results. Therefore, it is important to know something about the people who tagged the web pages with a genre label.

We obtained our information about the WeKaSa users from two sources: Firstly, we know where we promoted WeKaSa and the project, so we know who was likely to learn about it. And secondly, users of WeKaSa were asked for some personal information when using WeKaSa for the first time. Supplying this information was optional, so we do not know

if it was provided by all the users. Nevertheless, the data from the 31 users that did provide information about themselves gives us an idea of who used WeKaSa and what their background is.

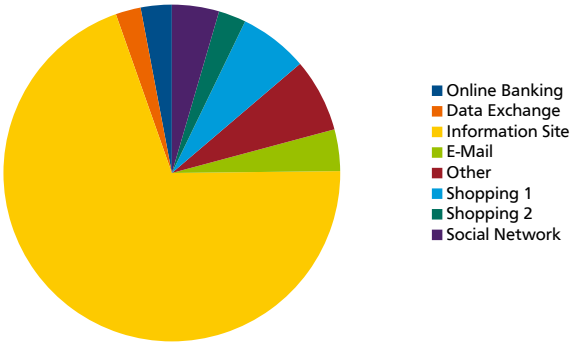
WeKaSa was promoted on the internet via mailing lists and message boards of the computer science department. Also, the link to WeKaSa's download page was given to friends and family via social networks or email. As a result, it is likely that many of the WeKaSa users are computer science students or people working in the computer science department at the TU Darmstadt. Also, most users probably live in Darmstadt or somewhere else in the Rhein-Main region. Therefore, it is possible that the web pages tagged have some local bias, i.e. that many of them are from organizations or people in the Rhein-Main region and are concerned with topics specific to that region. Also, if most users are studying or working in computer science, there will probably be many web pages related to computer science topics, more than there would be if WeKaSa users where just randomly selected people in Germany. As we are interested in the web pages' genre and not their topic, this may not be a problem in training a good classifier. It is, however, something to keep in mind when applying the classifier trained on this dataset to a broader use.

The users of WeKaSa were asked to provide some information about themselves. From this data, we know that WeKaSa has at least 31 users, namely those who answered the questionaire about themselves. They were asked questions like when they were born, how much they earn and how often they use the internet. In the following, an overview of the users' answers is given.

The WeKaSa users were born between 1946 and 1993, so they are between 21 and 67 years old. Most users are born between 1980 and 1993, so the majority of users is in their twenties or early thirties. Most users have a monthly net household income of less than 1500 euros. The majority of users has a university degree or Abitur, i.e. a qualification to study at the university, or Fachabitur, allowing them to study at a university of applied sciences. Judging from age, income and educational level, it is likely that most of the users are students. Since WeKaSa was mostly promoted within the university, this is not surprising. Of the WeKaSa users, 68 % are male and the remaining 32 % are female. Most users use the internet on a daily basis, while a few others use it at least several times a week. No users use the internet less frequently than that. Most have been using the internet for more than ten years (77 %), some more for seven to less than ten years (19 %), and one person for three to less than seven years. No one uses the internet for less than three years. We can see that most WeKaSa users have been using the internet both frequently and for quite some time, so they are likely to be experienced internet users. Most users owned several devices that can be used to access the internet: The most common device that nearly all users own is a laptop computer (94 %), followed by smart phones (77 %) and desktop computers (61 %). Tablet PCs are owned by about one third of the users (32 %), and some also own games consoles (19 %). No user answered that he owns none of the devices specified, which is not surprising given that he is taking part in a study requiring access to the internet via a computer. On average, each user owns about three of the devices they were asked about.

Table 2.3.: Distribution of the German web pages in the WeKaSa dataset over the different genres

Genre	web pages
Online Banking	79
Data Exchange	64
Information Site	1850
E-Mail	106
Other	187
Shopping 1	175
Shopping 2	70
Social Network	120
Total	2651



2.2.2 A First Look at the Data

This section describes what we can find out about the data we are going to use before processing it any further, i.e. before extracting features from it and applying a learning algorithm.

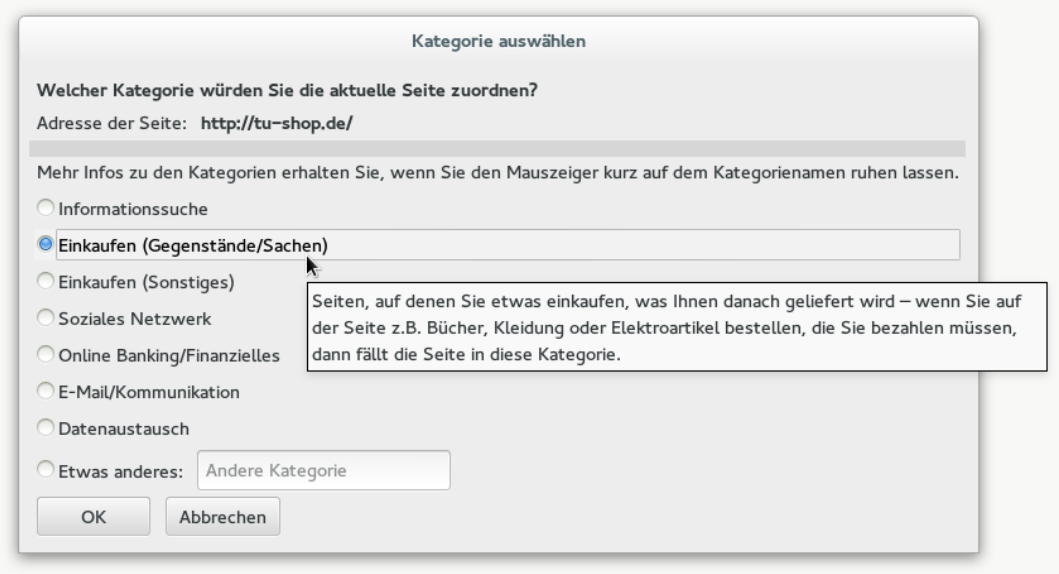


Figure 2.1.: WeKaSa is asking the user to assign a category (i.e. a genre) to a web page. The categories are the German versions of the genres we use, from top to bottom: Information Site, Shopping (Things), Shopping (Other), Social Networks, Online Banking, E-Mail, Data Exchange, and Other. When selecting Other, the user can enter an alternative category, which will also be stored. The tool tip provides more information on each category when the user lets the mouse hover over the category name.

In total, the WeKaSa data consists of 4906 web pages tagged with a genre label. Of those web pages, 2651 are in German and therefore suitable for our research. How many of the German web pages were tagged with each genre can be seen in table 2.3. Most web pages were tagged as being information sites, which is no surprise since people typically visit a lot of different information sites on the internet but only a few web pages for e.g. online banking or shopping.

Looking at the training data revealed that web pages within Banking and Shopping 1 or 2 are the most similar to other web pages from their genre, and web pages from data are rather diverse. The most different from each other are web pages in Information Site. This is not surprising, given that an Information Site is virtually anything where users do not need to log in.

When selecting arbitrary web pages from the genre *information*, many appear to be mislabelled, since they are tagged as information but are obviously web shops or belong to another genre. However, from the point of view of the user, the label Information Site may still be accurate because he might have visited this web page just to look for information and not to buy something. To our learning algorithm, this may pose a problem since web pages classified as information may esentially be anything, because any web page could be visited just to look for information at some point. However, we will for now leave these web pages as they are. Ideally, we will still be able to correctly identify the web page genre and maybe even distinguish cases where a web shop was visited to look for information from those where it was visited in order to buy something. And if that is not the case, we know that a possible cause for this are the incorrectly labelled web pages in our collection. We can then clean up the web page collection, train a new classifier on the clean data set and see if we get better results.

For this thesis, data from Other has mostly been left out. However, one can use the alternative genre suggestions entered by users in order to see if our genre set is sufficient to describe typical web pages or if more genres are needed. Furthermore, there are many pages that, according to what was entered as an alternative text, may fit well into the genres we already have and was mistakenly assigned the genre Other. These web pages can be sorted into the correct genre and be used to increase the number of labelled web pages. Both is left for future work.

In general, what the difference in similarity of the web pages in each genre suggests is that for some genres, the definition is rather obvious, and for others, it is not. Therefore, different people may assign the web pages to different genres because they have different assumptions as to what belongs into a genre.

We have 4957 URLs total in our data set, of which 4383 are distinct URLs. This means that on average, a URL gets labelled with a genre about 1.13 times. Therefore, most of the URLs in our web page collection will have only one genre label, so no two conflicting labels exist in most cases. If we group URLs by their host, we see that there are only 2717 different hosts in the collection. So for each host, about 1.82 URLs exist in our database. The maximum number of URLs per Host is achieved by www.google.de with 70 URLs per Host. The minimum number of URLs per host is one, which is the case for most of the hosts.

Table 2.4.: HTTP status codes returned for the web pages in the WeKaSa dataset. Almost all web pages were download with a status of *200 OK*.

HTTP Status	Web Pages
200 OK	4391
302 Found	254
0 Connection interrupted	129
301 Moved Permanently	38
403 Forbidden	27
404 Not Found	24
401 Unauthorized	18
500 Internal Server Error	13
303 See Other	6
503 Service Unavailable	4
307 Temporary Redirect	1
405 Method Not Allowed	1
Total	4906

HTTP Status

This section tells us something about the HTTP status codes that were returned when downloading the web pages. This is important because the HTTP status code informs us whether the web page was downloaded succesfully or not.

In order to see whether our web pages were downloaded succesfully, we also stored the HTTP status code returned by the server when downloading the web page. As you can see in table 2.4, the majority of web pages has an HTTP status of *200 OK*, which means no problems occured and the web pages was downloaded succesfully. For the web pages where this is not the case, WeKaSa stored the HTML that was returned regardless of the error code. This is the case because we assume that in cases where the server returned an error code like e.g. *401 Unauthorized* or *403 Forbidden*, it means that we access a web page that is only available after login, which does tell us something about that web page as well, so it may be useful in classifying web pages. Furthermore, WeKaSa did not follow redirects, since that would have implied storing a separate record for the originally requested URL and the one it redirects too, possibly several times if several redirects occur, which would have complicated matters too much. The code *0* is not actually an HTTP status code but was the default value it was set to if the sever could not be reached. We use all the web pages from the dataset regardless of HTTP status. Most of them were successful downloads after all, and for the others, it is possible that the error page that was shown is still useful for our classification.

Web Page Language

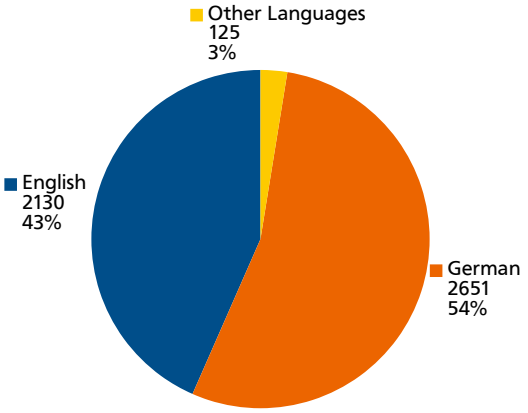
This section described what languages the web pages in our data set are in, and how many of them are German web pages, i.e. web pages we can use for training the classifier.

In this thesis and in the InUse project, we focus on German web pages, i.e. we want to be able to classify German web pages according to genre. We considered labelling only German web pages in the first place but decided against it for two reasons: Firstly, we did not find a good language detector for JavaScript which we could use in the WeKaSa extension, so in order to only ask users for a genre when the web page was German, we would have needed to send the web page's raw text to our webservice to find out the language. This was deemed to be impractical and also, not very desirable with respect to the user's privacy. And secondly, by collecting all web page classifications regardless of their language first and identifying the language later, we get an impression of what languages the web pages visited by the WeKaSa users are in. WeKaSa itself is only available in German and was promoted only locally, so we can assume that most or all WeKaSa users live in Germany. Therefore, the language distribution within the WeKaSa dataset can tell us how useful a tool can be for German users that is able to classify German web pages only. This language distribution is shown in table 2.5. Most web pages are in German (54 %), followed by English (43 %), and the remaining 3 % are in several other languages, none of which represents a large part of the data set by itself. Therefore, the two major languages present in our dataset are German and English. Since we are only interested in German web pages, this means we can use about 54 % of the full WeKaSa dataset for this thesis. The rest of the data set is for now only used to get the language distribution, which tells us approximately on what percentage of the web pages a user sees when browsing we can apply a German-only classifier. Furthermore, the non-German web pages in the data set can be used for future studies, for example if we decide to train a classifier for English web pages as well.

It is possible that there are less English and more German web pages in what the average German user sees normally. For one, it is likely that many of the WeKaSa users are computer science students or employees in computer science, since the study was conducted at the computer science department of the TU Darmstadt and naturally, WeKaSa was promoted there and is probably used especially by people there. In computer science, a lot of material, like technical documentation or support forums, is in English, so it is likely that the average German computer scientist sees more English web pages than the typical German user. Additionally, some web pages are available in several languages and decide which version to send to a user based on that user's HTTP ACCEPT LANGUAGE headers. Most German users will have German listed as their preferred language setting, so they will see the web page in German. Our webserver that stored the data did not have any such setting but just used the default language of each web page, which can be English even for web pages of German companies. Therefore, some of the web pages we have in our dataset in English may have been seen in German by the user. Similarly, whenever the query for a specific web page produced an error, our webserver probably has downloaded a default error page in English sometimes when the actual web page (e.g. one the user could see because he was logged in but our webserver could not) was, in fact, in German.

Table 2.5.: Distribution of the languages within the WeKaSa dataset.

Language	Web Pages
German	2651
English	2130
Kinyarwanda	59
Spanish	14
Portuguese	11
Italian	9
Quechua	9
Dutch	5
Luxembourgish	3
French	2
Polish	2
Aragonese	1
Basque	1
Czech	1
Hungarian	1
Latin	1
Lithuanian	1
Malagasy	1
Malay	1
Slovenian	1
Swahili	1
Swedish	1
Total	4906



2.2.3 Preprocessing

This section describes what preprocessing steps are applied to the data we collected in order to use it for our application: We need to partition it into a training and a test set in order to train a classifier on the former and evaluate its performance on the latter. And since the data only consist of the web pages and the pages they reference to but not their hosts, we need to download the host for all the web pages we are going to use. As a last step, we are going to construct different data sets from our data in order to see if we get better results for the trained classifier on one data set than on another. In the following, these steps are described in more detail.

Training and Test Set

The data collected by WeKaSa from September 1st, 2013 up until February 18th, 2014 is partitioned into a training set consisting of 85% of the data, and a test set consisting of the remaining 15% of the data. Training of the classifier and

optimization of the features used is done on the training set. When the learning process is finished, the results will be evaluated separately both on the training set and on the test set.

Web Page Host

In some cases, the web page that was downloaded by the WeKaSa server will look very different to what the user saw when he tagged that web page with a genre. This is the case for web pages where the user was logged in. For privacy reasons, we cannot use the web page as it looks to the user after log in, since that would allow us to, for example, read his private mail and store it in our database. So in order to not violate the privacy of *WeKaSa* users, the server downloads the web page the way anyone not logged in would see it. Therefore in this case, there may be large differences between what the user sees and what is stored on our server. Where the user sees a private email, for example, the server sees a variant of *403 Access denied*, i.e. a message informing him that the content behind this URL is only available after log in.

To account for these differences, we decided to also download the host of a web page. Usually the host of a web page's URL points to the home page of the website. The home page of a website is usually publicly available, so it will look the same to the user and to our server. Also, the home pages of websites may be more similar to the home pages of other websites from the genre than other web pages of that website. For example, the contact page of one website and the discussion board of another website may not exhibit similar features at all, but if both websites are social networks, their home pages will probably consist of a login form and a link for registration. Threfore, it is more easy to tell from the home page what genre the web page belongs to.

Since the host was not downloaded right away together with the web page itself, we downloaded all the hosts at once. They were downloaded on April 1st, 2014.

Data Sets and Corpora

We use different subsets of the WeKaSa data set for our research. For one, we use the web pages as instances for classification, and we consider their raw texts as a corpus of web documents.

The different raw texts from the web pages in our data set form a so-called corpus, which is essentially a large set of texts that can be used for analysis. For different parts of the analysis, either the full corpus or a subset is used. For the training phase, we will use the part of the corpus that consists only of the raw text from the web pages in our training set, whereas the full corpus contains the raw text from all web pages. We can also further divide the corpus according to genre, so a genre-specific training corpus would contain only the raw texts from web pages of one particular genre from the training data. Which of these subsets is used for a specific application will be explained using these terms.

We trained classifiers with different variants of the data set. There are several ways we can modify the data set we create based on the features we calculated from the web pages. We can choose to include all features or just a subset of them. A special case of that is that we can create a data set with just the web pages' feature vectors, or with a combined feature vector from each web page and its host. We can also choose if we want to include all genres or if we want to leave out the instances from specific genres. Additionally, we can map our genre set to a different one by specifying a different set of genre labels and a mapping from our genre labels to the different ones. These three ways to modify a data set - choosing a subset of attributes, choosing a subset of genres or mapping the genres to different ones - can also be combined with each other. The different data sets we use are described in the following.

Features were extracted for both the web page itself and its host. One data set includes just the web page's feature vectors, another uses a combined feature vector from web page and host as the feature vector for the web page.

We assume that web pages in *Other* do not actually form a genre on their own but rather need to be viewed separately in order to decide if these web pages can be sorted into the other genres or if new genres need to be added to our genre set. Therefore, we made one data set that does contain the web pages from Other and one that does not. The first is to test if our assumption is correct, the second to ensure that we get good results if our assumption is indeed correct.

If classification does not work well with our data set, we can use the two genre mappings described in 2.1.2 in order to simplify the classification problem: In one data set, we map the two genres Shopping1 and Shopping2 to one combined genre, Shopping and leave all other genres as they are. In the other data set, we map all genres to the risk levels they imply: Banking, Shopping1 and Shopping2 map to a high risk level. E-Mail, Data Exchange, Social Networks map to a medium risk level. Information Site maps to a low risk level. We hope to get better accuracy values for this simplified but still useful problem.

An overview of the different datasets we use can be seen in table 2.6. They are essentially characterized by three different properties: They can consist of instances that use only the web pages itself as instances (called parents), or they can combine the web page's information with that of its host into one instance. Therefore, we have parent datasets and parent-host-datasets. Since we use only web pages in German, the parent datasets contain all German web pages, while the parent-host-datasets contain only those web pages where both parent and host are in German. For this reason, there

Table 2.6.: Datasets we use in our experiments. Datasets can use just the web page itself (the parent) or both parent and host combined into one instance, use web pages from all genres or just from some of them and map web pages to a different set of genres. There is one dataset for training a classifier and another with the same properties on which that classifier is tested.

Pages	Genres	Mapping	Set	Instances
Parents	All		Training	2260
Parents	All		Test	391
Parents	All but Other	Risk Levels	Training	2100
Parents	All but Other	Risk Levels	Test	364
Parents	All but Other	Shopping combined	Training	2100
Parents	All but Other	Shopping combined	Test	364
Parents	All but Other		Training	2100
Parents	All but Other		Test	364
Parents & Hosts	All		Training	1398
Parents & Hosts	All		Test	243
Parents & Hosts	All but Other	Risk Levels	Training	1305
Parents & Hosts	All but Other	Risk Levels	Test	230
Parents & Hosts	All but Other	Shopping combined	Training	1305
Parents & Hosts	All but Other	Shopping combined	Test	230
Parents & Hosts	All but Other		Training	1305
Parents & Hosts	All but Other		Test	230

Table 2.7.: Distribution of the instances over the different genres for the datasets using web pages from all genres and no genre mapping.

Pages	Set	banking	data	info	mail	other	shopping1	shopping2	social
Parents	Training	69	55	1574	92	160	149	59	102
Parents	Test	10	9	276	14	27	26	11	18
Parents & Hosts	Training	14	19	1078	27	93	92	32	43
Parents & Hosts	Test	1	3	188	4	13	15	6	13

may be more instances in a parent dataset than in the corresponding parent-host-dataset. Furthermore, datasets can use instances from all the genres or leave out some genres. For our purposes, we created datasets containing all genres and datasets where web pages from Other are left out. Since we treat Other not as a genre by itself but rather a category to sort web pages into when we are not sure where they belong, web pages from Other can be seen as unlabelled data. They are useful to see if there are many web pages that do not fit into our existing genres, i.e. if we need to find different or more genres for our task. For the classification, however, it makes more sense to leave out web pages from Other and to concentrate on the remaining genres. The third property by which our datasets are characterized is which genre mapping they use, if any. As described in 2.1.2, a genre mapping maps some of the original genres to new genres, for example by combining Shopping 1 and Shopping 2 into Shopping (Shopping combined), or by mapping all genres to the risk levels they imply (Risk Levels).

For each variant of the datasets we described here, we have both a training and a test set. The training set contains approximately 85 % of the data and the test set contains the remaining 15 %. We split our full data into training and test before we did any actual work on the data, preserving the distribution of genres within the two sets. The distribution is approximately preserved in the subset of our data that consists of the German web pages, and it is also approximately preserved in the data sets we defined in the previous paragraph.

All the datasets we use and how many instances they contain can be seen in table 2.6. How many instances per genre the datasets contain can be seen in table 2.7 for the datasets using all genres and no genre mapping, in table 2.8 for the datasets using all genres but Other and no genre mapping, in table 2.9 for the datasets using all genres but Other and the risk level genre mapping and in table 2.10 for the datasets using all genres but Other and the shopping combined genre mapping.

Unfortunately, due to the restriction that all web pages need to be in German and if used, also their hosts, we end up with just one web page in the genre banking in test sets where pages and hosts are combined. This means we need to be careful with the results obtained by the evaluation on such a small set. It is possible that the accuracies on that set are not a good estimation for what they would be in reality.

Table 2.8.: Distribution of the instances over the different genres for the datasets using web pages from all genres but Other and no genre mapping.

