Bachelor Thesis

Researcher Homepage Identification and Name Extraction

Application of Machine Learning with Multiple Views

Marc Ingold

Examiner : Prof. Dr. Hannah Bast

Adviser : Prof. Dr. Hannah Bast

Albert Ludwigs University, Freiburg im Breisgau
Faculty of Engineering
Department of Computer Science
Chair of Algorithms and Data Structures

July 26, 2019

Writing Period

15.05.2019 - 15.08.2019

Examiner

Prof. Dr. Hannah Bast

Adviser

Prof. Dr. Hannah Bast

Declaration

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work. I hereby also declare, that my thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Place, Date	Signature

Abstract

The topic of this thesis is the implementation and evaluation of a machine learning approach for the information extraction from researcher homepages. The key aspects in this work are the data acquisition from the non-profit organization Common Crawl and the development and assessment of two machine learning models. A model for the identification of researcher homepages from a dataset of arbitrary web documents and a model for the extraction of information from the researchers homepage. Exemplary, the researchers name was extracted. The algorithm choice for both models was restricted to methods that are fast to train and straight forward to interpret such as the Random Forest and linear models with Stochastic Gradient Descent learning. The classification of the web document type was done using a multi-view approach. On the basis of two disjoint feature sets, namely the URL surface patterns and the web page content features, two machine learning models were trained. The final prediction was obtained by combining both of the models. A F1 score of 68% was achieved for the identification of researcher homepages. The extraction of the researchers name was done by extending a simple heuristic with machine learning features. An increase of 6 percentage points in F1 score was achieved in comparison to the heuristic, resulting in a F1 score of 94% for the final researcher name extraction model.

Zusammenfassung

In dieser Thesis wird ein Ansatz zur Extraktion von Informationen von Wissenschaftler Homepages mittels maschinellem Lernen entwickelt und evaluiert. Hierzu wurden zunächst geeignete Datensätze von Common Crawl heruntergeladen und vorverarbeitet. Auf der Basis dieser Daten wurden zwei machine learning Modelle entwickelt, evaluiert und bewertet. Das erste Modell dient der Identifizierung von Wissenschaftler Homepages und basiert auf dem maschinellen Lernen mit mehreren Sichten (Multi-View Learning). Das zweite Modell dient der Extraktion von Informationen von den gefunden Homepages. Exemplarisch wurde der Name des Wissenschaftlers extrahiert. Hierzu wurde eine einfache Heuristik durch maschinelles Lernen erweitert. Die untersuchten machine learning Algorithmen sind Random Forest und lineare Modelle mit Optimierung durch stochastischen Gradienten-Abstieg. Diese Algorithmen lassen sich schnell trainieren und leichter interpretieren als komplexere Methoden. Bei der Identifizierung der Wissenschaftler Homepages wurde ein F1 Wert von 68% erreicht. Bei der Extraktion des Namens wurde ein F1 Wert von 94% erreicht. Dies ist eine Verbesserung von 6 Prozentpunkten im Vergleich zu der Heuristik.

Contents

List of Tables

List of Figures

1. Introduction

Since the early stages of the internet as a research project at CERN, it became the largest collection of unstructured data and the main means of information exchange. It is reported that in 2018 the web consisted of around 200 million active websites and 7,5 times more inactive websites [1]. One of the challenges regarding the internet from a computer science perspective is the automated extraction of information from this massive collection of unstructured data (e.g. images, videos and natural language texts).

In this work, the application of a machine learning approach for the identification of researcher homepages and the extraction of the researcher's name was investigated. The homepage identification was strongly inspired by Gollapalli et al. [2, 3]. They utilise two separate views on the web page data, the page content and the URL surface patterns to identify researcher homepages in a binary classification manner. For each of the views, a separate machine learning model was trained and evaluated. The prediction, if a web document is a homepage or not was then made by a combination of the two models. The homepage owner identification was also formulated as a binary classification task. A machine learning classifier was trained, that estimates the probability for each name on the page to be the name of the homepage owner.

A pipeline was developed, which covers the initial data acquisition, the preparation of the data to be usable in the machine learning algorithms and the application of the models. The result of this pipeline was a dataset of predicted homepages with the predicted homepage owner and university for a given list of web page URLs or domains.

The main focus of this thesis is the presentation of the development process and the assessment of the researcher homepage and homepage owner identification pipeline. Following the introductory chapters (1. Introduction and 2. Related Work), the theoretical foundations are explained in chapter 3. In chapter 4 the acquisition of html data from the non profit organization Common Crawl [4] is described. In chapters 5 and 6 the applied data sampling, feature extraction, used machine learning algorithms and results of the machine learning models for the identification of researcher homepages and homepage owners are presented, analysed and evaluated. Chapter 7 gives an idea of the expected runtime of the pipeline. The work is concluded in chapter 8 with the overall verdict and the outline of possible future work.