Pages	Set	banking	data	info	mail	shopping1	shopping2	social
Parents	Training	69	55	1574	92	149	59	102
Parents	Test	10	9	276	14	26	11	18
Parents & Hosts	Training	14	19	1078	27	92	32	43
Parents & Hosts	Test	1	3	188	4	15	6	13

Table 2.9.: Distribution of the instances over the different genres for the datasets using web pages from all genres but other and the risk level genre mapping. The risk level genre mapping maps web pages from banking, shopping1 and shopping2 to high, web pages from data, mail and social to medium, and web pages from info to low.

Pages	Set	high	medium	low
Parents	Training	277	249	1574
Parents	Test	47	41	276
Parents & Hosts	Training	138	89	1078
Parents & Hosts	Test	22	20	188

Table 2.10.: Distribution of the instances over the different genres for the datasets using web pages from all genres but other and the shopping combined genre mapping. The shopping combined genre mapping maps web pages from shopping1 and shopping2 to one combined genre, shopping, and leaves the rest of the genres as they are.

Pages	Set	banking	data	info	mail	social	shopping
Parents	Training	69	55	1574	92	102	208
Parents	Test	10	9	276	14	18	37
Parents & Hosts	Training	14	19	1078	27	43	124
Parents & Hosts	Test	1	3	188	4	13	21

2.3 Features

If we want to find out what characterises web pages from a particular genre, we need to be able to compare web pages to each other. In order to do that, it is necessary to think about what aspects of the web pages we want to look at. Such aspects can be, for example: Does the web page contain pictures, and if so, how many? Is there a lot of text on the web page? Are there form fields where one can enter data? Choosing the right aspects is essential in trying to find common characteristics of web pages. From the point of view of a machine learning algorithm, a web page is nothing but a set of those aspects we choose to include in our analysis. Therefore it is important that we include all aspects that may be relevant in distinguishing our genres so that the machine learning algorithm can use them to compare the web pages.

In the context of Machine Learning, such aspects are called Features. We can describe each web page in terms of those features by calculating the web page's value for each feature and storing the value together with the feature's name (or some other identifier). Such a list of feature names and the corresponding value for the particular web page is that web page's Feature Vector. In order to train a classifier on our data, we need to calculate a feature vector for each web page in our dataset. This process of calculating the feature values for the web pages is called Feature Extraction. The set of feature vectors for the web pages is what we use as input for the machine learning algorithm. That way, the algorithm can use this information to compare the web pages and figure out what web pages from the same genre have in common in terms of those features.

The following paragraphs describe which aspects we choose to look at, i.e. which features we want to calculate for each web page. It is explained why we choose those features, based on earlier work in the field of web genre classification and on our own consideration of what is important for the security context where we want to apply our classifier eventually.

Essentially, three types of web page features were calculated: URL features, HTML features and text features. HTML features are features based on the HTML markup of the web page, like the count and frequency of certain HTML tags. Text features ignore the HTML markup and concentrate on the plain text instead. They include features like the count and frequency of punctuation marks or the appeareance of certain key words. URL features are explained in 2.3.1, HTML features are explained in 2.3.2 and Text features are explained in 2.3.3. A full list of all features can be found in table A.1.

Some of the features are independent of the web page's language, while others require the document to be in German and have been trained on German documents only. The URL and HTML features are language independent because they do not rely on the text. Some of the text features are also language-independent but most are not. Some of the language-dependent features can be generalized to more languages if we add a case distinction based on the language of the web page and perform the language-dependent calculations separately. This means we have to integrate Natural Language Processing tools for all the languages we want to cover.

2.3.1 URL Features

One set of features is calculated from the web page's URL. The URL features describe the URL's complexitiy in terms of path depth, domain depth and query parameters, and how many of the most frequent URL keywords from the genres it contains.

The first URL feature is the URL path depth. For the URL path depth, the number of path name levels is calculated. That is, for a URL like *https://www.tu-darmstadt.de/studieren/abschluesse/abschluesse_1.en.jsp*, the path name */studieren/abschluesse/abschluesse_1.en.jsp* is taken, and it is counted how many directory levels deep it goes - in this case, there are two levels. Essentially, the URL path depth is the number of / in the URL's path name minus one. This is the same as feature U1, *Depth of URL* from [LLK05] p. 1269.

The second URL feature is the URL domain depth. For the domain depth, it is calculated on which level of the domain name system hierarchy the URL's host is. We do not consider anything above second-level domains, i.e. a domain depth of zero corresponds to a second-level domain (e.g. *tu-darmstadt.de*), a domain depth of one to a third-level-domain (e.g. *www.tu-darmstadt.de*), and so on. This way, the domain path depth is essentially the number of dots in the URL's host minus one.

Both URL path depth and URL domain depth are included to capture a notion of the URLs complexity. How long and complex a URL is can indicate its function: Companies like e.g. Web Shop providers or banks usually have their own domain and maybe a subdomain for a particular function like log in, but their URLs do not usually have more than those two domain levels. It is expected that for a more specific function, the number of domain levels or path levels increases. Also, a special function on a general web page will usually not be at the top level but rather on a subdomain or within a subfolder of the web page. Functions like e.g. requesting a certain type of information, like a user's profile in a social network, require more information to be encoded within the URL in a structured way, therefore it is likely that this will also mean an increase in domain or path levels.

As the third feature, we record whether the URL contains any query parameters. Query parameters appear at the end of the URL after a question mark in the form *key=value*. If there exists a part behind the question mark in our URL, it is considered to contain query parameters. If not, it does not contain query parameters.

Whether the URL contains query parameters may indicate whether data was entered by the user, for example in a search request. Therefore, this feature may help in disinguishing genres. However, query parameters do not necessarily encode data the user entered. They are often used by Content Management Systems just to generate a URL for a web page, containing the ID the Content Management System has assigned to that page internally as a query parameter.

URL features four to eleven consist of a URL keyword count for each genre. This feature is similar to feature UL1 to UL35 from [LLK05] p. 1269 where the count for certain keywords from the training corpus was used as features. However, we obtain our URL keywords in a slightly different way. Additionally, we also include the file name and the domain area in our URL keywords, which in [LLK05] are considered in the features U4 and U5, p. 1269.

In order to obtain keywords from a URL, we split it into tokens by using every character that is neither a letter nor a number as a delimiter, and the part in between two delimiters as a token. Empty strings are discarded from the token list. The remaining tokens are what we consider to be our URL keywords.

For each genre, we extracted a list of all URL keywords from the training data appearing in this genre. These URL genre keywords are ranked according to their relative frequency within web pages of that genre minus their relative frequency in web pages from the full training corpus. This way, the URL keywords that are more frequent in the genre than in the full training corpus are ranked best. These URL keywords are considered to be best to distinguish a web pages from different genres.

Frequent URL keywords such as the domain ending, i.e. *de*, *com*, *net*, *org* were not excluded from the ranking, and neither was *www*. The reason is that either these URL keywords are so common among all genres that they score badly in the ranking and thus are not used as URL genre keywords in the end, or these keywords actually do help to distinguish genres and therefore should be included, although the latter is not expected to happen. However, the URL keywords *https* and *http* are omitted from the URL keyword lists, as they indicate an encrypted or unencrypted connection. Whether the connection is encrypted or not is one of the security measures we want to check based on what genre a web page belongs to, so we cannot use this information to distinguish between genres as well.

Why this is important is illustrated by the following example: Suppose we included the keywords *http* and *https* in our keyword list and found out that the occurence of *https* is indeed a good indicator that the web page belongs, for example, to Online Banking, while a web page with *http* belongs to Information Site. If the user now tries to do Online Banking via an unencrypted connection, the InUse scoring tool detects that the keyword *http* is present and the keyword *https* is not, so it assigns the label Information Site to this web page. Therefore, no warning will be shown that tells the user the connection is unencrypted - the tool assumes that the user is only looking for information when in fact they want to do bank transactions. Therefore, including the URL keywords http and https as a feature to distinguish between genres is not a good idea: It implies that the connection to a web page only needs to be encrypted if it is encrypted. This is essentially the same as not checking for an encrypted connection at all. Therefore, we remove http and https from the URL keyword list.

For each genre, the top 100 URL genre keywords from the ranked URL genre keyword list are used to calculate a URL keyword count score for a web page: Each appeareance of one of the URL keywords from the genre's list in the web page's URL is counted. The sum of appeareances of all the keywords from the genre's list in the URL are then taken as this web page's URL keyword count for the genre.

The URL keyword feature is included because when looking at URLs from web pages, they often give some information about the web page's function. Online Banking web pages, for example, often have terms like bank or banking in their URLs. Web pages where you can log in or log out often contain login or logout in the URL. By including keywords from the training corpus, we hope to find more such examples that characterize the genres well and to succesfully use these to distinguish between genres.

2.3.2 HTML Features

The HTML features use the web page's HTML DOM tree to get information about the web page's structure. The hypertext markup language (HTML) describes how the web page looks like and how it is structured. Since web pages from different genres are observed to show many differences in layout, looking at the HTML tags is likely to help in classifying them.

The most basic HTML features are the count of certain HTML tags and their frequency relative to the total count of tags from that set on the web page. The set of HTML tags we look at is H = {a, br, button, div, form, frame, frameset, h1, h2, h3, h4, h5, h6, hr, html, iframe, img, input, li, link, ol, option, p, script, select, style, table, tbody, td, textarea, th, thead, tr, ul}. These are expected to give a good indication of what the important elements are on the web page. For example, a platform for data exchange may contain many pictures, which shows in the HTML as a high count if IMG tags. A banking web page, on the other hand, will probably not contain many pictures. Rather, it will contain fields where the user can enter his login data, which can be observed in the HTML code by looking at FORM, INPUT, TEXTAREA and

similar tags. In order to be able to both use the absolute count to determine whether there are many of a certain kind of tag on a web page, we also use the relative frequency of the tags in order to see if there are some tags that are far more frequent than others. This way, we can e.g. find out both if there are many pictures on the page but also if pictures are many in relation to text (e.g. in P tags) or if, even though there are many pictures, there is still more textual information. In addition to simple tag counts, the number of password fields is also determined, i.e. the number of input fields of type *password*. This is especially important in our context, since for security purposes, it makes a lot of difference if the web page we look at is one where the user can log in or if it is not. Likewise, the presence or absence of the other FORM, INPUT etc. fields tells us if the user can enter private data on the web page or not.

Furthermore, all links are extracted from the web page and their total number is counted, and how many of them are external links, internal links, JavaScript links or email links. External links are links that reference a web page with a different host, while internal links reference a web page with the same host as the web page we calculate features for. JavaScript links reference JavaScript code, and email links link to an email address. A web page with many external links could be the search result from a search engine, while web pages with many internal links could be the search result for a product search in a web shop. Web pages where searching is not one of the primary functions will probably have less links in total. Also, web pages with a complex function such as shopping or banking might contain more script links than e.g. newspaper article. And a long list of email addresses is probably some sort of online address book rather than a shopping or banking web page. Therefore, links are expected to be a useful feature as well.

2.3.3 Text Features

The text features for each web page are calculated using the raw text extracted from that web page. They are included in our feature set to indicate the style in which a web page's text is written, which can differ greatly between genres.

One group of text features are punctuation features. For each punctuation mark in a set of punctuation marks, its count in the text is calculated and its frequency relative to the total number of punctuation marks from that set in the text. The total number of punctuation marks from that set in the text is also recorded as a feature. Punctuation marks have been found to be a good feature for genre classification before (cf. [LLK05] p. 2171 and p. 1275-1276), so we hope they perform well for our approach too. In a social network, for example, it is more likely to find many sentences ending in question marks or exclamation marks, while the more formal language of banking or shopping web pages will most likely contain more periods. Also, the number of commas within a text gives some indication to how complex the sentence structure is, because e.g. relative clauses are often separated by commas from the main clause. Therefore punctuation is included in the feature set. Punctuaction features do not depend on the language of the text. However, different languages may use different punctuation marks, so it may be useful to extend the list of punctuation marks we check for if we want to extend the approach to other languages.

Some text statistics like the number of sentences, the number of tokens and the number of types, i.e. pairwise distinct tokens, is also calculated. From that, the average number of tokens per sentence, the lexical diversity and the type token ratio of the text is calculated. This also tells us something about the style of a text and therefore is expected to be useful in determining the stylistic differences between texts from our genres. Also, the number of tokens tells us something about the length of the text, and in combination with e.g. the image count from the HTML tag, tells us something about what the web page primarily consists of, e.g. text, images, form fields.

The tokens from the text are assigned a Part-Of-Speech tag, i.e. a tag that describes whether the word is e.g. a noun, a verb, or a different type of word. and the count of each POS tag and its frequency relative to the total number of tokens is stored as a feature. POS tags are also used because they are expected to help distinguish the different styles of text in our genres.

Keywords

Also, for each genre keywords from the training corpus are used to calculate how many of those keywords appear in the new text. For keyword extraction, the tokens are converted to lower case and they are stemmed in order to normalize them because we want to treat the singular and plural form of a noun, for example, as one and the same. Additionally, punctuation is deleted from the token list before looking at the keywords, because punctuation is covered by its own set of features. The count of keywords found for each genre is used as a feature.

For each genre, we extracted a list of all keywords from the training corpus appearing in this genre. Like the URL keywords, these keywords are ranked according to their relative frequency within web pages of that genre minus their relative frequency in web pages from the full training corpus. This means that those keywords that are more frequent in the genre than in the full training corpus get a higher score.

From the ranked genre keyword list, the top 200 keywords are taken. A keyword count for each genre is then calculated by counting the number of times a keyword from the list appears in the text. The sum of occurences of all the keywords from the genre's list are then taken as this web page's keyword count for the genre.

Keywords are expected to be a good feature to distinguish genres because we observed certain keywords on web pages of specific genres: Web shops usually contain something like a shopping cart (*Warenkorb* or *Einkaufswagen* in German), a button to order something, information on shipping, and so on. Banking web pages, on the other hand, often contain information such as a short text on Online Banking security, request the user to enter a PIN and an account number, and so on. Social Networks contain keywords like the user's profile, friends or contacts, and similar words. Therefore, we hope to identify many such terms by looking at frequent words in the training corpus, and to use them to distinguish between genres.

Topic Model

From the training data, a topic model was calculated using Latent Dirichlet Allocation (LDA) as described in [BNJL03]. This was done outside of our tool, using JGibbLDA (cf. [PN08]) which implements Latent Dirichlet Allocation using Gibbs Sampling for Parameter Estimation and Inference.

The raw text of all the web pages in the training corpus was used to compute the topic model. Punctuation and newlines were stripped from the raw text, and all special characters were replaced by *. This is done once for the parent web pages, and once for their hosts, to keep the two separated but to also be able to calculate topic values for the hosts. The two topic models are then applied to the corresponding test instances, i.e. the parent topic model from the training corpus is applied to the parent instances in the test corpus, and the host topic model from the training corpus is applied to the host instances in the test corpus. This way, each instance from the training and test set gets a score for each of the 50 topics identified in the topic model. These 50 topic value scores are included in our feature set, one feature per topic. The value for each of these features is the score of that web page's raw text for the corresponding topic.

Looking at the topics' keywords, one can see, for example, keywords that indicate the web page is about travelling, or about banking. The former is probably a web page from Shopping2, i.e. where you can buy something that does not need to be delivered to you, like an air plane ticket or a hotel reservation. The latter is, of course, likely to be an Online Banking web page.

The topic model is not to be confused with the genre keywords: While the topics are extracted from the full training corpus independent of the web page's genre, the genre keywords are extracted for each genre from only the corpus of that genre. Therefore, both the topic models and the genre keywords result in keywords of some sort, but their meaning is different.

2.4 Learning Algorithms

This section describes which learning algorithms we will use in order to obtain classifiers which can predict the genre for new instances. These learning algorithms can all be applied to data sets where each instance is represented by its values for certain attributes, i.e. the feature vector. The learning algorithms use the training data to create a classifier, which can then predict the class for a new instance. In our case, the class attribute is the genre, so the classifier will predict the genre of new instances, i.e. web pages represented by their feature vector. Since each instance in the training data comes with a class label, this is a supervised machine learning problem, as opposed to unsupervised learning where the training data is unlabelled and the learning algorithm has to find structure in the data on its own.

There are different types of attributes, of which the two we use here are nominal and numeric attributes. A nominal attribute is one that can have values from a finite set of discrete values. The genre attribute, for example, is a nominal one: Its value can be one of the genre labels we have, i.e. it is an element of the set {banking, data, info, mail, shopping1, shopping2, social}. Numeric attributes are attributes that have a numeric value, for example an integer or real number. Contrary to nominal attributes, their values are not discrete but continous. While the genre attribute is a nominal one, all other attributes that we use are numeric ones.

We will use Weka (cf. [oW]) in order to train classifiers, so some information about the Weka implementation of these learning algorithms is given here as well.

2.4.1 Naive Bayes

The Naive Bayes classifier is a classifier that is based on Bayes' theorem. For a new instance represented by its feature vector, the Naive Bayes classifier calculates the probability for each class given the feature vector and then predicts the class with the highest probability. To do so, it assumes that all attributes are conditionally independent given the class. Applying Bayes' theorem, one can then easily calculate the probability that a new instance falls into a certain class given its feature vector: The estimated probability is simply the product of the probabilities estimated from the training data for the class given each attribute value. (cf. [JL95], p. 2)

The probabilities for each class given a specific attribute value are estimated from the training data. For nominal attributes, their frequency of occurence in the training data is taken. Numeric attributes are assumed to follow a normal distribution, and the mean and standard deviation of that normal distribution are estimated by taking the average and the standard deviation of that attribute's values in the training data. Different ways to estimate these probabilities from the training data are possible too. From these estimates, the probability for a class given the attribute value can be calculated for new instances. (cf. [JL95], p. 2-3)

Since the Naive Bayes classifier computes the probability for each class, it could also output a list of possible genres and their probabilities for the instance instead of just one genre label. This is useful for us because it provides us with more information and allows us to choose, for example, the second most likely genre if the first was Information Site but the user just started to enter their password on the web page, which implies they do not just look for information.

In our data sets, there are some features that do violate the independence assumption. Features that are not independent of each other are, for example, *password_field_count* and *input_tag_count*. When *password_field_count* is more than zero, it implies that *input_tag_count* is at least as high, since password fields are input fields of type *password*. Also, features that appear as an absolute count and as a relative frequency, for example *comma_count* and *comma_frequency*, are not independent from each other either. If there are only a few violations of the indepence assumption, this may not be a problem but if there are many, Naive Bayes will not work well (cf. [WBW05], p. 1).

In Weka, the method to estimate the probabilities for numeric attributes defaults to assuming a normal distribution, which is what we are going to use here. (cf. [Docb])

2.4.2 Averaged One-Dependence Estimators (AODE)

Averaged one-dependence estimators (AODE) is a classifier based on Naive Bayes, or, more generally, on Bayesian classifiers. It has weaker independence assumptions for the features than Naive Bayes. As the name suggests, it averages the predictions of several one-dependence estimators in order to predict the class for new instances. A one-dependence estimator is a k-dependence estimator with $k = 1$, and a k-dependece estimator (or k-dependence classifier) is a Bayesian classifier where each attribute can be dependent on at most k other attributes. For Naive Bayes, attributes must not be dependent on each other, so Naive Bayes is a 0-dependence classifier. (cf. [WBW05], p. 8-10 and cf. [Sah96], p. 2-3)

From all possible one-dependence classifiers, those are selected that have one attribute that all other attributes depend on. Additionally, when classifying an instance, only those models are used where the attribute value occurs at least m times in the training data. The predictions of the one-dependence classifiers are then averaged in order to get one aggregated prediction. (cf. [WBW05], p. 8-10)

Like Naive Bayes, AODE can output a probability for each class and not just predict the most probable one, so it can also be used to get a genre vector with probabilities for each genre. Additionally, it can be updated with new examples, which may prove useful in future work. (cf. [WBW05], p. 10-11)

In Weka, the default value for m is 1, which is what we are going to use (cf. [Doca]). AODE can only work with nominal attributes, so we need to apply a discretization filter to our test and training set in order to convert the numeric attributes into discrete ones.

2.4.3 Support Vector Machines (SVM)

Support Vector Machines (SVM) are a common choice in text classification problems. Good performance on web genre classification by an SVM was, for example, reported in [zES04], p. 266.

A Support Vector Machine (SVM) is a classifier that distinguishes between two classes. The simplest SVM is the linear one, which considers all instances to be points in multidimensional space and tries to find the hyperplane that best separates the points from one class from those of the other class. This means that it tries to find the hyperplane which has the maximum possible distance to the nearest points from either class, which is called the margin. Training an SVM means to solve the optimization problem of maximizing the margin. (cf. [Pla99], p. 2)

There also exist non-linear SVMs, which differ from the linear one in that they apply a different function to measure the distance or similarity between two vectors, the so-called kernel function. If the kernel function is linear, the SVM is, too, but the kernel function can also be a polynomial or something else. (cf. [Pla99], p. 3-4)

Sequential Minimal Optimization is a method to solve the optimization problem of maximizing the SVMs margin. It breaks the optimization problem down into smaller ones that can be solved analytically. Since they can also be solved quickly, SMO is an efficient way to train an SVM. (cf. [Pla99], p. 6 and p. 18)

Support Vector Machines can be generalized to distinguish between more than two classes. A possible approach is to use one SVM per class to distinguish that class from all others. Another approach is to train one SVM to distinguish each possible pair of two classes and then predict the class for the instance that is selected by the majority of those SVMs.

We use Sequential Minimal Optimization (*weka.classifiers.functions.SMO*) as implemented in Weka with a polynomial kernel (*weka.classifiers.functions.supportVector.PolyKernel*) to train the Support Vector Machine. The multi-class problem

is addressed by training one SVM for each pair of two classes and then combining them to output one prediction. (cf. [Docc])

In the following, this classifier will be referred to as *SMO*.

2.4.4 Repeated Incremental Pruning to Produce Error Reduction (RIPPER)

JRip is the Weka implementation of Repeated Incremental Pruning to Produce Error Reduction (RIPPER), which is a learning algorithm that produces a classifier consisting of a set of rules.

RIPPER learns rule sets for multiple classes in the following way: First, it sorts the classes by their number of instances in the data set in ascending order, i.e. the class with the smallest number of instances comes first. It then builds rules to distinguish between instances from the first class (positive instances) and all other instances (negative instances). After that, all instances that are covered by these rules are deleted and new rules are built to distinguish the second class from the rest, and so on. This way, the multi-class problem is turned into several two-class problems. (cf. [Coh95] and cf. [Docd])

To distinguish between positive and negative instances, rules are built by starting with an empty rule and then adding conditions to it. In each step, the condition with the highest information gain is chosen to be added to the rule. This is done until the rule covers no more negative instances. Then, each rule is pruned, i.e. the last conditions or conditions may be removed from it according to a pruning metric. Several rules are learned in this fashion until a stopping criterion is reached. Then, the rule set is optimized by generating two variants of each rule and selecting the one out of the three with the minimum description length for the final rule set. If after that there are still positive instances that are not covered by the rules yet, new rules are generated for these. Then, all rules that would increase the description length of the resulting rule set are deleted, and the rest is added to the rule set. Then, the same is done for the next two-class problem. (cf. [Coh95] and cf. [Docd])

Since RIPPER is a rule-based approach, the classifier that results from this learning algorithm is a set of rules. These rules specify a set of conditions and to which genre an instance belongs if it satisfies those conditions. This rule set can be intuitively understood by a human being, so looking at the rule set can tell us what the properties of the data are that make web pages belong to one genre or another, far better than any of the other classifiers could. Therefore, RIPPER can provide us some insight into the genres' properties. Also, the resulting rule set can easily be implemented directly in a Firefox extension, using standard comparison and boolean operators. Because of these advantages, we decided to include Weka's JRip in our analysis. If its results are good, it gives us more information about the properties of web pages in a genre, and the implementation of the classifier in a Firefox extension is straightforward and simple.

2.4.5 ZeroR

ZeroR is a classifier that assigns the most frequent class to all examples. We use it here to get a baseline for our classifiers but do not plan to use this classifier in any actual application like the InUse scoring tool - simply classifying all web pages as belonging to one genre essentially means that we do the same thing we can do now, but with more effort.

More information about the ZeroR implementation in Weka can be found in [Doce].

2.5 Training and Evaluation of the Classifier

Each data set contains the web pages' feature vectors as instances. The feature vector contains the features described in 2.3, plus the page ID and, of course, the genre. The genre is the so-called class attribute, i.e. the attribute which the classifier is supposed to predict for new examples. The page ID is removed from the data set with a filter before the data set is used to train a classifier. While we do not want the classifiers to use the page ID, we do want to preserve this information for later analysis, i.e. in order to see which web pages where misclassified. Therefore, we have to apply the filter so that the classifier does not use the page ID but we can output it for the genre predictions on the test and training set.

In a first step, we use the full training set in order to train classifiers and perform a stratified ten-fold cross validation for a first evaluation of our results. As a second step, we evaluate our results on the test set that was kept back for this purpose. The test set was not used for training or for improving the feature set, e.g. by providing keyword lists or by influencing the topic model. The results of the different classifiers on training and test set will be shown in section 4.1.1.

Additionally, we train classifiers on data sets with a uniform genre distribution. The reason for this is that the genre distribution in our data sets is very skewed, which may pose a problem to the learning algorithm. To see if that is the case, we compare the results of classifiers on the standard data sets and the ones with a uniform genre distribution, which can be found in section 4.1.2.

We also perform feature selection for some of the data sets. There are two reasons why Features Selection may improve our results: Firstly, if we can concentrate on only the most useful features, we reduce the computation time of the web page's feature vector and thus make the classification of a web page in the InUse scoring tool faster. And secondly, feature selection often improves a classifiers accuracy because it removes noise features, i.e. it helps the learning algorithm to concentrate on the essential features and therefore leads to better results.

For Feature Selection, we choose those data sets that look the most promising for our approach based on their test accuracy on the standard data sets. For those data sets, we rank all the features according to their Chi Square metric and then create new data sets that consist of only the top ten percent of the features. We then train new classifiers on the data sets with only the selected features and compare their results to those of the classifiers trained on data sets without feature selection. The accuracies of the classifiers after feature selection can be found in section 4.1.3, and the list of selected features in section 4.2.

We will then shortly sum up the results of all the different classifiers and select the most promising ones for further analysis. For this, we take into account both the classifiers trained on standard and on uniform data sets, and those with and without feature selection. The summary of the results and which classifiers we selected is explained in section 4.1.4.

As a next step, we will look in more detail at the predictions made by the selected classifiers. We analyse their confusion matrices and look at the instances they misclassified. Furthermore, we analyse their confidence values for both correctly and incorrectly classified web pages. These results can be found in section 4.1.5.

So far, we compare classifiers by accuracy and assume that the classifier with a higher accuracy is better for our purposes. If accuracy values are close together, however, it is difficult to say if one of the classifiers performs significantly better than the other or if no real difference between the two can be determined. To test for significant differences in accuracy, we perform a paired t-test for selected classifiers on selected data sets. The t-test divides the given data set into several partitions and estimates accuracy on unseen data by training the classifiers several times on part of the data set and evaluating them on another part. Since we cannot specify a training and test set directly for this, we use a set that consists of both training and test data. This is the only case where we mix the two sets together. This way, both the web pages that helped in creating the feature set, e.g. by keyword lists, and the ones that did not will influence the results of the t-test. The t-test will tell us which of the classifiers performs significantly better than the others on a given data set and thus give us a better idea of which one is best suited to the task. The results for the t-test can be found in 4.1.6.

Based on all these results, we identify points which can lead to an improvement of the classifiers' performance and present them in section 4.5.

3 Implementation and Architecture

After we presented our approach, we now explain how we implemented the different parts of the approach and which external tools and libraries we use. Since the approach requires many different pieces of software to work together, we want to give readers an overview of what those pieces of software are and how they are connected. Furthermore, we want to ensure that our results are reproducible for others as well, so it needs to be clear how exactly we computed the features, classifiers and everything else that is used in our approach.

The Feature Extraction is split into two parts. One is concerned with the extraction of text features from the raw text of the web page, such as the frequency of certain POS tags, type-token-ratio and similar features. The other part uses the web page's URL and the HTML markup to extract features such as the frequency of certain HTML tags, the URL depth and others.

Since the system is eventually meant to be integrated into a Firefox browser extension and since a Firefox extension can use the HTML parsing capabilities implemented in the Firefox browser to calculate HTML features easily, the second part is implemented as a Firefox extension. The first part cannot be implemented so easily within a Firefox extension since the necessary Natural Language Processing functions are not available as JavaScript libraries and therefore would have to be implemented by hand. To avoid this, a web service based on Python NLTK is used to calculate the text features. The Firefox extension send the web page's raw text to this web service and gets a vector with the text features of that text in return, which it can combine with the URL and HTML features it calculated itself. This works well for a research prototype and to find out which features work best to distinguish the different web genres. These results can then be used to later on implement exactly those functions that are really needed in JavaScript, since the web service approach would not be good in practice because of privacy considerations.

An average calculation time of about 815 milliseconds per page was recorded for a set of 9809 web pages from the WeKaSa data set. One page in this context means either the web page tagged with a genre or its host. The three web pages for which it took the longest to compute the feature vector had calculation times of 825870 milliseconds (about 14 minutes), 50376 milliseconds (about 50 seconds) and 42188 milliseconds (about 42 seconds). In those cases, the calculation time was definitely too high. The minimum calculation time we observed was 15 ms. On average calculation times are acceptable. The cases with very high computation times should be further analysed to find out why it took so long to compute their feature vectors, which is left for future work.

As described in section 2.3.3, topic features are calculated by an external tool called JGibbsLDA (cf. [PN08]). They are imported into Fea by providing a file with the topic values and a file which specifies which line in the topic file corresponds to which page ID. Eventually, the estimation of topic model values for new web pages needs to be done within a Firefox extension in order to be useful for the InUse scoring tool. Since we first want to evaluate which features work best for our tasks, this is left for future work if the topic model features work well in distinguishing the genres.

The training and evaluation of the classifier is also done outside of the Firefox extension. For this purpose, Weka is used, a Data Mining Software from the University of Waikato (cf. [oW]). It provides the possibility to load a data set, train different classifiers on this data and evaluate their performance, so it offers all the functionality we need. Hence, the Firefox extension has a function to export the feature vectors in the Attribute-Relation File Format that is recognized by Weka, so the data we collected can be used in Weka easily.

Once classifiers are trained and evaluated, the best classifier can be selected and implemented in a Firefox extension, which is left for future work. Note that only the classification itself and not the training and evaluation of the classifier needs to be part of the Firefox extension. Ideally, this can be implemented directly in the Firefox extension, for example using existing JavaScript libraries that provide the functionality to train and load a classifier, for example [Kar12] for a Support Vector Machine. Alternatively, one can try to include Weka directly into a Firefox extension because it is possible to include Java files inside a Firefox extension.

3.1 HTML and URL Features

The Firefox extension Fea is used to extract features from a web page. It works both on the web pages stored in a database by WeKaSa and on the web page currently watched in the browser. This way, the feature extraction can be integrated into the tool that will later on analyse the web page the user is loading in the browser.

Fea calls several webservices for the feature extraction: For the training phase, Fea needs to get the web pages from WeKaSa, which are accessible via a webservice. Fea needs to retrieve the web page's HTML and its meta data (URL, page ID, and so on) separately. When the web page's HTML and meta data are stored, Fea starts calculating features for that web page. The HTML and URL features are calculated locally, and for the text features, another webservice is called. The topic model data is read from a file, where each line corresponds to a web page and the n-th entry in that line is the topic value for topic n, with n between 0 and 49. The feature vector is complete when all locally calculated features - HTML,

URL and Topic Model - are computed and the response from the NLP webservice with the text features has been received and stored. When the feature vector is complete, it is stored locally in an SQLite database, along with some of the web page's meta data, like the URL and the page ID.

When all web pages have been processed, Fea can perform several post-processing steps necessary for the training phase: Keyword Extraction, URL Keyword Extraction, Keyword and URL Keyword Ranking, Raw Text Export and Export to ARFF.

Fea can extract all the keywords from the web page's raw text, using the NLP webservice to filter out punctuation and stem the words in order to normalize them, and to count how often each word appears in the text. Fea can then calculate how often they were used in each genre and in the full corpus. Based on that, a keyword ranking for each genre is computed: The keywords are sorted in descending order according to the difference between their relative frequency within the genre and their relative frequency in the full corpus. The motivation behind this metric is that the keywords that appear much more frequently in a certain genre than in the full corpus are most likely to be good indicators for what genre a text with many of these keywords belongs to. The top 200 words from the keyword ranking are then stored in a file for each genre, which can then be used to calculate each web page's genre keyword count, for example online_banking_keyword_count. The chosen keywords are stored as an array in the NLP webservice's code.

Something similar is done for the URL keywords. Each web page's URL is split into tokens. Token delimiters are all characters allowed in an URL that are neither letters nor numbers. Empty strings do not count as tokens and are deleted. Unlike the keywords taken from the text, the URL keywords are not stemmed or otherwise normalized. This is because even on German web pages, parts of the URL need not necessarily be German words or even be words at all, so using a German stemmer on them would not make much sense. They are, however, decoded before they are split into tokens, i.e. escaped special characters are replaced by the corresponding character. Just like with the keywords taken from the web pages' raw text, all URL keywords are collected for each genre and their relative frequency within that genre and within all pages is calculated. Again, the difference between those two values is used to rank the keyword's usefulness in identifying the genre in question. The URL keyword ranking is then stored in a file for each genre.

From that ranking, the keywords *http* and *https* are deleted because they indicate whether the connection to the web page is encrypted. Since an encrypted connection is one of the security measures we want to check for later based on the genre, information about the connection's encryption should not be part of our features in any way.

Other keywords common to many web pages, like www or domain endings like de, net, com are kept, because they might not be useful in distinguishing genres but also do no harm, so they are left there in case they may be unexpectedly useful. The top 100 URL keywords are stored in an array in Fea's code.

Fea can also export the raw texts as a whole and all texts from each genre. They are stripped of punctuation and newlines and are then stored in one file for all texts and one for each genre. Each line in such a document corresponds to one web page in our data set. The order of the web pages is stored in a separate file as a list of the web pages' page IDs, and the file also contains the path to the file with the actual text it belongs to. The file containing all texts is necessary for calculating the topic model, which is done outside of Fea and the NLP webservice. In order to later import the calculated topic, the order is important: The topic model calculation outputs a file with one line for each of the web pages from the input file, which can then be imported into Fea. The topic values can then be assigned to the correct page's feature vector by using the page ID list from the order file. This page ID list must be provided to Fea as an array in the corresponding function that loads the topic model.

When all web pages' feature vectors are stored in the Fea SQLite database, they can also be exported to the Attribute-Relation File Format. This format is used by Weka in order to describe datasets, i.e. this is the file we need to let Weka load the feature vectors from our training data and train a classifier on it.

3.2 Text Features

The Feature Extraction webservice calculates all text features. It has several functions and defines a URL for each of them. Its main function is to calculate a feature vector and some meta information when provided with a web page's raw text (or any raw text, for that matter). This function is available if you call the webservice's base URL. When adding /keywords/ to the base URL, the webservice will use the provided text and return a frequency distribution of the text tokens. Adding /keyword_ranking/ to the base URL yields a list of the keywords and their ranking according to the difference in relative frequency between texts from the corresponding genre only, and all texts. Since the webservice is stateless, both the genre's texts and all texts need to be provided for this function. Finally, attaching /language/ to the base URL and providing a text will answer the question if the text's language is German or not.

The Fea Web Service is implemented in Python. For the Natural Language Processing tasks, it uses several libraries, namely the Natural Language Toolkit (NLTK) (cf. [NLT13]), Pattern (cf. [Com14]) and langid (cf. [Lui]). In order to create a web service, Flask (cf. [Ron14]) is used.

To compute the features described in 2.3.3, several Natural Language Processing tasks need to be done. These tasks include tokenization into words and sentences, computation of frequency distributions, POS tagging, stemming, language identification and usage of a list of German stopwords.

In order to tokenize the text into words, the WordPunctTokenizer from NLTK is used. Sentence tokenization is done by using the Punkt sentence detector from NTLK. NLTK is also used to compute frequency distributions of the tokens. POS tagging of the text is done with pattern.de from Pattern. Stemming of the tokens is done with the Snowball German stemmer from NLTK. The text's language is identified with langid, and with a method we implemented ourselves that uses German stopwords from NLTK and decides whether a text is German or not based on the German stopword count in that text.

For the decision which texts we want to consider as German texts and use in our classification, the output from langid was used. The other method was implemented because it does not rely on Natural Language Processing libraries but only needs the list of stopwords from the language we want to identify, so it can be easily implemented in JavaScript as well, i.e. it can be used in Firefox extensions. Compared to the results from langid, it identified less texts as being in German but we found no texts it believed to be German that were actually in another language. Id did identify some texts as German that langid did not, but these all proved to be borderline cases, e.g. web pages with barely any text, or the menu of an American-style restaurant that listed the English names of all the dishes while the rest of the text was in German. In general, neither langid nor our method seemed to identify any texts as German that were not, but langid identified more texts as German texts, so langid's output was more useful for our research in this thesis.

3.3 Topic Model Features

The topic model was calculated outside of Fea and the Fea webservice by using an external tool called JGibbLDA (cf. [PN08]), which is described in more detail in section 2.3.3. It uses as input all texts from the web pages in the training data, stripped of punctuation and newlines and saved to a file with one line corresponding to one web page.

Since the topic values are calculated outside of our tool, Fea's export functionality is used to export the raw texts from the web pages into the input format required by JGibbLDa. Fea also exports the page IDs in the order the raw texts are included in the input file for JGibbLDA.

The topic model that is calculated is then available as a file where each line corresponds to the topic values for the web page on the same line in the input file. Having saved the web pages' order when exporting them, the topic model can then be imported into Fea again to add the topic values to the web pages' feature vectors.

3.4 Access to Web Page Collection

The original WeKaSa web service was modified to allow some additional functions needed to process the data from the WeKaSa data set. WeKaSa pages allowed to inspect the stored page's HTML in the browser by providing the script with the page ID. This way, the many examples where Information Sites seem to be Web shops came to light as well. In order to crawl the hosts corresponding to the web pages we have stored, WeKaSa hosts was created, which allows to crawl the corresponding hosts when given a list of page IDs from parent pages. WeKaSa Fea is then used to get the WeKaSa data from the database into the Fea Firefox Extension. The Fea Firefox extension can query the WeKaSa Fea webservice in order to get a list of all pages to crawl or the HTML or meta information for a specific page ID. This is how the data collected by WeKaSa can be processed by Fea and turned into a set of feature vectors.

4 Presentation and Discussion of the Results

This section describes how good the results of the different classifiers were, compared in terms of their accuracy, and which feature sets worked best to determine a web page's genre.

4.1 Classifiers

This section presents and discusses the results obtained by training classifiers on our different data sets. Classifiers are compared with regard to their accuracy on the test and training set. Furthermore, we look at the confusion matrices of the best classifiers and at the web pages they misclassified in order to see where their problems lie.

4.1.1 Accuracy on Standard Data Sets

As explained in 2.5, we first trained all classifiers on the training data and performed a stratified ten-fold cross-validation to get an estimate for the accuracy. We then evaluated them on the corresponding test set. The resulting accuracies can be seen in table 4.1, where the training accuracy is the one from cross validation on the training and test accuracy is the one from the evaluation on the test set. For each dataset, classifiers are sorted in descending order according to their accuracy when evaluated on the test set. The row showing the result for ZeroR is colored in gray in order to visualize which classifiers ranked below and which ranked above the baseline for the dataset. A ranking of the top ten classifiers, i.e. the ones that achieved the best test accuracies, can be seen in table 4.1a.

ZeroR assigns the most frequent class to all instances and thus gives us an accuracy baseline. In our dataset, the most frequent genre is Information Site, so ZeroR classifies all web pages as Information Site.

The results for classfiers on data sets with all genres including Other are shown in table 4.1 as well. As expected, they did not achieve good results: Their best test accuracy is below the best test accuracy of all other data sets. In the following, we will therefore concentrate on those data sets which do not include the web pages classified as Other.

No matter on which dataset, Naive Bayes always achieves an accuracy below the baseline. Hence we can assume that Naive Bayes does not work well for our application. This may be the case because the indenpence assumption is violated by many of the features we choose. If that is the case, it is possible that Naive Bayes shows better accuracy after Feature Selection. Another potential problem for Naive Bayes is that our dataset is rather small, so it is possible that we have not enough data for a simple classifier to work well and need a more complicated classifier.

In general, the best accuracies achieved on the Parents & Hosts data sets are higher than those achieved on the corresponding Parents data set. Also, eight out of ten classifiers that ranked among the top ten were classifiers trained on Parents & Hosts data sets. Therefore, we can conclude that including the web page's host helps to classify the web page correctly.

However, there are less instances in the Parent & Hosts data sets than in the Parents data sets because of the requirement that both the parent and host web page have to be in German. Especially in the test sets with Parents & Hosts, there may not remain a sufficient amount of web pages for each genre in order for the results to be meaningful. Note that we did not apply any kind of cost function, i.e. misclassifying the one Online Banking instance in the Parents & Hosts test sets is treated as equally good or bad as misclassifying one of the almost 200 instances in Information Site, and it does not make a difference which genre is misclassified as which other genre. Therefore, we will look at this in more detail in the next section and analyse the misclassifications of the different classifiers.

The highest accuracy of 86.9565 % on the test set is achieved twice, each time on a Parents & Hosts data set: Once by AODE on the data set with Risk Level mapping and once by SMO on the data set with no genre mapping. SMO achieves a slightly better training accuracy than AODE, and of course we prefer the data set with no mapping to the one using the Risk Level mapping because the former gives us more information about the web page context.

If we compare training and test accuracies within the top ten classifiers, we see that the classifiers have a smaller test accuracy than training accuracy, which is normal. For the best two classifiers, the difference is less than one percentage point and for the rest of the classifiers, the difference is between 2 and 3.3 percentage points. This is good sign that the classifiers do not overfit the training data too much, i.e. they do actually generalize from the training data and so can be applied to unseen examples with good results.

Looking closer at the ranking of the top ten classifiers, we see that it there are three algorithms tpresent there: SMO, AODE and JRip. Five out of the ten best classifiers are SMO classifiers, three are AODE classifiers and the remaining two are JRip classifiers. SMO achieved good results on the Parents & Hosts data sets with either mapping or no mapping and slightly less accuracy on the Parents data sets with either the Shopping Combined or the Risk Level mapping. This makes SMO the only classifier that ranked in the top ten on a Parents data set as well, not just on Parents & Hosts data

Table 4.1.: Accuracies of the classifiers on test and training set, sorted by test accuracy and then training accuracy within each dataset. One dataset is specified by the pages it uses, the genres that were included when selecting instances and the genre mapping that was used. Datasets are separated by horizontal lines in this table. The lines showing the baseline value, i.e. the results of the classifier ZeroR, are colored in gray in order to visualize which classifiers score below or above the baseline within each dataset.

Pages	Genres	Mapping	Algorithm	Accuracy Training [%]	Accuracy Test [%]
Parents	All		SMO	79.469	78.5166
Parents	All		JRip	76.8584	76.2148
Parents	All		AODE	75.6637	73.913
Parents	All		ZeroR	69.646	70.5882
Parents	All		NaiveBayes	38.0531	37.8517
Parents	All but Other	Risk Levels	SMO	86.4762	84.6154
Parents	All but Other	Risk Levels	JRip	83	81.8681
Parents	All but Other	Risk Levels	AODE	82.7143	81.5934
Parents	All but Other	Risk Levels	ZeroR	74.9524	75.8242
Parents	All but Other	Risk Levels	NaiveBayes	59.0476	57.1429
Parents	All but Other	Shopping Combined	SMO	85.8571	83.7912
Parents	All but Other	Shopping Combined	JRip	82.7143	82.6923
Parents	All but Other	Shopping Combined	AODE	82.7619	81.5934
Parents	All but Other	Shopping Combined	ZeroR	74.9524	75.8242
Parents	All but Other	Shopping Combined	NaiveBayes	50.3333	51.3736
Parents	All but Other		SMO	85	82.967
Parents	All but Other		AODE	82.1429	80.2198
Parents	All but Other		JRip	82.1905	78.8462
Parents	All but Other		ZeroR	74.9524	75.8242
Parents	All but Other		NaiveBayes	47.8571	50
Parents & Hosts	All		SMO	82.0458	82.3045
Parents & Hosts	All		JRip	78.97	81.07
Parents & Hosts	All		ZeroR	77.1102	77.3663
Parents & Hosts	All		AODE	79.1845	76.5432
Parents & Hosts	All		NaiveBayes	47.8541	47.3251
Parents & Hosts	All but Other	Risk Levels	AODE	87.433	86.9565
Parents & Hosts	All but Other	Risk Levels	SMO	87.9693	84.7826
Parents & Hosts	All but Other	Risk Levels	ZeroR	82.6054	81.7391
Parents & Hosts	All but Other	Risk Levels	JRip	83.0651	76.5217
Parents & Hosts	All but Other	Risk Levels	NaiveBayes	64.9042	70.4348
Parents & Hosts	All but Other	Shopping Combined	SMO	88.1226	86.087
Parents & Hosts	All but Other	Shopping Combined	AODE	88.1226	85.6522
Parents & Hosts	All but Other	Shopping Combined	JRip	85.0575	83.4783
Parents & Hosts	All but Other	Shopping Combined	ZeroR	82.6054	81.7391
Parents & Hosts	All but Other	Shopping Combined	NaiveBayes	64.8276	66.087
Parents & Hosts	All but Other		SMO	87.7395	86.9565
Parents & Hosts	All but Other		AODE	88.046	84.7826
Parents & Hosts	All but Other		JRip	84.4444	83.0435
Parents & Hosts	All but Other		ZeroR	82.6054	81.7391
Parents & Hosts	All but Other		NaiveBayes	64.751	68.2609

Table 4.2.: Top ten classifiers on the non-uniform data sets, ranked by test accuracy and then training accuracy. A diagram visualizing this table can be seen in figure 4.1a.

Rank	Pages	Genres	Mapping	Algorithm	Accuracy Training [%]	Accuracy Test [%]
1	Parents & Hosts	All but Other		SMO	87.7395	86.9565
2	Parents & Hosts	All but Other	Risk Levels	AODE	87.433	86.9565
3	Parents & Hosts	All but Other	Shopping Combined	SMO	88.1226	86.087
4	Parents & Hosts	All but Other	Shopping Combined	AODE	88.1226	85.6522
5	Parents & Hosts	All but Other		AODE	88.046	84.7826
6	Parents & Hosts	All but Other	Risk Levels	SMO	87.9693	84.7826
7	Parents	All but Other	Risk Levels	SMO	86.4762	84.6154
8	Parents	All but Other	Shopping Combined	SMO	85.8571	83.7912
9	Parents & Hosts	All but Other	Shopping Combined	JRip	85.0575	83.4783
10	Parents & Hosts	All but Other		JRip	84.4444	83.0435

sets. AODE also achieves good results on the Parents & Hosts data sets with either mapping or no mapping. On those three, AODE achieved the best accuracy on the one with Risk Level mapping, then on Shopping Combined and then on the data set with no mapping. For SMO, the order is the other way around: Its best accuracy is on the data set with no mapping, the second best on the one with Shopping Combined mapping, and the third best on the one with no mapping. Hence SMO seems to work better for the data sets with more information and AODE seems to work better for those with less information. JRip occupies the last two ranks of the Top Ten, both on the Parents & Hosts data sets, once with the Shopping combined mapping, and then with no mapping.

However, all the accuracy values within the Top Ten are quite close together: The best test accuracy and the worst test accuracy in the top ten only differ by 3.913 percentage points. Therefore, the exact order of the classifiers in the ranking may be somewhat coincidental and could be different if the classification problem is varied a little. In order to see if significant differences between any of the classifiers can be found, we will evaluate their performance on the data set with no mapping with a paired t-test. The results will be presented in 4.1.6.

4.1.2 Accuracy on Data Dets with Uniform Genre Distribution

The baseline set by ZeroR is rather high, which is due to the fact that our datasets contain much more Information Sites than instances of any other genre. Therefore, we created new data sets from the existing ones which contain an approximately equal number of instances from each genre. We hope to see the differences in accuracy between the classifiers more clearly if the classifier does not have the option anymore to simply assign the most frequent genre if in doubt. Additionally, we do not know whether the distribution of genres within our training data is representative for their distribution in the real world, so it does make sense to train a classifier which cannot take advantage of this distribution.

We created a corresponding uniform data set for each of the data sets we analysed so far, one for each training set and one for each test set. We used the same approach as before and trained all classifiers on each of the uniform training sets, performing a cross validation to estimate the accuracy, and then evaluated them on the corresponding uniform test set. The results can be seen in table 4.3.

Like in the standard data sets, accuraccies on the uniform data sets with all genres are lower than those of the other data sets. Just like for the standard data sets, this was the expected result. Therefore, we do not further discuss those classifiers in the following.

On the uniform data sets, the classifiers' accuracies on the test set decrease notably compared to the accuracies on the standard data sets. On the standard data sets, a maximum test accuracy of 86.9565 % was achieved while on the uniform data sets, the maximum test accuracy is 81.3187 %, which is 5.6378 percentage points lower. This accuracy was only achieved on the Parent data set using the risk level mapping, and it is the only accuracy on a uniform data set that is greater than 80 %. On all other uniform data sets, the accuracies are even lower than that. The maxima for the different data sets range from about 55 to 78 %. Surprisingly, the accuracies on the training set are better than before: While the maximum training accuracy on the standard data sets is 88.1226 %, it is 95.4023 % on the uniform data sets, which is 7.2797 percentage points higher. While on the standard set, training accuracies usually are about the same as test accuracies, there is much difference between these values on the uniform data sets: Looking at just the best classifier for each data set, we can observe differences that range from around 6 to almost 40 percentage points between training and test accuracy. This suggest that on the uniform data sets, the classifiers are overfitting the training data far more than their equivalents trained on the standard data sets.

Table 4.3.: Accuracies of the classifiers on uniform training and test set. For AODE, many values could not be calculated because time and space requirements were too high.

Pages	Genres	Mapping	Algorithm	Accuracy Training [%]	Accuracy Test [%]
Parents	All		SMO	83.2743	63.1714
Parents	All		NaiveBayes	63.4956	53.4527
Parents	All		JRip	86.3717	52.1739
Parents	All		ZeroR	14.0708	14.3223
Parents	All but Other	Risk Levels	SMO	87.5238	81.3187
Parents	All but Other	Risk Levels	AODE	93.5714	74.1758
Parents	All but Other	Risk Levels	NaiveBayes	73.1429	65.3846
Parents	All but Other	Risk Levels	JRip	87.1429	63.1868
Parents	All but Other	Risk Levels	ZeroR	36.1905	36.8132
Parents	All but Other	Shopping Combined	SMO	89.7143	68.4066
Parents	All but Other	Shopping Combined	NaiveBayes	70.4762	56.8681
Parents	All but Other	Shopping Combined	JRip	90.7143	56.5934
Parents	All but Other	Shopping Combined	ZeroR	18.3333	18.6813
Parents	All but Other		SMO	90.381	77.7473
Parents	All but Other		JRip	90.381	64.5604
Parents	All but Other		NaiveBayes	71.8095	59.8901
Parents	All but Other		ZeroR	15.2857	10.4396
Parents & Hosts	All		SMO	90.701	59.6708
Parents & Hosts	All		JRip	88.6266	46.5021
Parents & Hosts	All		NaiveBayes	80.5436	46.0905
Parents & Hosts	All		ZeroR	14.5923	11.9342
Parents & Hosts	All but Other	Risk Levels	AODE	95.4023	65.2174
Parents & Hosts	All but Other	Risk Levels	SMO	94.1762	65.2174
Parents & Hosts	All but Other	Risk Levels	NaiveBayes	77.1648	64.3478
Parents & Hosts	All but Other	Risk Levels	JRip	91.341	52.1739
Parents & Hosts	All but Other	Risk Levels	ZeroR	35.7088	35.6522
Parents & Hosts	All but Other	Shopping Combined	JRip	93.8697	55.2174
Parents & Hosts	All but Other	Shopping Combined	SMO	95.0958	52.6087
Parents & Hosts	All but Other	Shopping Combined	NaiveBayes	88.4291	43.4783
Parents & Hosts	All but Other	Shopping Combined	ZeroR	17.931	17.8261
Parents & Hosts	All but Other		SMO	94.4828	73.913
Parents & Hosts	All but Other		JRip	93.41	65.6522
Parents & Hosts	All but Other		NaiveBayes	89.1954	47.8261
Parents & Hosts	All but Other		ZeroR	15.4789	13.913

Table 4.4.: Top ten classifiers on the uniform data sets, ranked by test accuracy and then training accuracy. A diagram visualizing this table can be seen in figure 4.1b.

Rank	Pages	Genres	Mapping	Algorithm	Accuracy Training [%]	Accuracy Test [%]
1	Parents	All but Other	Risk Levels	SMO	87.5238	81.3187
2	Parents	All but Other		SMO	90.381	77.7473
3	Parents	All but Other	Risk Levels	AODE	93.5714	74.1758
4	Parents & Hosts	All but Other		SMO	94.4828	73.913
5	Parents	All but Other	Shopping Combined	SMO	89.7143	68.4066
6	Parents & Hosts	All but Other		JRip	93.41	65.6522
7	Parents	All but Other	Risk Levels	NaiveBayes	73.1429	65.3846
8	Parents & Hosts	All but Other	Risk Levels	AODE	95.4023	65.2174
9	Parents & Hosts	All but Other	Risk Levels	SMO	94.1762	65.2174
10	Parents	All but Other		JRip	90.381	64.5604

Contrary to what we observed on the standard data sets, the Parents & Hosts data sets do not seem to be advantageous over the Parents data sets here. Only five out of ten classifiers from the top ten ranking use the Parents & Hosts version of a data set now, and the other five use the Parents only version. Rank one to three is occupied by classifiers trained on Parent data sets.

On the uniform sets, all classifiers perfom better than the baseline set by ZeroR. Assigning the most frequent class is not so much of an advantage anymore when none of the classes is far more frequent than the others. Even Naive Bayes is now not only above the baseline but also performs better than JRip on three of the eight data sets.

On six out of eight data sets, SMO achieved the best test accuracy.

For AODE, not all the values could be computed on the uniform data sets because for the computation it required too much time and space on a normal desktop computer. Since the run time of the classification of new instances is essential for our application, AODE is probably a bad choice for our application, at least when using the full feature set. Training and classification time may improve when there are less features to be considered, so we take a further look at AODE's results after feature selection. In the two cases where the values could be computed, AODE achieved good results compared to the other classifiers: On the Parents data set with Risk Level mapping, it ranked second, and on the Parents & Hosts data set it had the best test accuracy of all classifiers on that data set. However, on the data set where AODE came first, its accuracy is almost ten percentage points lower than on the one where it was the second best classifier.

Table 4.4 shows the top ten classifiers on the uniform data sets, ranked by test accuracy and then training accuracy. This is visualized as a bar chart in figure 4.1b. This time, we can find AODE, JRip, Naive Bayes and SMO among the top ten. On the standard data sets, Naive Bayes was not included in the top ten, but the other three were. Like before, five out of ten of the top ten classifiers are SMO classifiers. These are not necessarily the same ones as for the standard data sets, i.e. SMO performed well but partly on different data sets. Of the remaining five, two are AODE classfiers, two JRip classifiers, and there is also one Naive Bayes classifier among the top ten.

As mentioned before in this section, we see that the highest accuracy on a uniform test set is 81.3187 %, while it is 86.9565 % on the standard data sets, so they differ by 5.6378 percentage points. However, the best classifier on the standard data set is SMO for the Parents & Hosts data set with no genre mapping while on the uniform data sets it is SMO on the Parents data set with the Risk Level mapping, which is less useful for our application since it provides us with less information about the context. SMO on the Parents data set with no genre mapping occupies rank two for the uniform classifier ranking with a test accuracy of 77.7473 %, which is 3.5714 percentage points less than the maximum on the uniform sets and 9.2092 percentage points less than the maximum on the standard data sets.

The six classifiers that appear among the top ten in both the standard and uniform rankings are: SMO and JRip on the Parents & Hosts data set with no genre mapping, SMO and AODE on the Parents & Hosts data set with the risk level mapping, and SMO on the Parents data set with Risk Level mapping and with Shopping Combined mapping. This means there are four SMO classifiers, one AODE classifier and one JRip classifier in the top ten in both rankings. Four out of the six use Parents & Hosts data sets, and two use Parent data sets. The risk level mapping appears three times, no mapping appears twice and the shopping combined mapping once.

4.1.3 Accuracy After Feature Selection

All accuracies after Feature Selection can be seen in table 4.5, and the ranking of the top ten classifiers is shown in table 4.6. The maximum test accuracy achieved by a classifier after Feature Selection is better than that before feature selection, although for individual classifiers the accuracies may decrease slightly. Overall, the feature selection did not result in much better results, but different classifiers than before achieve the better accuracies.

The best accuracy after Feature Selection is achieved by JRip on the Parents & Hosts data set with the Risk Level mapping - the accuracy is 87.8261 %. However, the accuracies for the more informative data set with no genre mapping closely follow on rank 2 to 4: They all use the Parents & Hosts data set and, as mentioned, no genre mapping. First comes AODE with an accuracy of 87.3913 % on that set, then SMO with 86.5217 % and then JRip with 86.087 %. Since all of these accuracies are very close together, we can assume that on the data with a reduced feature set, all three algorithms show good results. While SMO already performed well before the feature selection, it shows a slightly decreased accuracy afterwards, 0.4348 percentage points less. AODE, on the other hand, gained 2.6087 percentage points in accuracy, and JRip gained 3.0435 percentage points. A good result of the feature selection is that now the data set using no genre mappings comes before the ones with a mapping in the ranking. This means that with the reduced feature set, the data set with no genre mapping, i.e. the one that gives us more detailed information about the context than the others, has an advantage over the ones which provide less information. Since more information is better for more specific security warnings, this is a good thing for us.

Compared to the performance before feature selection, there is only a slight increase in the accuracies achieved. However, the training and classification efficiency, as well as the computation time for extracting the features from a web page, can also be improved by reducing the number of features.

Table 4.5.: Accuracies of the classifiers on test and training set after Feature Selection, sorted by test accuracy and then training accuracy within each dataset. One dataset is specified by the pages it uses, the genres that were included when selecting instances and the genre mapping that was used. Datasets are separated by horizontal lines in this table. The lines showing the baseline value, i.e. the results of the classifier ZeroR, are colored in gray in order to visualize which classifiers score below or above the baseline within each dataset.

Pages	Genres	Mapping	Algorithm	Accuracy Training [%]	Accuracy Test [%]
Parents	All but Other	Risk Levels	SMO	84.7143	84.6154
Parents	All but Other	Risk Levels	AODE	86.7619	84.3407
Parents	All but Other	Risk Levels	JRip	84.5714	82.4176
Parents	All but Other	Risk Levels	ZeroR	74.9524	75.8242
Parents	All but Other	Risk Levels	NaiveBayes	50.4286	53.022
Parents	All but Other	Shopping Combined	SMO	81.8571	82.4176
Parents	All but Other	Shopping Combined	JRip	82.0476	81.8681
Parents	All but Other	Shopping Combined	AODE	82.0952	76.6484
Parents	All but Other	Shopping Combined	ZeroR	74.9524	75.8242
Parents	All but Other	Shopping Combined	NaiveBayes	28.8571	34.8901
Parents	All but Other		SMO	82.4762	82.4176
Parents	All but Other		JRip	82.5714	80.2198
Parents	All but Other		AODE	82.3333	79.9451
Parents	All but Other		ZeroR	74.9524	75.8242
Parents	All but Other		NaiveBayes	26.1905	34.8901
Parents & Hosts	All but Other	Risk Levels	JRip	84.8276	87.8261
Parents & Hosts	All but Other	Risk Levels	AODE	86.3602	85.2174
Parents & Hosts	All but Other	Risk Levels	SMO	85.977	85.2174
Parents & Hosts	All but Other	Risk Levels	ZeroR	82.6054	81.7391
Parents & Hosts	All but Other	Risk Levels	NaiveBayes	40.9195	46.087
Parents & Hosts	All but Other	Shopping Combined	SMO	85.9004	84.7826
Parents & Hosts	All but Other	Shopping Combined	JRip	85.1341	83.4783
Parents & Hosts	All but Other	Shopping Combined	AODE	87.5096	82.6087
Parents & Hosts	All but Other	Shopping Combined	ZeroR	82.6054	81.7391
Parents & Hosts	All but Other	Shopping Combined	NaiveBayes	36.2452	37.8261
Parents & Hosts	All but Other		AODE	87.7395	87.3913
Parents & Hosts	All but Other		SMO	87.1264	86.5217
Parents & Hosts	All but Other		JRip	84.6743	86.087
Parents & Hosts	All but Other		ZeroR	82.6054	81.7391
Parents & Hosts	All but Other		NaiveBayes	32.1839	33.4783

Table 4.6.: Top ten classifiers after Feature Selection, ranked by test accuracy and then training accuracy. A diagram visualizing this table can be seen in figure 4.1c.

Rank	Pages	Genres	Mapping	Algorithm	Accuracy Training [%]	Accuracy Test [%]
1	Parents & Hosts	All but Other	Risk Levels	JRip	84.8276	87.8261
2	Parents & Hosts	All but Other		AODE	87.7395	87.3913
3	Parents & Hosts	All but Other		SMO	87.1264	86.5217
4	Parents & Hosts	All but Other		JRip	84.6743	86.087
5	Parents & Hosts	All but Other	Risk Levels	AODE	86.3602	85.2174
6	Parents & Hosts	All but Other	Risk Levels	SMO	85.977	85.2174
7	Parents & Hosts	All but Other	Shopping Combined	SMO	85.9004	84.7826
8	Parents	All but Other	Risk Levels	SMO	84.7143	84.6154
9	Parents	All but Other	Risk Levels	AODE	86.7619	84.3407
10	Parents & Hosts	All but Other	Shopping Combined	JRip	85.1341	83.4783

Table 4.7.: Confusion matrix for **SMO** on the **Parents & Hosts** data set with **all genres but Other** and **no genre mapping** Entries are percentage values, i.e. the number of instances in a cell is divided by the number of all instances that belong in that genre.

Classified incorrectly as ... [% of genre instances]							Genre	Classified correctly
banking	data	info	mail	shopping1	shopping2	social		[% of genre instances]
	0	0	0	0	0	0	banking	100
0		100	0	0	0	0	data	0
0	1		0	3	1	1	info	94
0	0	50		0	0	0	mail	50
0	0	20	0		0	0	shopping1	80
0	0	50	0	0		0	shopping2	50
0	0	46	8	8	0		social	38

4.1.4 Conclusions So Far

In order to compare all the classifiers, we also made one ranking where all of them are sorted according to their test accuracy. The top ten of all classifiers we trained, both on standard and uniform data sets and with and without feature selection, can be seen in table 4.20. In figure 4.1, a bar chart for this ranking as well as the three others is presented in order to visualize the different accuracies.

With the regard to the choice of data set, we can see that data sets with Parents & Hosts perform better than those with Parents only. The data sets using all genres perform worse than those using all genres but Other, which confirms that Other is not a genre on its own but rather a mixture of things that do not fit elsewhere or that were mistakenly labelled as Other. The genre mapping does not seem to make a big difference, since variants with each genre mapping appear among the top ten in each ranking. Therefore, we should use the Parents & Hosts version of the data set, use all genres but Other and choose the mapping that best suits our needs. The data set with no genre mapping provides the most information about the web page context, so it is best if we take that one.

Therefore, we will now concentrate on finding an algorithm that yields the best results on the data set with Parents & Hosts, all genres but Other and no genre mapping. The algorithm can use either the data set with all features or the one after feature selection, although the data set after feature selection is preferable because less features means reduced computation time. We do not consider ZeroR of course because it assigned all instances to Information Site and just served as a baseline this way. From the accuracies we observed, we can also discard Naive Bayes as a possibility because it did not show good results on that data set. Hence SMO, AODE and JRip remain for further analysis. All showed good results on the data set. Some performed better before, some after feature selection. If we look at all the rankings, SMO appears most often among the top ten, then AODE and then JRip. However, their accuracy results differ only a little, so we do not know yet if any of them performs significantly better or worse than the others. This is what will find out when we perform a paired t-test for these selected algorithms on the selected data sets.

We will now concentrate on further analysing the performance of AODE, SMO and JRip on the Parents & Hosts data set with all genres but Other and no genre mapping. We will look at the confusion matrices for the standard and the uniform version of the data set and also the version after feature selection. We then analyse the misclassified web pages and confidence scores for predictions for the best variant of each of the three algorithms, i.e. the one that had the highest accuracy on the Parents & Hosts data set with all genres but Other and no genre mapping.

4.1.5 Predictions of Selected Classifiers

In this section, we have a closer look at the examples where the selected classifiers misclassified an instance, i.e. assigned it to a wrong genre. This will help us to get a better understanding of where the problems lie and if something can be done against them. First, we look at the confusion matrices created by the different methods, and then we look at some sample web pages that were misclassified.

Confusion Matrices

We choose to have a closer look at the confusion matrices of SMO, AODE and JRip on the Parents & Hosts data sets with no genre mapping. We look at the version of the data set with standard genre distribution, with uniform genre distribution, and after feature selection.

We use a variant of the typical confusion matrix here. Instead of listing the absolute number of misclassified instances in each cell of the matrix, we list the percentage of instances in that genre that were classified correctly or misclassified as

Table 4.8.: Confusion matrix for **SMO** on the **uniform Parents & Hosts** data set with **all genres but Other** and **no genre mapping**. Entries are percentage values, i.e. the number of instances in a cell is divided by the number of all instances that belong in that genre.

Classified incorrectly as ... [% of genre instances]							Genre	Classified correctly
banking	data	info	mail	shopping1	shopping2	social		[% of genre instances]
	0	0	0	0	0	0	banking	100
0		25	0	25	0	0	data	50
0	6		0	13	3	9	info	69
0	0	39		0	0	0	mail	61
0	0	0	0		0	0	shopping1	100
0	0	0	0	0		16	shopping2	84
0	12	30	3	6	0		social	48

Table 4.9.: Confusion matrix for **SMO** on the **Parents & Hosts** data set with **all genres but Other** and **no genre mapping** after Feature Selection. Entries are percentage values, i.e. the number of instances in a cell is divided by the number of all instances that belong in that genre.

Classified incorrectly as ... [% of genre instances]							Genre	Classified correctly
banking	data	info	mail	shopping1	shopping2	social		[% of genre instances]
	0	0	0	0	0	0	banking	100
0		100	0	0	0	0	data	0
0	0		0	1	0	0	info	99
0	0	75		0	0	0	mail	25
0	0	40	0		0	0	shopping1	60
0	0	100	0	0		0	shopping2	0
0	0	77	8	0	0		social	15

another genre. We do this because the genre distribution is so skewed that all values for Information Site would always be much higher than those for the rest just because there are much more instances that belong to Information Site. So in order to allow us to compare the misclassifications for each genre, we choose to work with percentages instead. To see how this looks in practice, consider the following example: The genre Online Banking contains 10 instances in a data set, of which 3 were classified as Information Site and 7 are classified as Online Banking. Therefore, 30 % of the instances, i.e. 3/10*100 %, are misclassified. The remaining 70 %, i.e. 7/10*100 %, are correctly classified. So the numbers in the corresponding cells of the confusion matrix will be 30 % and 70 % respectively, and not 3 and 7.

Based on the percentage values, we color-coded the cells of the confusion matrix to visualize which genres were most often confused with each other. Green encodes a good value, yellow an average one, and red a bad one, with some shades of each color in between. For the misclassifications, a low value is better, i.e. not many misclassified instances. For the correct classifications, a high value is better, i.e. many correctly classified instances. Table 4.15 shows all the colors which were used for the visualization, and to which percentages from either misclassifications or correct classifications they apply. Note that a lot of red cells means that a lot of misclassifications occured, which is bad. However, a lot of green cells does not necessarily imply that everything is fine - it can also mean that the misclassifications occured in different places, so for each cell, there are not so many misclassifications as to render it red, but in total, the number of misclassifications may still be high. This occurs for the confusion matrices of SMO on the standard and on the uniform data set, which can be seen in tables 4.7 and 4.8, respectively. While the standard data set version of the classifier has the better accuracy, misclassifications are more concentrated there, so its confusion matrix looks much worse than the one for the uniform data set version, although in reality, it is the other way around.

First, we will look at the confusion matrices for SMO on the Parents & Hosts data set with all genres but Other and no genre mapping. Table 4.7 shows the matrix for classifier trained on the standard version of that data set while table 4.8 is the one for uniform genre distribution, and table 4.9 are the values after feature selection.

First, we look at SMO's results for the standard data set as shown in table 4.7. We see that most misclassifications of SMO are instances that are classified as Information Site but belong to another genre. All the instances from Data Exchange, about half the instances from E-Mail, Shopping 2 and Social Networks, and one fifth of the instances from Shopping 1 were incorrectly classified as Information Site. Apart from instances being misclassified as Information Site, there are only a small number of instances misclassified as something else. Some Social Network instances have been misclassified as either E-Mail (8 % of all Social Network instances) or Shopping 1 (8 %). Information Sites have been misclassified as Data Exchange (1 %), Shopping 1 (3 %), Shopping 2 (1 %) and Social Networks (1 %), but all of these percentages are rather small.

Table 4.10.: Confusion matrix for **AODE** on the **Parents & Hosts** dataset with **all genres but Other** and **no genre mapping**. Entries are percentage values, i.e. the number of instances in a cell is divided by the number of all instances that belong in that genre.

Classified incorrectly as ... [% of genre instances]							Genre	Classified correctly [% of genre instances]
banking	data	info	mail	shopping1	shopping2	social		
	0	100	0	0	0	0	banking	0
0		100	0	0	0	0	data	0
0	0		0	4	1	2	info	94
0	0	75		0	0	0	mail	25
0	0	33	0		0	0	shopping1	67
0	0	33	0	33		0	shopping2	33
0	0	46	8	0	0		social	46

The only genre for which all instances were classified correctly is Online Banking, which is good because this is one of the genres corresponding to the highest risk level. For Information Site, almost all instances were classified correctly as well, and Shopping 1 has 80 % of correctly classified instances. For all other genres, at most half the instances were classified correctly. For Data Exchange, none of the instances was classified correctly.

It is possible that the misclassifications are a result of the genre Information Site having much more instances than any other genre: If one genre occurs much more frequently than all the others, it is a good idea for the classifier to decide on that genre when no genre seems to fit, because the probability that this guess is correct is the highest for that genre.

For this reason, we will look next at the confusion matrix of SMO on the data set with uniform genre distribution, which is shown in table4.8. As we can see, the misclassifications appear more distributed in those data sets. Still, a tendency to misclassifiy instances as Information Site is visible there as well: 25 % of the instances in Data Exchange, 39 % of the instances in E-Mail and 30 % of the instances in Social Networks have been misclassified as Information Site. Another 25 % of the Data Exchange instances have been misclassified as Shopping 1, but all other percentages of misclassification are lower than those in the Information Site column. Curiously, 12 % of instances from Social Networks now get misclassified as Data Exchange and 16 % of Shopping 2 instances get misclassified as Social Networks, although in neither of those two cases there were any misclassified instances on the standard data set. The percentage of Social Network instances that were misclassified as E-Mail or Shopping 1 decreased, compared to the standard data set. On the other hand, the percentage of Information Site instances misclassified as Data Exchange, Shopping 1, Shopping 2 or Social Network increased.

Overall, we see that the tendency to misclassify instances as Information Site is still there although less strongly so. Additionally, using the data set with uniform genre distribution also introduced misclassifications were before there were none or at least less.

This suggests that the many misclassifications in Information Site are only partly due to the fact that Information Site is the most frequent genre. There seem to be other problems responsible as well, maybe the mislabelled web pages in the Wekasa collection. Also, the newly introduced errors on the uniform data set where there were none on the standard data set explain why the accuracy on the uniform data set is, in fact, worse than on the standard data set.

SMO was one of the classifiers for which feature selection did not improve the accuracy, but it did not decrease it much either. Since feature selection is good for our application as the computation time will decrease when the number of features is smaller, we will look at the confusion matrix of SMO after feature selection as well.

We see that after the feature selection, the misclassifications are even more concentrated in the Information Site column, i.e. there are even more instances misclassified as Information Site than on the standard data set without feature selection. In exchange, the numbers for other misclassifications get smaller or even become zero. The only misclassifications that occur now apart from that are Social Networks misclassified as E-Mail (8 %), and Information Sites misclassified as Shopping 1 (1 %).

For AODE, only the confusion matrices for the standard data set and the one after feature selection exist because for the uniform data set, no classifier could be computed due to AODE's rather high memory requirements. For the two data sets where the AODE classifier could be computed, the confusion matrices are given in table 4.10 for the standard data set and in table 4.11 for the data set after feature selection.

If we look at AODE's confusion matrix for the standard data set as shown in table 4.10, we can see that it has a similar problem as SMO: Most of the misclassifications are instances that are classified as Information Site but belong to another genre.

Finally, we look at the confusion matrices for JRip on the Parents & Hosts data sets with no genre mapping, first the one using the standard set, which is shown in table 4.12. Like the other two algorithms, JRip also shows a strong tendency to misclassify instances as Information Site. There is one misclassification with a notable percentage that is not in the Information Site column, which is 33 % of Data Exchange Instances that are misclassified as Social Networks. Apart

Table 4.11.: Confusion matrix for **AODE** on the **Parents & Hosts** dataset with **all genres but Other** and **no genre mapping** after **Feature Selection**. Entries are percentage values, i.e. the number of instances in a cell is divided by the number of all instances that belong in that genre.

Classified incorrectly as ... [% of genre instances]							Genre	Classified correctly
banking	data	info	mail	shopping1	shopping2	social		[% of genre instances]
	0	100	0	0	0	0	banking	0
0		100	0	0	0	0	data	0
0	0		0	2	0	0	info	98
0	0	75		0	0	0	mail	25
0	0	40	0		0	0	shopping1	60
0	0	83	0	0		0	shopping2	17
0	0	54	8	0	0		social	38

Table 4.12.: Confusion matrix for **JRip** on the **Parents & Hosts** dataset with **all genres but Other** and **no genre mapping**. Entries are percentage values, i.e. the number of instances in a cell is divided by the number of all instances that belong in that genre.

Classified incorrectly as ... [% of genre instances]							Genre	Classified correctly
banking	data	info	mail	shopping1	shopping2	social		[% of genre instances]
	0	0	0	0	0	0	banking	100
0		67	0	0	0	33	data	0
0	1		1	3	1	2	info	93
0	0	50		0	0	0	mail	50
0	7	33	0		0	0	shopping1	60
0	0	83	0	0		0	shopping2	17
0	0	62	8	0	0		social	31

Table 4.13.: Confusion matrix for **JRip** on the **Parents & Hosts** dataset with **all genres but Other**, **uniform** genre distribution and **no genre mapping**. Entries are percentage values, i.e. the number of instances in a cell is divided by the number of all instances that belong in that genre.

Classified incorrectly as ... [% of genre instances]							Genre	Classified correctly
banking	data	info	mail	shopping1	shopping2	social		[% of genre instances]
	0	0	0	0	0	0	banking	100
0		50	0	0	0	0	data	50
0	22		9	3	3	6	info	56
0	0	0		17	0	0	mail	83
0	0	23	0		0	0	shopping1	77
0	0	59	0	0		0	shopping2	41
0	15	30	15	0	0		social	39

Table 4.14.: Confusion matrix for **JRip** on the **Parents & Hosts** dataset with **all genres but Other** and **no genre mapping** after **Feature Selection**. Entries are percentage values, i.e. the number of instances in a cell is divided by the number of all instances that belong in that genre.

Classified incorrectly as ... [% of genre instances]							Genre	Classified correctly
banking	data	info	mail	shopping1	shopping2	social		[% of genre instances]
	0	0	0	0	0	0	banking	100
0		67	0	0	0	33	data	0
0	1		0	2	0	2	info	95
0	0	50		0	0	0	mail	50
0	0	33	0		0	0	shopping1	67
0	0	100	0	0		0	shopping2	0
0	0	46	8	0	0		social	46

Table 4.15.: Colors of the confusion matrix visualization - Each cell here has the color the corresponding confusion matrix cell has if its value is in the specified interval. Color codes are different for correctly and incorrectly classified instances because for the former, a high value is good and for the latter, a low value is good, so the color codes are adapted accordingly.

Correctly classified instances	84-100 %	67-83 %	50-66 %	33-49 %	16-32 %	0-15 %
Incorrectly classified instances	0-16 %	17-33 %	34-50 %	51-67 %	68-84 %	85-100 %

from that, there is a small percentage of instances from Information Site that is misclassified as one of the other genres except Online Banking. A similar behaviour was shown by both SMO and AODE, although SMO misclassified Information Sites only as all genres but Online Banking and E-Mail, and AODE as all genres but Online Banking, E-Mail and Data Exchange. Apart from misclassifications that have to do with Information Sites, two more can be observed: Some Shopping 1 instances (7 %) were misclassified as Data Exchange, and some Social Networks (8 %) were misclassified as E-Mail. In general, JRip classified and equal or lower number of instances correctly than SMO.

If we look at JRip's confusion matrix for the data set with uniform genre distribution as shown in table 4.13, we see that the tendency to misclassify instances as Information Site is still present, although less strongly. The same thing was observed for SMO as well. Compared to JRip's confusion matrix for the standard data set, the percentage of instances that belong to Information Site but are classified as something else has increased. Data Exchange instances are no longer misclassified as Social Networks but now E-Mail instances are misclassified as Shopping 1 (17 %) and Social Networks are misclassified as Data Exchange (15 %), which did not happen on the standard data set. Just like for SMO, we see that on the uniform data set, the problem that many instances are misclassified as Information Site is reduced but in exchange for that, other problems occur.

For JRip, feature selection improved the accuracy on the test set. If we compare the confusion matrix before feature selection (table 4.12) and after (table 4.14), we see that the percentage of correctly classified instances increased for the genres Information Site, Shopping 1 and Social Networks. It decreased for Shopping 2, and for Online Banking, Data Exchange and E-Mail, it stayed the same. We see that most misclassifications are instances misclassified as Information Site, just like before the feature selection. The percentage of instances misclassified as Information Site has decreased for the instances from Social Networks but increased for those from Shopping 2, while they stay the same for the rest of the genres. There are still some instances from Information Site that are misclassified as another genre, but now only as Data Exchange, Shopping 1 or Social Network. The percentage of Data Exchange instances misclassified as Social Networks and Social Networks misclassified as E-Mail stays the same than before the Feature Selection.

In conclusion, we see that all three algorithms have a tendency to misclassify instances as Information Site. This tendency is less strong but still visible on the data sets with uniform genre distribution, so we can conclude that the reason why so many instances are misclassified as Information Site cannot be just the skewed genre distribution alone. Additionally, using uniform data sets decreases the overall accuracy and introduces new misclassifications that were not present before, so they do not present a solution to the problem that many instances are misclassified as Information Site. We can also observe a tendency to misclassifiy instances belonging to Information Site as something else, which is not so obvious as the misclassfication the other way around because the percentages are rather low. However, since Information Site is such a large genre, this may well mean that this concerns a notable number of instances anyway and therefore should be considered. Apart from misclassfications concerning Information Sites one way or the other, the percentage of misclassifications is low or zero. This means that if we can solve the problem of Information Site misclassifications, we can greatly improve the classifier as for the other cases, it already works well.

Some classification errors may be compensated by the InUse scoring tool, so not all of them are equally bad. Web pages where the user wants to log in that are misclassified as an Information Site are not so bad, because they can still be recognized as such when the user tries to log in. Some errors may also occur because there are, for example, a lot of web shops in Information Site in the WeKaSa data set, as discussed in 2.2.2. Classifying such an instance as Shopping1 or Shopping2 would not be problem, since that actually fits better than Information Site. In order to find out whether such cases occur, we have to look at the misclassifications.

Misclassified Web Pages

In the following, we will look closer at those web pages that the selected classifiers misclassified. We look at the misclassifications of SMO, AODE and JRip on the Parents & Hosts data set with all genres but Other. Depending on which worked best for each algorithm, we choose either the variant with or without feature selection. For SMO, that is the one without feature selection and for JRip and AODE, it is the one with feature selection.

What we want to find out by looking at the misclassifications is if they have something in common that may give us a hint as to where the problems lie. Also, we want to see if the misclassification is really wrong or if the predicted genre

Table 4.16.: Categorization of misclassifications

SMO	JRip	AODE	Description
3	1	1	Not relevant
4	6	3	Predicted fits better
0	0	0	Both fit
7	7	9	Actual fits better, but predicted is not completely wrong
14	16	13	Actual fits better
2	2	3	Neither fits

actually fits better than the one assigned by a human being - it is possible, for example, that the classifier assigned the genre Shopping 1 or Shopping 2 to one of the web shops that have been labelled as Information Site. And we want to see if the three classifiers misclassify the same web pages or different ones. If they classifiy different web pages correctly, we might be able to achieve better accuracy if we combine several classifiers.

In total, we have 48 web pages that were misclassified by at least one of the classifiers. Out of those, AODE has misclassified 29 web pages, SMO has misclassified 30 and JRip has misclassified 32. There are 19 web pages all three classifiers misclassified. 5 were misclassified by two classifiers, and the remaining 24 are misclassified by one classifier only.

We can use this information to combine the classifiers and take the genre label that the majority of classifiers agree on. This means that in the 24 cases where just one classifier misclassified the web page and the other two agree on the correct genre, we can use that genre because the majority selected it. In that case, this means we get the correct genre label. However, there are also cases where just one classifier selects the correct genre and the other two agree on a different genre, so the majority of classifiers votes for the wrong genre. Here, this case is true for four web pages. Also, we have one case where all three classifiers select different genres, where we would have to either choose a genre label according to some other criterion or assign a default class. In total, we get 24 more correct labels but have five cases where either a wrong label or a default genre needs to be selected. Therefore, we are indeed able to classify more web pages correctly if we combine the three classifiers and choose the genre label that the majority of them votes for.

In table 4.16, we can see the results of categorising each misclassified web page. For some of the web pages, the misclassification is irrelevant, for example if the web page was just an empty page, because due to the lack of information, we cannot expect the classifier to sort empty pages correctly into the different genres. There are only a few cases were the web page falls into that category. Sometimes, the genre label predicted by the classifier actually fits better than the one assigned by a human being. Some web pages in Information Site for example are actually web shops, so the classifier was right in assigning them to the genre Shopping 1. This happened for some of the web pages but not many, so we cannot see a tendency suggesting that our classifiers are able to correct the wrong genre labels in our data set. There were no cases where both the actual genre and the predicted genre fit equally well. In some cases, the actual genre label fits better then the predicted one but the predicted one is not completely wrong either. In the majority of cases, however, the actual genre fits better than the predicted one. There are some more cases where neither one of the genre labels fits. Most of these were examples where a discussion board was labelled as E-Mail by users and as Information Site by the classifiers. In that case, the most fitting label would actually be Social Networks, as a discussion board can be seen as a simple form of a social network; however, the users who did label discussion boards as E-Mail probably have reasons for doing so as well, so this essentially just shows that finding a genre label is hard and that even humans do not always agree on the correct label.

Confidence

Each classifier in Weka can output a class distribution for each instance, i.e. a list of probabilities for each class, which we can use as a confidence score. We can analyse these to see if the confidence gives us any indication of whether the classification is going to be correct or not. Ideally, we find a confidence threshold below which the classification is usually wrong and above which it is usually correct. We can then use this information in order to assign all web pages with a confidence above the threshold to the genre the classifier predicted, and all web pages with a confidence below the threshold get assigned a genre *unknown*. Apart from the threshold, we can use the confidence values to not just output a class label but to output a vector of possible class labels, sorted in descending order according to how likely the classifier thinks they are for this instance.

If we look at the confidence scores, we see that there is no obvious threshold. No matter which value we choose, we will always include some of the incorrect classifications in those we believe to have a sufficient confidence, and we will always exclude some of the correct classifications and assign the genre *unknown* to them. However, for each of the classifiers, the incorrectly assigned instances have a slightly lower confidence on average than the correctly assigned ones.

Table 4.17.: Confidence values for SMO. The table shows how many instances fall into each class, i.e. which had a confidence below or equal to the given maximum value. The total number of instances in each frequency class is shown, as well as the number of correctly and incorrectly classified instances.

Class Maximum	Instances	Instances Correct	Instances Incorrect
0.1	0	0	0
0.2	0	0	0
0.3	230	200	30
0.4	0	0	0
0.5	0	0	0
0.6	0	0	0
0.7	0	0	0
0.8	0	0	0
0.9	0	0	0
1	0	0	0

Table 4.18.: Confidence values for AODE. The table shows how many instances fall into each class, i.e. which had a confidence below or equal to the given maximum value. The total number of instances in each frequency class is shown, as well as the number of correctly and incorrectly classified instances.

Class Maximum	Instances	Instances Correct	Instances Incorrect
0.1	0	0	0
0.2	0	0	0
0.3	0	0	0
0.4	0	0	0
0.5	1	0	1
0.6	5	4	1
0.7	8	5	3
0.8	8	6	2
0.9	5	5	0
1	203	181	22

Therefore, we may be able to find a threshold such that the number of correct classifications with a confidence below the threshold and the number of incorrect classifications with a confidence above the threshold is minimized. Finding a useful threshold for the classifier that is going to be used in the InUse scoring tool is left for future work, as the classifier has not been choosen yet.

The only case where it is easy to choose the threshold is for SMO, because only two confidence values occur: Correct classifications have a confidence score of 0.286 and incorrect classfications have a confidence score of either 0.286 or 0.238. Therefore, we can define a threshold so that 0.286 means we use the assigned genre label and 0.238 means we use the genre label *unknown*. This way, we can assign the label *unknown* to 4 out of the 30 misclassifications and none of the correct classifications is labelled as *unknown*.

We can use the confidence scores to get a vector of genre labels instead of just one. We only need to account for cases where two genres have the same confidence score, because in a vector, no two genres can occupy the same slot. We could rank them according to the number of instances of that genre in the data set, because we can assume that the more frequent genre label is more likely if in doubt. We can use this genre vector, for example, if the most probable label is Information Site but the user attempts to log in to that web page, so we know the user's intention is not to search for information at that point. We could then choose the second most likely label as the web page's genre and show a corresponding warning if necessary.

We can also use the confidence scores to combine classifiers. We can use one classifier to classify an instance first, and if its confidence for the genre it selected is below a certain threshold, we use another classifier and let it assign the genre label. We can also do this with more than two classifiers, of course. And we can let all three select a genre for a web page and then use the result with the highest confidence score. In this case, however, we have to be careful because the confidence values of the different classifiers are rather different from each other and fall into different ranges. Even if a classifier has a higher accuracy than another, it can still have lower confidence scores than another, which means that the genre assignment of the classifier with lower accuracy would be preferred over the one with higher accuracy, which would not result in a good combination of classifiers.

Table 4.19.: Confidence values for JRip. The table shows how many instances fall into each class, i.e. which had a confidence below or equal to the given maximum value. The total number of instances in each frequency class is shown, as well as the number of correctly and incorrectly classified instances.

Class Maximum	Instances	Instances Correct	Instances Incorrect
0.1	0	0	0
0.2	0	0	0
0.3	0	0	0
0.4	0	0	0
0.5	0	0	0
0.6	1	1	0
0.7	13	9	4
0.8	10	6	4
0.9	6	3	3
1	200	179	21

4.1.6 Paired T-Test for Selected Classifiers

We compare the best classifiers from the previous section with the Weka Experimenter in order to find out if differences in the classifiers results are statistically significant. What we want to find out is if one of them is actually better than the others, or if their results are so close together that it is not possible to recommend one over another.

The Weka Experimenter allows to set up an experiment where several classifiers are trained on one or more data sets and then compared with a paired T-Test. The T-Test essentially estimates the results of the classifiers on all unseen data by performing several runs on the available data and averaging their results. Since you do not usually specify a training and a test set explicitly, we use ARFF files with both the training and the test examples in one data set, so both the instances that influenced the development of the features etc. and those that did not are part of the experiment.

As discussed in 4.1.4, we will focus on finding the best classifier for the data set with Parents & Hosts, all genres but Other and no genre mapping. We use both the variant with and without feature selection because either one is suitable for our application. Since we do not want to evaluate the average performance of a learning algorithm over various data sets but rather find one algorithm that works best for each data set, we set up two experiments: One where all algorithms are compared on the data set without feature selection and one where all are compared on the data set with feature selection. The algorithms we compare are SMO, AODE and JRip.

Our results are generated by a CrossValidationResultProducer that uses ten folds, as before in our cross validation on the training set. They are sent to an InstancesResultListener so we can analyse them further, i.e. perform the T-Test. We perform a corrected resampled T-test between either classifier and the others. We compare the classifiers' results, i.e. the number of instances they classified correctly, both at the 0.05 level and at the 0.1 level.

The first experiment compares the three algorithms on the data set without Feature Selection. At the 0.05 level, SMO performs significantly better than the other two. AODE and JRip however do not show a significant difference in accuracy, neither at the 0.05 nor at the 0.1 level.

The second experiment compares the three algorithms on the data set with Feature Selection. At the 0.05 level, both SMO and AODE perform significantly better than JRip. Between SMO and AODE, no significant difference in accuracy can be detected. At the 0.1 level, the same happens: SMO and AODE are better than JRip, but no significant difference between SMO and AODE is observed.

As SMO performs either significantly better than or equally well as the other two on both data sets, we can conclude that SMO is our best choice. Since its accuracy was higher on the data set before feature selection, it is better to use that data set. However, since feature selection also means improved computation time, it is recommended to try other feature selection methods that may be more beneficial to SMO.

4.1.7 The Best Classifiers

If we take all the classifiers we trained, on each data set, with each algorithm, with and without Feature Selection and Standard or Uniform, we get the top ten classifiers from table 4.20.

The best classifiers for our preferred data set, i.e. the one using no genre mapping, are the following ones:

- AODE on the Parents & Hosts data set after Feature Selection, with a test accuracy of 87.3913
- SMO on the Parents & Hosts data set before Feature Selection, with a test accuracy of 86.9565

Table 4.20.: Top ten classifiers of all 120 classifiers we trained. These are all classifiers on Parents & Hosts data sets with all genres but Other and standard genre distribution in the set (as opposed to uniform distribution), so the corresponding columns are left out in the table.

Rank	Mapping	Algorithm	Feature Selection	Accuracy Training [%]	Accuracy Test [%]
1	Risk Levels	JRip	Chi Square	84.8276	87.8261
2	None	AODE	Chi Square	87.7395	87.3913
3	None	SMO	All	87.7395	86.9565
4	Risk Levels	AODE	All	87.433	86.9565
5	None	SMO	Chi Square	87.1264	86.5217
6	Shopping Combined	SMO	All	88.1226	86.087
7	None	JRip	Chi Square	84.6743	86.087
8	Shopping Combined	AODE	All	88.1226	85.6522
9	Risk Levels	AODE	Chi Square	86.3602	85.2174
10	Risk Levels	SMO	Chi Square	85.977	85.2174

- SMO on the Parents & Hosts data set after Feature Selection, with a test accuracy of 86.5217
- JRip on the Parents & Hosts data set after Feature Selection, with a test accuracy of 86.087

The difference between these accuracies is small, so we can conclude that it is likely that either one works well. Since SMO is the algorithm most commonly used in text classification and the one appearing most often in either one of our rankings, we would recommend to use it for the InUse scoring tool. Using AODE is more complicated since it requires the numeric attributes to be discretized first, and because it has some efficiency problems. Hence SMO is likely to produce a faster classifier than AODE, which is important for our application. Implementing JRip is less complicated than implementing SMO in the InUse scoring tool, but it showed a less consistent performance than SMO and AODE. While SMO and AODE usually both landed in the top ten in all rankings with the same data sets, JRip's performance differed a lot more. Since our classification problem may change in the future as new features or new training examples are added, SMO is a better choice because its performance was less susceptible to changes in the data set.

Before using SMO in the InUse scoring tool, some time should be spend on trying to further improve its results. SMO has some parameters and different kernels that can be changed, so different settings of these should be evaluated against each other. Finding the best combination of them can lead to a better accuracy. Furthermore, the Feature Selection method we used here did not improve SMO's results but instead lead to a slight decrease in accuracy. It is however possible that a different Feature Selection method select features that work better for SMO and thus improve both its performance and the computation time for a feature vector, which is important when using the classifier in practice. Therefore, some more research should be done on a good selection of features for an SMO classifier.

Before improving any individual classifier's performance, however, it is recommend to try these experiments again on a collection of web pages where the genre labels have been corrected if they were wrong. In the collection we use here, many instances are labelled as Information Site although they do not appear to be one. This may be a reason why all the classifiers have the problem that they assign too many web pages to the genre Information Site although they belong elsewhere. Therefore, the first thing that should be done is to go through the WeKaSa data set and see if the genre labels are correct and if not, to correct them. One can then run the experiments that were done here again on the new data set. If the findings from this thesis still remain valid on that web page collection, one should try to improve the classifier that appears best suited to the task, which in this case is SMO. If one the corrected web page collection the result but improve overall, one should of course used the best classifier that can be trained on the corrected web page collection.

4.2 Features

This section describes our results with regard to the features, i.e. which features worked best to distinguish the genres. This was determined by applying a Feature Selection method to some of the data sets.

Feature Selection was performed by computing the Chi Square metric for each feature in the data set, ranking the features according to it and using the top 10 % of the features. This was done for the data sets that proved to be most useful when training the classifier even without feature selection. These were both the Parents and the Parents & Hosts version of the data sets using all genres but Other and either no genre mapping, the Shopping Combined mapping or the Risk Level mapping.

The selected features for the parents data set can be seen in table 4.21, and those for the Parent & Host data sets in table 4.22. We can see that on the Parents version of the data set using no mapping, especially the URL keyword counts

and the topic features were selected. On the Parents & Hosts version, almost all selected attributes are host attributes, 48 out of the selected 54 attributes to be exact.

In the Parents & Hosts data sets, most of the selected features are host features. The total number of features selected for the host data sets is 54. Out of those 54, 48 are host attributes for the data set with no mapping, 51 for the data set with Shopping Combined mapping and 46 for the data set with Risk Level mappings. This further confirms that using the host information is beneficial to our results, which the accuracies on different data sets as discussed in 4.1.1 suggest as well.

Apart from host attributes, topic features seem to work well: In the Parents & Hosts data sets, there were 11 topic features selected for the data set with no mapping, 10 for the data set with Shopping Combined Mapping, and 25 for the one with Risk Level mapping. For the Parent version of the no mapping and shopping combined data sets, topic features seem to be even more important: 17 out of 27 features selected for the data set with no mapping are topic features, and 19 out of 27 for the data set with shopping combined mapping. For the Parents data set with risk level mapping, however, the number of topic features selected is smaller than in the Parents & Hosts version of the data set: Only 7 topic features were selected. Since the many topic features selected for the Parents & Hosts data set with risk level mapping were mostly host topic features, this makes sense. For the other two Parent data sets, it seems that when the host information is not available anymore, the topic features play the most important role in identifying the web page's genre.

Also, URL keywords appear to be successful in distinguishing genres: There are 16 URL keyword features for the Parents & Hosts data set and 8 for the Parents data set in total. For the Parents & Hosts data sets, 5 of these were selected for the data set with no genre mapping, 3 for the shopping combined and 6 for the risk level mapping. For the Parents data sets, 5 were selected for the data set with no genre mapping, 4 for the shopping combined mapping and 6 for the risk level mapping.

Table 4.21.: Features selected for the Parents data sets

Feature	Shopping Combined	No Mapping	Risk Levels
a_tag_count			x
a_tag_frequency			x
banking_url_keyword_count	x	x	x
data_url_keyword_count	x	x	x
info_keyword_count			x
info_url_keyword_count			x
input_tag_count	x	x	
lexical_diversity			x
link_count			x
mail_keyword_count			
mail_url_keyword_count	x	x	x
password_field_count			x
percent_sign_frequency			
pos_cd_count			x
pos_nn_count	x	x	x
pos_nnp_count			x
pos_prpz_frequency			x
pos_vbn_frequency	x	x	x
punctuation_count			x
shopping1_keyword_count			x
shopping1_url_keyword_count	x	x	x
social_url_keyword_count		x	x
token_count		x	
topic_03	x		
topic_04	x	x	x
topic_06	x	x	x
topic_07	x	x	
topic_08	x	x	
topic_10			x
topic_16	x	x	
topic_17	x	x	x

Continued on next page

Feature	Shopping Combined	No Mapping	Risk Levels
topic_19	x	x	
topic_23		x	
topic_24	x	x	x
topic_25	x	x	
topic_27	x	x	
topic_32			x
topic_33	x	x	
topic_36	x		
topic_37	x	x	
topic_38	x		
topic_40	x	x	
topic_41	x	x	x
topic_44	x	x	
topic_47	x	x	
type_count	x	x	
type_token_ratio			x

Table 4.22.: Features selected for the Parents & Hosts data sets

Feature	Shopping Combined	No Mapping	Risk Levels
host_a_tag_count	x	x	x
host_a_tag_frequency	x	x	
host_banking_url_keyword_count	x	x	
host_br_tag_frequency	x	x	x
host_button_tag_frequency			x
host_colon_frequency	x		
host_comma_frequency		x	
host_div_tag_count	x	x	x
host_div_tag_frequency	x		x
host_double_quotation_mark_frequency	x	x	
host_exclamation_mark_count	x		
host_form_tag_frequency	x	x	
host_h1_tag_frequency	x		
host_html_tag_count	x	x	x
host_img_tag_count	x	x	
host_img_tag_frequency		x	x
host_input_tag_count	x	x	
host_input_tag_frequency			x
host_left_curly_bracket_count	x	x	
host_left_curly_bracket_frequency	x	x	
host_link_count	x	x	x
host_link_tag_count	x	x	
host_link_tag_frequency	x	x	
host_mail_keyword_count	x	x	x
host_mail_url_keyword_count	x	x	
host_password_field_count	x	x	x
host_percent_sign_frequency	x	x	x
host_pos_cd_frequency	x		
host_pos_colon_count			x
host_pos_colon_frequency	x	x	x
host_pos_dt_frequency			x
host_pos_period_count	x	x	
host_pos_period_frequency	x		

Continued on next page

Feature	Shopping Combined	No Mapping	Risk Levels
host_pos_vb_frequency	x	x	x
host_p_tag_count	x	x	x
host_p_tag_frequency		x	
host_punctuation_count	x	x	
host_right_curly_bracket_count	x	x	
host_right_curly_bracket_frequency	x	x	
host_shopping1_url_keyword_count	x	x	x
host_social_url_keyword_count		x	x
host_table_tag_frequency	x	x	
host_tbody_tag_frequency	x	x	
host_td_tag_count	x	x	x
host_td_tag_frequency	x	x	
host_th_tag_count	x		
host_topic_02			x
host_topic_03			x
host_topic_04	x	x	x
host_topic_11			x
host_topic_12			x
host_topic_13	x	x	x
host_topic_14		x	x
host_topic_15	x		x
host_topic_17	x	x	x
host_topic_18		x	x
host_topic_24	x	x	x
host_topic_25			x
host_topic_26			x
host_topic_27			x
host_topic_33			x
host_topic_35			x
host_topic_38			x
host_topic_39			x
host_topic_41			x
host_topic_43			x
host_topic_44			x
host_topic_45	x	x	x
host_topic_46	x	x	x
host_topic_47	x	x	
host_topic_49			x
host_tr_tag_count	x	x	x
host_tr_tag_frequency	x	x	
host_type_count	x		
host_ul_tag_count	x	x	x
host_ul_tag_frequency	x	x	
img_tag_count			x
info_url_keyword_count			x
input_tag_count		x	
mail_keyword_count		x	
mail_url_keyword_count		x	x
password_field_count	x	x	x
percent_sign_frequency			x
shopping1_url_keyword_count			x
social_url_keyword_count			x
topic_17	x	x	x
topic_24	x	x	

4.3 Classifier Updates

Looking at the keywords extracted from the web pages in the training set, there seem to be many product names listed as typical keywords on Shopping web pages and also on Social Networks. These seem to be products that were new or popular at the time the web page was visited, so the keywords for Shopping web pages may change over time as different products become popular. In general, the topics found in the web page text and the keywords chosen as indicators for the topics may change a lot over time. Therefore, both feature extraction and the trained classifier need to be updated regularly in order to work well for current web pages.

In the case of the product names, it may be possible to detect if a given token (or a combination of two or more tokens) from a text is a product name. Maybe one can get a list of product names from somewhere, e.g. a semantic web tool, or identify them e.g. by their position on the web page or in a sentence. One could then add a feature that counts the occurence of product names in addition to specific keywords, thus making the feature more generic and less susceptible to changes in popular products over time. One can also download the labelled web pages more than once at different points in time and then use all these versions in order to extract keywords. This way, the keywords that only appear during a certain time will be ranked lower as they are less frequent than before, and the keywords that are always present on the web page will be ranked higher and thus be selected for our keyword list. This will make the keyword list more robust to web page changes over time.

As soon as the InUse scoring tool is available for users and has reached a certain circulation, it is likely that those people who design fraudulent web pages will adapt their web pages so that the tool cannot recognize them anymore. Therefore, the InUse scoring tool must be regularly updated in order to account for this. This is similar to the regular updates that are needed for an antivirus software which needs to adapt to new viruses as well. This means that it is not a problem that the classifier also needs to be updated because updates are necessary in any case.

The tool developed for this thesis facilitates this by automating as much as possible of the process of calculating features for a new training set and training the classifier. Furthermore, an easy way to tag web pages is provided by WeKaSa, which can be used to easily annotate new sample web pages with a genre and download their HTML code for further processing.

4.4 Genres

Contrary to [SVK12], we are not necessarily restricted to just seven genres. We determine the genre automatically, so we do not present the user with a list of genres to select from and hence do not have the risk of confusing them when presenting them with too many options to choose from. It is possible that we have to ask the user for their intention sometimes, so we should not select too many genres; still, having a list that is longer than seven genres that is only presented to the user once in a while is likely to be acceptable for the user. Rather than limiting the number of genres too much, we should focus on presenting the user context-specific information that makes sense to them, i.e. to call a discussion forum a discussion forum rather than calling it a Social Network or E-Mail web page in order to have less genres. However, we still do have the limit that the number of genres needs to be a number the learning algorithm can handle. If we need many genres, we can address this problem by defining a hierarchy of genres rather than just a genre set. We can then train a classifier to choose a tfirst level genre and for each first level genre, train another classifier that assigns the second level genre. That way, every classifier has to deal with only a small number of genres instead of the full set. To classify a new web page, the first classifier would then calculate the first level genre label and based on that label, the corresponding second level classifier would be invoked to calculate the second level genre label. This way, two classifier results need to be computed in row but not the results of all the first and second level classifiers. In the InUse scoring tool, executing the two classifiers one after another will increases the calculation time but if both classifiers work efficiently, this should not be a problem.

Only field-testing the current classifier can tell us whether we have enough genres in order for the warnings to make sense to users. Therefore, analysing the list of Other alternative texts in order to extract new genres is left for future work. Some of the alternative texts suggest that the web page can be sorted into our genres as users entered the web page topic rather than its genre. Others like *Software Download* might be turned into a new genre.

4.5 Improving the Results

This section sums up the points we identified that one can look further into in order to improve the classifiers.

Essentially, there are two things one can do now: We can use the SMO classifier as is and correct the mistakes where web pages are assigned to Information Site but actually belong to another genre by observing the user's behaviour. Or we can go through the web pages, especially those from info, and maybe assign them to a different genre if that is more applicable. This would both reduce the number of instances in Information Site, and also improve the data's quality. The

feature vectors for the web pages would not have to be newly calculated when assigning web pages to another genre, with one exception: The keywords and URL keywords which are extracted for each genre from the training corpus would have to be extracted again, and the keyword scores for all web pages would have to be calculated again based on these new keywords. Apart from that, the feature vectors remain as they are. The topic model also stays the same, as it does not depend on the genre but only whether a text is from the test or training set.

Generally, the results from the Parent & Host data sets are a bit problematic because due to the restriction that both parent and host have to be in German, the number of instance was reduced. Especially for genres with not so many instances to begin with, there were not many left in the set, especially in the test set. Therefore, results on the test set may not say all that much about how the accuracy is in real life. However, they do work well in the classification, so it is a good idea to use the host information but to obtain more data and have another look at their results in order to make sure.

One important results is that we have to look in detail at the web pages labelled as Information Site and sort them into the other genres if applicable. This can improve our classifier's accuracy. But it can also mean that web pages which are frequently used to search for Information but also allow to buy things get classified as Shopping1 or Shopping2, resulting in a warning every time the user visits them via an unencrypted connection. It is important to closely look at those cases and see if warnings appear too often in cases where they are unnecessary. It may be useful to re-classify the web pages from Information Site but keep the label Information Site as some sort of secondary label, in order to keep in mind that these web pages might actually be used primarily as Information Site. One can then use this information to only warn the user when he tries to log in to such a web page.

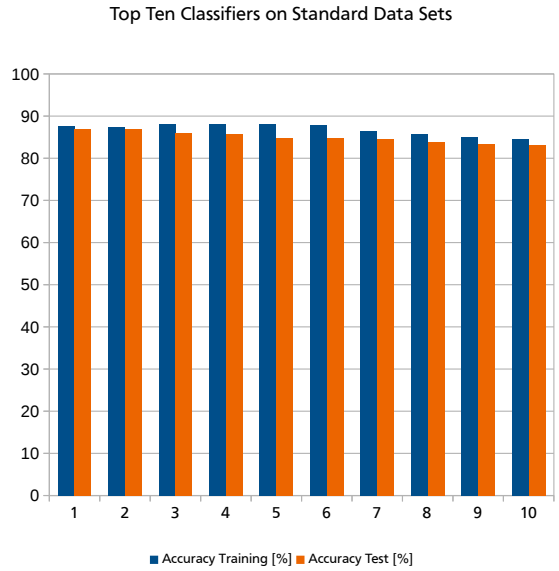
Furthermore, if we can not only assign a genre label to each web page but a genre vector with a confidence score for each genre, we can also train a classifier for just Shopping, Banking, E-Mail, Data Exchange and Social Networks. If this classifier selects one genre label with sufficient confidence, this genre label can be taken, and if for all genres the confidence score is not high enough, a genre Default can be assigned in order to express that the classifier was not able to decide on a genre label. For web pages with this Default label, we can again monitor user behaviour and base our final decision on that.

As discussed in 2.2.1, the users who labelled the web pages we used here may not be representantive for our target group. Therefore, it is possible that more data needs to be collected. If this is a problem or not can only be seen when testing the classifier in a user study with a representative target group, which was not part of this thesis. Therefore, this is one of the points to keep in mind when further working on the automatic context recognition of the InUse scoring tool: If user studies show that the classifier trained here is not good enough, the fact that the web pages were not labelled by a representative user group is one of the potential causes for the problem.

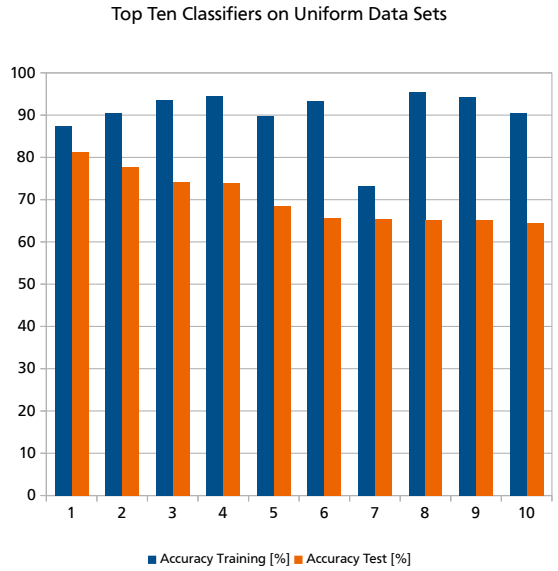
Another result with regard to the data is that correcting wrong genre labels in the original WeKaSa web page collection will probably improve the classifiers' results. As pointed out in 2.2.2, many web pages in Information Site appear to have been mislabelled from the beginning: While they appear to be web shops (or something else), users classified them as Information Site. We did not change this before training the classifiers for two reasons: Firstly, some misclassified web pages in the training data usually do not present a problem as long as enough correct instances are present. Therefore, instead of sorting them into different genres, we first trained the classifier to see if results need improvement before deciding if it makes sense to have a closer look at the web pages and maybe change their label. Secondly, as discussed in 2.2.2 as well, classifying a web shop as Information Site does make sense from the user's perspective if they visited the web shop in order to look for information and not in order to buy something. By leaving the genre label as it is for those web pages, we were hoping to train a classifier that does not only distinguish the different web page types but also captures the user's intention in some way. For example, we might be able to see a user intention if, whenever a web shop is visited to look for information, there are always search parameters in the URL, or it contains keywords like *faq* or *agb*. However, our results suggest that this did not work; in order to get information about the user's intention, we need to monitor the user's behaviour and cannot just look at static web page characteristics extracted from its URL, HTML and raw text.

If we look at the classifiers' results, we see that they share a common problem: They all misclassify web pages as Information Site when the web pages do, in fact, belong to other genres. Distinguishing the rest of the genres from each other resulted to be much easier for the classifiers. Also, some web pages in Information Site were misclassfied as something else, also that concerns only a small percentage of all web pages in Information Site. Information Sites already form a rather diverse category in the first place since they basically contain any web page where users do not log in or otherwise provide information about themselves. They can be seen as more of a default genre than an actual genre, which makes it more difficult to find typical characteristics for Information Sites. Therefore, it is possible that mislabelled web pages in the WeKaSa collection are especially problematic when they are erroneously labelled as Information Site, as this makes it impossible to distinguish Information Site from other genres. For this reason, it is recommended to repeat the approach presented here on a version of the WeKaSa web page collection where wrong labels have been corrected. In order to keep the information that the user may have visited the web page only to search for information, one should keep the original label as some kind of secondary genre label and see if this is helpful in any way.

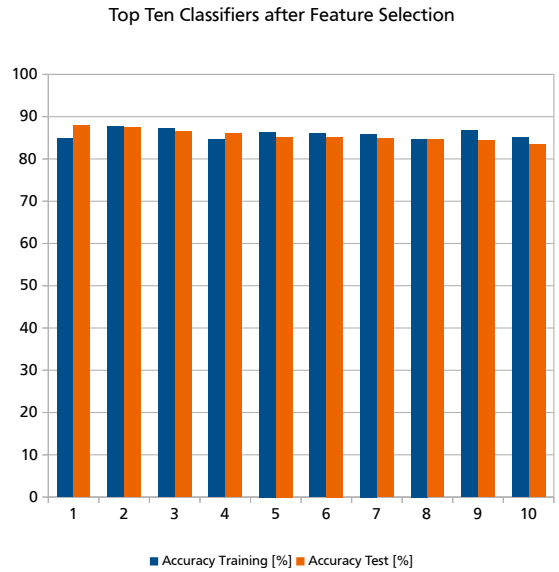
To conclude our findings with regard to the data, we now identified two things we can do to improve the classifiers' results: If the classifier does not perform sufficiently well in a user study, we can collect more data from a group of users that is more representative for our target group. And as most of the classifiers' misclassfications were web pages classified as Information Site although they were not, it is likely to improve the results if the wrongly labelled web pages in our training data are corrected and new classifiers are trained on the resulting corrected data set.



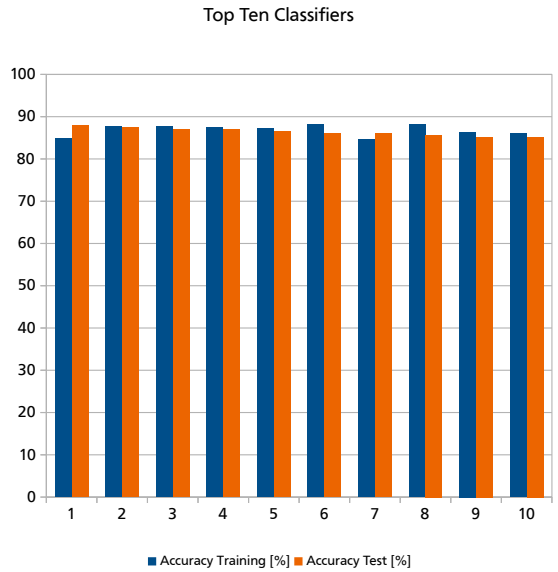
(a) Top ten classifiers on standard data sets, corresponding to the ranking in table 4.2



(b) Top ten classifiers on uniform data sets, corresponding to the ranking in table 4.4



(c) Top ten classifiers on standard data sets after Feature Selection, corresponding to the ranking in table 4.6



(d) Top ten of all classifiers on standard and uniform data sets and before and after feature selection , corresponding to the ranking in table 4.20

Figure 4.1.: Accuracy diagrams for the top ten classifiers in each ranking. The Y axis denotes the accuracy in percent while the X axis shows the classifier's rank in the corresponding ranking. The exact numbers and which rank corresponds to which classifier can be seen in the ranking table, which is referenced in each diagram's description. Note that the classifiers can have different ranks in the different rankings, i.e. the number on the X axis may refer to one classifier in one diagram and to a different one in another diagram.

5 Suggestions for the InUse Scoring Tool

This chapter addresses some of the things that are important when using the results presented here for the InUse scoring tool, as well as some things that may be interesting for future work.

5.1 Default Class

Since the InUse project also mentions the definition of a default class, two suggestions are presented here, based on what we found out in this thesis: Information Site could be used as the default class, or a dedicated default class could be used.

Looking at the training data, we see that many web pages classified as Information Site are actually web shops or something else entirely. This may not be wrong from the user's perspective, as he may have visited these web pages only to search for information. Labelling them as Information Site in that context was indeed correct. Also, web pages in Information Site are very different from each other in general. This makes sense, since an Information Site is essentially anything where the user does not log in, and a wide variety of such web pages exists. Therefore, it may be much harder to extract criteria for when a web page is an Information Site than it is for the other genres.

We could define Information Site as the default class and train a classifier on only the remaining genres. Since the web pages from the other genres are more similar to each other, the classifier might have a better accuracy. Also, we loose the problem that there are far more Information Site instances than instances in any other genre, so the resulting classifier will be less biased towards one of the genres. We can then use this classifier in the InUse scoring tool like this: If this classifier decides on a genre with a sufficiently high confidence, we use that genre. Otherwise, we assume that the web page belongs to Information Site.

The consequence of this decision is, of course, that we assume all web pages we cannot classify to be safe by default, which may not be a good idea. However, the InUse scoring tool will consist of more than the classifier we trained here, so we always have the option to change our genre assignment once the user tries to log in to the web page. This can be detected by monitoring if he starts entering data into a password field on the web page. Prototype implementations for the InUse scoring tool showed that this can easily be implemented within a Firefox extension. Also, the reaction of the Firefox extension is sufficiently quick to prevent the user from actually logging in before being warned that the web page is unsafe. Therefore, using *Information Site* as the default class when no other genre applies is a justifiable simplification of the classification problem, because in the InUse scoring tool, we can still warn the user when he tries to login.

However, the cleaner solution is to keep Information Site as a genre and to define a new class, Default, to which we assign all web pages for which we cannot predict a genre with sufficiently high confidence. The behaviour of the web browser for Default web pages could then be to act exactly as it does now, while for Information Sites, no warning or a not very obtrusive one is displayed when there is a problem with its security. We should still monitor the user's behaviour on both Default and Information Site web pages in order to be able to warn them properly when they do try to log in. And we should clean up the training data, i.e. look at the web pages in Information Site and assign them to another genre if this fits better, for example one of the Shopping genres if the web page turns out to be a web shop. This is necessary in order to reduce misclassications where a web page is assigned to Information Site but does belong elsewhere, so we can be sure that when we decide not to show a warning, the user's security really is not threatened on that web page.

5.2 Classifier Implementation

In order to use the classifiers presented here, they have to be implemented within the InUse scoring tool. For SVMs, there exist implementations both for JavaScript and Python, so one could use either one of those to re-train the SVM or to input the one from trained here directly if possible and then integrate it into either the Firefox extension or the web service. The latter would have the advantage that the SVM implementation could be used both for classifying instances and for receiving new examples and retraining the classifier. The disadvantage is that this requires the full feature vector to be send to the web service, which is not optimal for user privacy. In general, though, the feature vector does not contain so much personal information anymore; e.g. the web page's text is reduced to some statistics, not even keywords are kept directly but only a score of how many of the genre keywords (or topic keywords) appeared in the text. So it may be okay to start with using the web service at least for a beta version, and then to implement everything in the Firefox extension for the final version. In JavaScript, you can use [Kar12], but currently it only comes with a linear and RBF kernel. You can use it to train an SVM and also store the SVM as JSON for later use. In Python, you can use [pym]. It supports training of an SVM as well as storing it for later use, and it also provides an implementation of the polynomial kernel.

5.3 Supporting the Classifier

Apart from the classifier, other things are possible that help to ensure the system works well in practice.

It is a good idea to simply tag the most popular web pages with a genre by hand and not use the classifier for those. This way, the genre assigned to the most popular web pages is guaranteed to be accurate and is also available more quickly than it would be when performing feature extraction and classification first. Tagging, for example, the top 1,000 web pages by hand can be done quickly, hence the cost is rather low but the benefit is high. A modified version of WeKaSa could be used to facilitate tagging the web pages and to store the information into a database that can easily be re-used in the InUse scoring tool.

Monitoring user behaviour is also helpful. It is possible that a distinction between web shops and information sites is never fully possible, as web shops can also be visited for information search, and web pages which are primarily for information search may offer shopping possibilities. However, since in the actual tool we do not only have the static web page features but also information about the user's behaviour, e.g. if he just entered data into a password field, it makes sense to use that if in doubt.

Since a way to quickly ask the user about the web page's genre - or rather, his intentions with regard to the web page - is already implemented in WeKaSa, this can easily be used in the InUse scoring tool as well. If the classifier cannot provide a good result, and if user behaviour does indicate that there may be a risk, it is best to ask him in order to show a warning if necessary. As long as this only occures in rare cases, it will not be too annoying for the user. Plus, the information obtained in this way can be send to a server (if the user agreed to it), and then be used to improve the classifier by providing it with new examples.

Updating the classifier with new examples is also an important point for the system to work well in practice. As web pages change over time, their characteristics do, too, and things such as the most frequent keywords need to be updated.

5.4 Other Languages

Data from the WeKaSa data set, which consists of web pages users visited during normal surfing, only contains one half of pages in German, even though WeKaSa is only available in German and likely to be used by only German users. This suggests that a system that can only classify German web pages will not work well in practice, even if used only by German users. For one half of the web pages, it cannot calculate a genre. However, an inspection of the non-German web pages from the WeKaSa data set suggests that most of the remaining pages are in English, so extending the system to English pages as well will help with the classification. Since NLP tools for English are far more readily avaible than for German, this can be probably be done without much effort and is probably a good thing to do in order to make the system effective in practice.

5.5 Web Page Access

All web pages must be accessed as not logged in even if the user is logged in, since this is how the web service creating the WeKaSa data set saw the data. If this does not work well within a Firefox extension, one can always use an external web service to pull the data and do the scoring. However, this has some privacy issues and may not be desirable.

Furthermore, the web page's host needs to be downloaded in the background if its features are also used in the classification. This is also not implemented inside the tool yet. The same thing as for the original web page applies: It must be accessed not as a logged in user but as the general public would see it.

5.6 Integrating Text Feature Calculation

If possible, the calculations that the NLP web service does should be integrated into the Firefox extension. Otherwise, a lot of user data needs to be send to an external web service, which is bad practice when actually trying to improve the user's security and the safety of his private data. Also, having all calculations within the Firefox extension will likely increase response times, since only the calculation time and not also the network transmission latency is relevant then.

An important point is also that in order to see if the tool's accuracy is acceptable, one has to do a user study. Sometimes more than one genre label may work for a web page, and in other cases the correct genre may appear wrong to users. Therefore, only the user's reaction to the warnings using that genre label will tell us if the system is good enough or if it still needs improvement.

6 Conclusion and Future Work

Web Genre classification has been applied to the problem of assigning a genre to a web page that provides information about the web page context in order to improve security warnings with context information.

We trained several classifiers on several data sets and analysed their accuracy and the prediction they made. Overall, accuracy results are good for web genre classification, so we can conclude that it is possible to sort web pages into genres encoding the security context. In the cases where the classifier does not provide a correct genre label, we can often compensate this by observing user behaviour.

With regard to the choice of data set, we observed that data sets using both parents and hosts achieved better results than the ones with only the parents. Additionally, host features were favored in the feature selection as well whenever they were available. Therefore, we recommend to use the host information in our data sets.

We further observed that all genre sets work well - they all appear among the top ten in rankings. Therefore, we can choose the basic genre set which provides the most information about web page context, as classifiers worked well on this set.

Of the several learning algorithms we applied, SMO, AODE and JRip showed the best results. Since their accuracies are quite close together in the different rankings, we cannot conclude from the accuracies alone which one of them is best suited to our task. Furthermore, they all share the problem that they misclassify many instances as Information Site when they do belong to another genre. However, if we do evaluate their overall performance and look at different aspects, we do arrive at a recommendation based on our results.

Since JRip classifiers sometimes perform well but sometimes not, we conclude that they are not a good choice for our task. The reason for this is that even if JRip performs well for a data set we choose now, the fact that it performs much worse on other data sets makes it a risky choice for our application. The classification problem will change over time, for example because new training data is added, new features are designed or the set of genres is changed. Therefore, we need a classifier with a more consistent performance among different data sets, as that makes it more likely that it will also show good results for the changed but similar classification problems we may have to solve in the future.

AODE, which also performed quite well on the standard data sets, could not be computed for many of the uniform data sets because it needed to much time and space in order to do that. Since efficiency is important for our application, AODE is probably not the best choice either. Even if we do train our classifier on the standard data sets rather than the uniform data sets, AODE will likely be slower in classifying new instances than other classifiers. Furthermore, AODE needs the numeric attributes to be discretized, which is another step that can increase the computation time.

SMO performed well both on standard and uniform sets. In general, it is the learning algorithm most often found among the top ten in either classifier ranking. It did not profit from feature selection, but its accuracy did not drop dramatically either. In the paired t-test experiment, it performed significantly better than the other two algorithms on the data set with all features and significantly better than JRip and equally well as AODE on the one with only selected attributes. Since SMO showed a consistently good performance, we conclude that it is the best choice for our task. Therefore, future work should concentrate on further improving SMO's results: First, by re-training it on a clean data set where mislabelled instances in Information Site have been assigned the correct genre in order to improve the overall accuracy; Second, by applying a feature selection method that is more beneficial to SMO classifiers in order to reduce the computation time and ideally also the accuracy; and third, by finding the optimal parameter choice for training the classifier, also in order to improve the accuracy.

Once a good SMO classifier is found, it can be tested in a user study in order to see how well it works in practice. Should it not work sufficiently well, it is possible that more genres are needed in order to better capture the user's perception of context, or an adaption to other languages is necessary in order to be able to classifiy most web pages a user visits when surfing on the internet. However, this can only be seen in a user study because not the accuracy of the classifier itself is what is important for the practical implementation but rather, as how accurate the user perceives the genre assignment and how helpful the warnings using this genre are.

There are some further points that could be addressed in future work: More sophisticated features, like the presence of absence of a web shop quality seal, can be included. The classifier could be adapted to output a genre vector with probabilities instead of just one genre label. A cost function could be defined that penalizes some misclassifications more than others. Phishing web pages can be distinguished from trustworthy ones by training a classifier in a similar way.

In conclusion, we have provided a basis for automatically assigning security genres to web pages and identified the key points where improvements can be achieved and where further reseach is necessary. With some further improvements, such an automatic genre classification can be used in a browser add-on like the InUse scoring tool in order to improve security warnings by making them more understandable for the users. This will help to better protect users from the dangers that arise when using web pages that are not safe in some way or another.

Bibliography

[BNJL03] David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:2003, 2003.

[Coh95] William W. Cohen. Fast effective rule induction. In *Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[Com14] Computational Linguistics & Psycholinguistics Research Center. pattern.de, 2014. URL: *http://www.clips.ua.ac.be/pages/pattern-de* [Visited: 02/18/2014].

[Doca] Weka Documentation. weka.classifiers.bayes - Class AODE. URL: *http://weka.sourceforge.net/doc.stable/weka/classifiers/bayes/AODE.html* [Visited: 04/16/2014].

[Docb] Weka Documentation. weka.classifiers.bayes - Class NaiveBayes. URL: *http://weka.sourceforge.net/doc.stable/weka/classifiers/bayes/NaiveBayes.html* [Visited: 04/16/2014].

[Docc] Weka Documentation. weka.classifiers.functions - Class SMO. URL: *http://weka.sourceforge.net/doc.stable/weka/classifiers/functions/SMO.html* [Visited: 04/16/2014].

[Docd] Weka Documentation. weka.classifiers.rules - Class JRip. URL: *http://weka.sourceforge.net/doc.stable/weka/classifiers/rules/JRip.html* [Visited: 04/16/2014].

[Doce] Weka Documentation. weka.classifiers.rules - Class ZeroR. URL: *http://weka.sourceforge.net/doc.stable/weka/classifiers/rules/ZeroR.html* [Visited: 04/16/2014].

[JL95] George H. John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345, San Mateo, 1995. Morgan Kaufmann.

[Kar12] Karpathy, Andrej. svmjs, 2012. URL: *https://github.com/karpathy/svmjs* [Visited: 04/14/2014].

[Kar14] Karpf, Anne-Christine. Website-Kategorie-Sammler, 2014. URL: *https://ackarpf.de/download/wekasa.pdf* [Visited: 03/28/2014].

[LLK05] CS Lim, KJ Lee, and GC Kim. Multiple sets of features for automatic genre classification of web documents. *INFORMATION PROCESSING & MANAGEMENT*, 41(5):1263–1276, SEP 2005.

[Lui] Marco Lui. langid.py. URL: *https://github.com/saffsd/langid.py* [Visited: 10/16/2013].

[NLT13] NLTK Project. Natural language toolkit, 2013. URL: *http://www.nltk.org/* [Visited: 04/16/2014].

[oW] The University of Waikato. Weka 3: Data mining software in java. URL: *http://www.cs.waikato.ac.nz/ml/weka/* [Visited: 12/12/2013].

[Pla99] John C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. In *Advances in Kernel Methods-Support Vector Learning*, 1999.

[PN08] Xuan-Hieu Phan and Cam-Tu Nguyen. Jgibblda, 2008. URL: *http://jgibblda.sourceforge.net/* [Visited: 04/16/2014].

[pym] PyML - machine learning in Python. URL: *http://pyml.sourceforge.net/index.html* [Visited: 04/14/2014].

[Ron14] Armin Ronacher. Flask - web development, one drop at a time, 2014. URL: *http://flask.pocoo.org/* [Visited 04/16/2014].

[Sah96] Mehran Sahami. Learning limited dependence bayesian classifiers. In *In KDD-96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 335–338. AAAI Press, 1996.

[SEA+09] Joshua Sunshine, Serge Egelman, Hazim Almuhimedi, Neha Atri, and Lorrie Faith Cranor. Crying wolf: An empirical study of ssl warning effectiveness. usenix security, 2009.

[SVK12] Christoph Seikel, Melanie Volkamer, and Michaela Kauer. A new generation of browser security warnings. 2012.

[VBBK11] Volkamer, Melanie, Bruder, Ralph, Buchmann, Johannes, and Kauer, Michaela. Benutzerunterstützung zur Bewertung der Vertrauenswürdigkeit von Webseiten und Webshops (Internet Usage Support: InUse). July 2011.

[WBW05] Geoffrey I. Webb, Janice R. Boughton, and Zhihai Wang. Not so naive bayes: Aggregating one-dependence estimators. *Machine Learning*, 58(1):5–24, 2005.

[zES04] Sven Meyer zu Eissen and Benno Stein. Genre classification of web pages: User study and feasibility analysis. In *IN: BIUNDO S., FRUHWIRTH T., PALM G. (EDS.): ADVANCES IN ARTIFICIAL INTELLIGENCE*, pages 256–269. Springer, 2004.

List of Figures

List of Tables

A List of Features

In table A.1, you can find the full list of features we used to describe the web pages. They are sorted alphabetically according to their name.

Table A.1.: Features

Feature	Description	Type	Group
a_tag_count	Number of A tags	Numeric	HTML Tags
a_tag_frequency	Number of A tags divided by total number of HTML tags	Numeric	HTML Tags
average_tokens_per_sentence	Number of tokens divided by number of sentences	Numeric	Text Statistics
banking_keyword_count	Number of Online Banking Keywords found in the text	Numeric	Keywords
banking_url_keyword_count	Number of Online Banking URL keywords found in the URL	Numeric	URL Keywords
br_tag_count	Number of BR tags	Numeric	HTML Tags
br_tag_frequency	Number of BR tags divided by total number of HTML tags	Numeric	HTML Tags
button_tag_count	Number of BUTTON tags	Numeric	HTML Tags
button_tag_frequency	Number of BUTTON tags divided by total number of HTML tags	Numeric	HTML Tags
colon_count	Number of colons in the text	Numeric	Punctuation
colon_frequency	Number of colons divided by total number of punctuation symbols	Numeric	Punctuation
comma_count	Number of commas in the text	Numeric	Punctuation
comma_frequency	Number of commas divided by total number of punctuation symbols	Numeric	Punctuation
dash_count	Number of dashes in the text	Numeric	Punctuation
dash_frequency	Number of dashes divided by total number of punctuation symbols	Numeric	Punctuation
data_keyword_count	Number of Data Exchange keywords found in the text	Numeric	Keywords
data_url_keyword_count	Number Data Exchange URL keywords found in the URL	Numeric	URL Keywords
div_tag_count	Number of DIV tags	Numeric	HTML Tags
div_tag_frequency	Number of DIV tags divided by total number of HTML tags	Numeric	HTML Tags
double_quotation_mark_count	Number of double quotation marks in the text	Numeric	Punctuation
double_quotation_mark_frequency	Number of double quotation marks divided by total number of punctuation symbols	Numeric	Punctuation
ellipsis_count	Number of ellipses (...) in the text	Numeric	Punctuation
ellipsis_frequency	Number of ellipses (...) divided by total number of punctuation symbols	Numeric	Punctuation
exclamation_mark_count	Number of exclamation marks in the text	Numeric	Punctuation
exclamation_mark_frequency	Number of exclamation marks divided by total number of punctuation symbols	Numeric	Punctuation
external_link_count	Number of links referencing a site with a different host than this site	Numeric	Links
form_tag_count	Number of FORM tags	Numeric	HTML Tags
form_tag_frequency	Number of FORM tags divided by total number of HTML tags	Numeric	HTML Tags

Continued on next page

Feature	Description	Type	Group
frame_tag_count	Number of FRAME tags	Numeric	HTML Tags
frame_tag_frequency	Number of FRAME tags divided by total number of HTML tags	Numeric	HTML Tags
frameset_tag_count	Number of FRAMESET tags	Numeric	HTML Tags
frameset_tag_frequency	Number of FRAMESET tags divided by total number of HTML tags	Numeric	HTML Tags
genre	Genre of the web page	Nominal	Genre
greater_than_count	Number of greater than symbols in the text	Numeric	Punctuation
greater_than_frequency	Number of greater than symbols divided by total number of punctuation symbols	Numeric	Punctuation
h1_tag_count	Number of H1 tags	Numeric	HTML Tags
h1_tag_frequency	Number of H1 tags divided by total number of HTML tags	Numeric	HTML Tags
h2_tag_count	Number of H2 tags	Numeric	HTML Tags
h2_tag_frequency	Number of H2 tags divided by total number of HTML tags	Numeric	HTML Tags
h3_tag_count	Number of H3 tags	Numeric	HTML Tags
h3_tag_frequency	Number of H3 tags divided by total number of HTML tags	Numeric	HTML Tags
h4_tag_count	Number of H4 tags	Numeric	HTML Tags
h4_tag_frequency	Number of H4 tags divided by total number of HTML tags	Numeric	HTML Tags
h5_tag_count	Number of H5 tags	Numeric	HTML Tags
h5_tag_frequency	Number of H5 tags divided by total number of HTML tags	Numeric	HTML Tags
h6_tag_count	Number of H6 tags	Numeric	HTML Tags
h6_tag_frequency	Number of H6 tags divided by total number of HTML tags	Numeric	HTML Tags
hr_tag_count	Number of HR tags	Numeric	HTML Tags
hr_tag_frequency	Number of HR tags divided by total number of HTML tags	Numeric	HTML Tags
html_tag_count	Total number of HTML tags (only the HTML tags for which we calculated the count and frequency)	Numeric	HTML Tags
iframe_tag_count	Number of IFRAME tags	Numeric	HTML Tags
iframe_tag_frequency	Number of IFRAME tags divided by total number of HTML tags	Numeric	HTML Tags
img_tag_count	Number of IMG tags	Numeric	HTML Tags
img_tag_frequency	Number of IMG tags divided by total number of HTML tags	Numeric	HTML Tags
info_keyword_count	Number of Information Site keywords found in the text	Numeric	Keywords
info_url_keyword_count	Number of Information Site URL keywords found in the URL	Numeric	URL Keywords
input_tag_count	Number of INPUT tags	Numeric	HTML Tags
input_tag_frequency	Number of INPUT tags divided by total number of HTML tags	Numeric	HTML Tags
internal_link_count	Number of links referencing a site with the same host as this site	Numeric	Links
javascript_link_count	Number of links referencing JavaScript code	Numeric	Links
left_curly_bracket_count	Number of left curly brackets in the text	Numeric	Punctuation
left_curly_bracket_frequency	Number of left curly brackets divided by total number of punctuation symbols	Numeric	Punctuation
left_parenthesis_count	Number of left parentheses in the text	Numeric	Punctuation

Continued on next page

Feature	Description	Type	Group
left_parenthesis_frequency	Number of left parentheses divided by total number of punctuation symbols	Numeric	Punctuation
left_square_bracket_count	Number of left square brackets in the text	Numeric	Punctuation
left_square_bracket_frequency	Number of left square brackets divided by total number of punctuation symbols	Numeric	Punctuation
lexical_diversity	Lexical Diversity of the text	Numeric	Text Statistics
li_tag_count	Number of LI tags	Numeric	HTML Tags
li_tag_frequency	Number of LI tags divided by total number of HTML tags	Numeric	HTML Tags
link_count	Number of links	Numeric	Links
link_tag_count	Number of LINK tags	Numeric	HTML Tags
link_tag_frequency	Number of LINK tags divided by total number of HTML tags	Numeric	HTML Tags
mail_keyword_count	Number of E-Mail keywords found in the text	Numeric	Keywords
mail_link_count	Number of links to e-mail-addresses	Numeric	Links
mail_url_keyword_count	Number of E-Mail URL keywords found in the URL	Numeric	URL Keywords
ol_tag_count	Number of OL tags	Numeric	HTML Tags
ol_tag_frequency	Number of OL tags divided by total number of HTML tags	Numeric	HTML Tags
option_tag_count	Number of OPTION tags	Numeric	HTML Tags
option_tag_frequency	Number of OPTION tags divided by total number of HTML tags	Numeric	HTML Tags
other_keyword_count	Number of Other keywords found in the text	Numeric	Keywords
other_url_keyword_count	Number of Other URL keywords found in the URL	Numeric	URL Keywords
p_tag_count	Number of P tags	Numeric	HTML Tags
p_tag_frequency	Number of P tags divided by total number of HTML tags	Numeric	HTML Tags
page_id	The page ID	Numeric	ID
password_field_count	Number of password fields (input fields of type password) on this site	Numeric	Form Fields
percent_sign_count	Number of percent signs in the text	Numeric	Punctuation
percent_sign_frequency	Number of percent signs divided by total number of punctuation symbols	Numeric	Punctuation
period_count	Number of peridos in the text (not counting those appearing as part of an ellipsis)	Numeric	Punctuation
period_frequency	Number of periods (without ellipsis) divided by total number of punctuation symbols	Numeric	Punctuation
pos_cc_count	Number of tokens with the POS tag CC	Numeric	POS Tags
pos_cc_frequency	Frequency of tokens with the POS tag CC	Numeric	POS Tags
pos_cd_count	Number of tokens with the POS tag CD	Numeric	POS Tags
pos_cd_frequency	Frequency of tokens with the POS tag CD	Numeric	POS Tags
pos_colon_count	Number of tokens with the POS tag : (colon)	Numeric	POS Tags
pos_colon_frequency	Frequency of tokens with the POS tag : (colon)	Numeric	POS Tags
pos_comma_count	Number of tokens with the POS tag , (comma)	Numeric	POS Tags
pos_comma_frequency	Frequency of tokens with the POS tag , (comma)	Numeric	POS Tags
pos_dt_count	Number of tokens with the POS tag DT	Numeric	POS Tags
pos_dt_frequency	Frequency of tokens with the POS tag DT	Numeric	POS Tags
pos_ex_count	Number of tokens with the POS tag EX	Numeric	POS Tags
pos_ex_frequency	Frequency of tokens with the POS tag EX	Numeric	POS Tags

Continued on next page

Feature	Description	Type	Group
pos_fw_count	Number of tokens with the POS tag FW	Numeric	POS Tags
pos_fw_frequency	Frequency of tokens with the POS tag FW	Numeric	POS Tags
pos_in_count	Number of tokens with the POS tag IN	Numeric	POS Tags
pos_in_frequency	Frequency of tokens with the POS tag IN	Numeric	POS Tags
pos_jj_count	Number of tokens with the POS tag JJ	Numeric	POS Tags
pos_jj_frequency	Frequency of tokens with the POS tag JJ	Numeric	POS Tags
pos_jjr_count	Number of tokens with the POS tag JJR	Numeric	POS Tags
pos_jjr_frequency	Frequency of tokens with the POS tag JJR	Numeric	POS Tags
pos_jjs_count	Number of tokens with the POS tag JJS	Numeric	POS Tags
pos_jjs_frequency	Frequency of tokens with the POS tag JJS	Numeric	POS Tags
pos_leftpar_count	Number of tokens with the POS tag ((left parenthesis)	Numeric	POS Tags
pos_leftpar_frequency	Frequency of tokens with the POS tag ((left parenthesis)	Numeric	POS Tags
pos_ls_count	Number of tokens with the POS tag LS	Numeric	POS Tags
pos_ls_frequency	Frequency of tokens with the POS tag LS	Numeric	POS Tags
pos_md_count	Number of tokens with the POS tag MD	Numeric	POS Tags
pos_md_frequency	Frequency of tokens with the POS tag MD	Numeric	POS Tags
pos_nn_count	Number of tokens with the POS tag NN	Numeric	POS Tags
pos_nn_frequency	Frequency of tokens with the POS tag NN	Numeric	POS Tags
pos_nnp_count	Number of tokens with the POS tag NNP	Numeric	POS Tags
pos_nnp_frequency	Frequency of tokens with the POS tag NNP	Numeric	POS Tags
pos_nnps_count	Number of tokens with the POS tag NNPS	Numeric	POS Tags
pos_nnps_frequency	Frequency of tokens with the POS tag NNPS	Numeric	POS Tags
pos_nns_count	Number of tokens with the POS tag NNS	Numeric	POS Tags
pos_nns_frequency	Frequency of tokens with the POS tag NNS	Numeric	POS Tags
pos_pdt_count	Number of tokens with the POS tag PDT	Numeric	POS Tags
pos_pdt_frequency	Frequency of tokens with the POS tag PDT	Numeric	POS Tags
pos_period_count	Number of tokens with the POS tag . (period)	Numeric	POS Tags
pos_period_frequency	Frequency of tokens with the POS tag . (period)	Numeric	POS Tags
pos_pos_count	Number of tokens with the POS tag POS	Numeric	POS Tags
pos_pos_frequency	Frequency of tokens with the POS tag POS	Numeric	POS Tags
pos_prp_count	Number of tokens with the POS tag PRP	Numeric	POS Tags
pos_prp_frequency	Frequency of tokens with the POS tag PRP	Numeric	POS Tags
pos_prpz_count	Number of tokens with the POS tag PRP$	Numeric	POS Tags
pos_prpz_frequency	Frequency of tokens with the POS tag PRP$	Numeric	POS Tags
pos_rb_count	Number of tokens with the POS tag RB	Numeric	POS Tags
pos_rb_frequency	Frequency of tokens with the POS tag RB	Numeric	POS Tags
pos_rbr_count	Number of tokens with the POS tag RBR	Numeric	POS Tags
pos_rbr_frequency	Frequency of tokens with the POS tag RBR	Numeric	POS Tags
pos_rbs_count	Number of tokens with the POS tag RBS	Numeric	POS Tags
pos_rbs_frequency	Frequency of tokens with the POS tag RBS	Numeric	POS Tags
pos_rightpar_count	Number of tokens with the POS tag) (right parenthesis)	Numeric	POS Tags
pos_rightpar_frequency	Frequency of tokens with the POS tag) (right parenthesis)	Numeric	POS Tags
pos_rp_count	Number of tokens with the POS tag RP	Numeric	POS Tags
pos_rp_frequency	Frequency of tokens with the POS tag RP	Numeric	POS Tags
pos_sym_count	Number of tokens with the POS tag SYM	Numeric	POS Tags
pos_sym_frequency	Frequency of tokens with the POS tag SYM	Numeric	POS Tags
pos_to_count	Number of tokens with the POS tag TO	Numeric	POS Tags
pos_to_frequency	Frequency of tokens with the POS tag TO	Numeric	POS Tags

Continued on next page

Feature	Description	Type	Group
pos_uh_count	Number of tokens with the POS tag UH	Numeric	POS Tags
pos_uh_frequency	Frequency of tokens with the POS tag UH	Numeric	POS Tags
pos_vb_count	Number of tokens with the POS tag VB	Numeric	POS Tags
pos_vb_frequency	Frequency of tokens with the POS tag VB	Numeric	POS Tags
pos_vbd_count	Number of tokens with the POS tag VBD	Numeric	POS Tags
pos_vbd_frequency	Frequency of tokens with the POS tag VBD	Numeric	POS Tags
pos_vbg_count	Number of tokens with the POS tag VBG	Numeric	POS Tags
pos_vbg_frequency	Frequency of tokens with the POS tag VBG	Numeric	POS Tags
pos_vbn_count	Number of tokens with the POS tag VBN	Numeric	POS Tags
pos_vbn_frequency	Frequency of tokens with the POS tag VBN	Numeric	POS Tags
pos_vbp_count	Number of tokens with the POS tag VBP	Numeric	POS Tags
pos_vbp_frequency	Frequency of tokens with the POS tag VBP	Numeric	POS Tags
pos_vbz_count	Number of tokens with the POS tag VBZ	Numeric	POS Tags
pos_vbz_frequency	Frequency of tokens with the POS tag VBZ	Numeric	POS Tags
pos_wdt_count	Number of tokens with the POS tag WDT	Numeric	POS Tags
pos_wdt_frequency	Frequency of tokens with the POS tag WDT	Numeric	POS Tags
pos_wp_count	Number of tokens with the POS tag WP	Numeric	POS Tags
pos_wp_frequency	Frequency of tokens with the POS tag WP	Numeric	POS Tags
pos_wpz_count	Number of tokens with the POS tag WP$	Numeric	POS Tags
pos_wpz_frequency	Frequency of tokens with the POS tag WP$	Numeric	POS Tags
pos_wrb_count	Number of tokens with the POS tag WRB	Numeric	POS Tags
pos_wrb_frequency	Frequency of tokens with the POS tag WRB	Numeric	POS Tags
punctuation_count	Number of colons in the text	Numeric	Punctuation
question_mark_count	Number of colons divided by total number of punctuation symbols	Numeric	Punctuation
question_mark_frequency	Number of colons in the text	Numeric	Punctuation
right_curly_bracket_count	Number of colons divided by total number of punctuation symbols	Numeric	Punctuation
right_curly_bracket_frequency	Number of colons in the text	Numeric	Punctuation
right_parenthesis_count	Number of colons divided by total number of punctuation symbols	Numeric	Punctuation
right_parenthesis_frequency	Number of colons in the text	Numeric	Punctuation
right_square_bracket_count	Number of colons divided by total number of punctuation symbols	Numeric	Punctuation
right_square_bracket_frequency	Number of colons in the text	Numeric	Punctuation
robots_allowed	1 if robots were allowed on the web page, 0 if not	Numeric	Query
script_tag_count	Number of SCRIPT tags	Numeric	HTML Tags
script_tag_frequency	Number of SCRIPT tags divided by total number of HTML tags	Numeric	HTML Tags
select_tag_count	Number of SELECT tags	Numeric	HTML Tags
select_tag_frequency	Number of SELECT tags divided by total number of HTML tags	Numeric	HTML Tags
semicolon_count	Number of colons divided by total number of punctuation symbols	Numeric	Punctuation
semicolon_frequency	Number of colons in the text	Numeric	Punctuation
sentence_count	Number of sentences in the text	Numeric	Text Statistics
shopping1_keyword_count	Number of Shopping 1 keywords in the text	Numeric	Keywords
shopping1_url_keyword_count	Number of Shopping 1 URL keywords in the URL	Numeric	URL Keywords
shopping2_keyword_count	Number of Shopping 2 keywords in the text	Numeric	Keywords
shopping2_url_keyword_count	Number of Shopping 2 URL keywords in the URL	Numeric	URL Keywords
single_quotation_mark_count	Number of colons in the text	Numeric	Punctuation

Continued on next page

Feature	Description	Type	Group
single_quotation_mark_frequency	Number of colons divided by total number of punctuation symbols	Numeric	Punctuation
smaller_than_count	Number of colons in the text	Numeric	Punctuation
smaller_than_frequency	Number of colons divided by total number of punctuation symbols	Numeric	Punctuation
social_keyword_count	Number of Social Network keywords in the text	Numeric	Keywords
social_url_keyword_count	Number of Social Network URL keywords in the URL	Numeric	URL Keywords
style_tag_count	Number of STYLE tags	Numeric	HTML Tags
style_tag_frequency	Number of STYLE tags divided by total number of HTML tags	Numeric	HTML Tags
table_tag_count	Number of TABLE tags	Numeric	HTML Tags
table_tag_frequency	Number of TABLE tags divided by total number of HTML tags	Numeric	HTML Tags
tbody_tag_count	Number of TBODY tags	Numeric	HTML Tags
tbody_tag_frequency	Number of TBODY tags divided by total number of HTML tags	Numeric	HTML Tags
td_tag_count	Number of TD tags	Numeric	HTML Tags
td_tag_frequency	Number of TD tags divided by total number of HTML tags	Numeric	HTML Tags
textarea_tag_count	Number of TEXTAREA tags	Numeric	HTML Tags
textarea_tag_frequency	Number of TEXTAREA tags divided by total number of HTML tags	Numeric	HTML Tags
th_tag_count	Number of TH tags	Numeric	HTML Tags
th_tag_frequency	Number of TH tags divided by total number of HTML tags	Numeric	HTML Tags
thead_tag_count	Number of THEAD tags	Numeric	HTML Tags
thead_tag_frequency	Number of THEAD tags divided by total number of HTML tags	Numeric	HTML Tags
token_count	Number of tokens in the text	Numeric	Text Statistics
topic_00	The text's score for Topic 0 from the Topic Model	Numeric	Topic Model
topic_01	The text's score for Topic 1 from the Topic Model	Numeric	Topic Model
topic_02	The text's score for Topic 2 from the Topic Model	Numeric	Topic Model
topic_03	The text's score for Topic 3 from the Topic Model	Numeric	Topic Model
topic_04	The text's score for Topic 4 from the Topic Model	Numeric	Topic Model
topic_05	The text's score for Topic 5 from the Topic Model	Numeric	Topic Model
topic_06	The text's score for Topic 6 from the Topic Model	Numeric	Topic Model
topic_07	The text's score for Topic 7 from the Topic Model	Numeric	Topic Model
topic_08	The text's score for Topic 8 from the Topic Model	Numeric	Topic Model
topic_09	The text's score for Topic 9 from the Topic Model	Numeric	Topic Model
topic_10	The text's score for Topic 10 from the Topic Model	Numeric	Topic Model
topic_11	The text's score for Topic 11 from the Topic Model	Numeric	Topic Model

Continued on next page

Feature	Description	Type	Group
topic_12	The text's score for Topic 12 from the Topic Model	Numeric	Topic Model
topic_13	The text's score for Topic 13 from the Topic Model	Numeric	Topic Model
topic_14	The text's score for Topic 14 from the Topic Model	Numeric	Topic Model
topic_15	The text's score for Topic 15 from the Topic Model	Numeric	Topic Model
topic_16	The text's score for Topic 16 from the Topic Model	Numeric	Topic Model
topic_17	The text's score for Topic 17 from the Topic Model	Numeric	Topic Model
topic_18	The text's score for Topic 18 from the Topic Model	Numeric	Topic Model
topic_19	The text's score for Topic 19 from the Topic Model	Numeric	Topic Model
topic_20	The text's score for Topic 20 from the Topic Model	Numeric	Topic Model
topic_21	The text's score for Topic 21 from the Topic Model	Numeric	Topic Model
topic_22	The text's score for Topic 22 from the Topic Model	Numeric	Topic Model
topic_23	The text's score for Topic 23 from the Topic Model	Numeric	Topic Model
topic_24	The text's score for Topic 24 from the Topic Model	Numeric	Topic Model
topic_25	The text's score for Topic 25 from the Topic Model	Numeric	Topic Model
topic_26	The text's score for Topic 26 from the Topic Model	Numeric	Topic Model
topic_27	The text's score for Topic 27 from the Topic Model	Numeric	Topic Model
topic_28	The text's score for Topic 28 from the Topic Model	Numeric	Topic Model
topic_29	The text's score for Topic 29 from the Topic Model	Numeric	Topic Model
topic_30	The text's score for Topic 30 from the Topic Model	Numeric	Topic Model
topic_31	The text's score for Topic 31 from the Topic Model	Numeric	Topic Model
topic_32	The text's score for Topic 32 from the Topic Model	Numeric	Topic Model
topic_33	The text's score for Topic 33 from the Topic Model	Numeric	Topic Model
topic_34	The text's score for Topic 34 from the Topic Model	Numeric	Topic Model
topic_35	The text's score for Topic 35 from the Topic Model	Numeric	Topic Model
topic_36	The text's score for Topic 36 from the Topic Model	Numeric	Topic Model
topic_37	The text's score for Topic 37 from the Topic Model	Numeric	Topic Model
topic_38	The text's score for Topic 38 from the Topic Model	Numeric	Topic Model

Continued on next page

Feature	Description	Type	Group
topic_39	The text's score for Topic 39 from the Topic Model	Numeric	Topic Model
topic_40	The text's score for Topic 40 from the Topic Model	Numeric	Topic Model
topic_41	The text's score for Topic 41 from the Topic Model	Numeric	Topic Model
topic_42	The text's score for Topic 42 from the Topic Model	Numeric	Topic Model
topic_43	The text's score for Topic 43 from the Topic Model	Numeric	Topic Model
topic_44	The text's score for Topic 44 from the Topic Model	Numeric	Topic Model
topic_45	The text's score for Topic 45 from the Topic Model	Numeric	Topic Model
topic_46	The text's score for Topic 46 from the Topic Model	Numeric	Topic Model
topic_47	The text's score for Topic 47 from the Topic Model	Numeric	Topic Model
topic_48	The text's score for Topic 48 from the Topic Model	Numeric	Topic Model
topic_49	The text's score for Topic 49 from the Topic Model	Numeric	Topic Model
tr_tag_count	Number of TR tags	Numeric	HTML Tags
tr_tag_frequency	Number of TR tags divided by total number of HTML tags	Numeric	HTML Tags
type_count	Number of types, i.e. pairwise distinct tokens in the text	Numeric	Text Statistics
type_token_ratio	Type-Token-Ratio, i.e. number of types divided by number of tokens	Numeric	Text Statistics
ul_tag_count	Number of UL tags	Numeric	HTML Tags
ul_tag_frequency	Number of UL tags divided by total number of HTML tags	Numeric	HTML Tags
url_contains_query_parameters	1 if the URL contains query parameters, 0 otherwise	Numeric	URL Complexity
url_domain_depth	Domain levels	Numeric	URL Complexity
url_path_depth	Path levels	Numeric	URL Complexity