1.1. Problem Description

When extracting information from web pages with a machine learning approach, three main tasks have to be solved. Firstly, the web page data has to be obtained. Secondly, the web pages of interest have to be identified. Thirdly, the desired information has to be extracted.

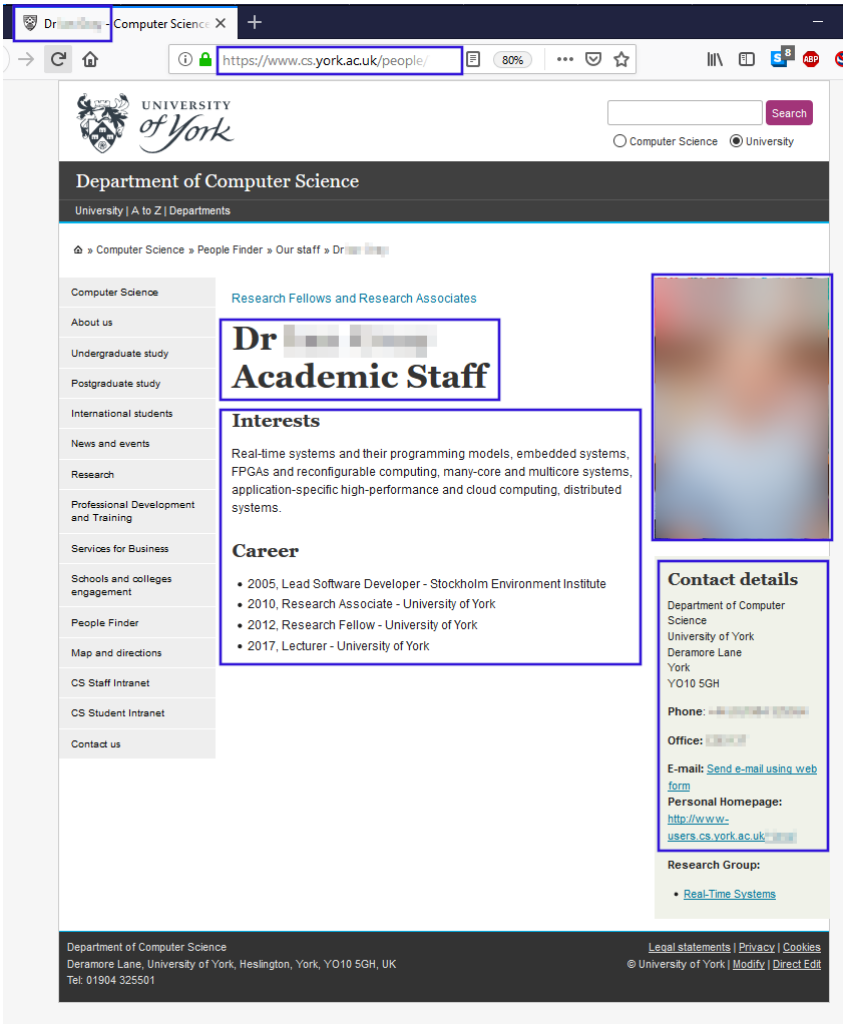

Figure 1: *Exemplary researcher homepage. Elements of the rendered web page considered fruitful for the identification of homepages are marked in blue.*

The data acquisition task can be described as follows: For a given list of URLs or domains, find, download and filter the html data for each URL or all the web pages in that domain. With the use of existing datasets, bibliographic databases and manual labour, generate a dataset which is usable in supervised machine learning algorithms, i.e. the data contains labelled homepage examples and counter examples. After the data acquisition, the html data typically has to be transformed to a representation that is favourable for the machine learning task. For each web page, a set of properties was extracted from the html, which was assumed to make the different types of web pages separable. These properties are called features. For web pages there are many possible feature types and feature combinations.

Lets consider the rendered homepage as it is displayed in the web browser. Figure 1 illustrates a typical researcher homepage. Key page elements, that could support the classification of the web page type, can be easily found.

The text in the title area or headings of the page usually contain the academic degree and the name of the person. Often times there is a portrait and the contact details of the researcher. Additionally, text segments about the area of research, the career path and publications can be frequently found on these homepages. The range of different combinations of those page elements in different homepages is quite large. In some homepages only the name and email address of the researcher are displayed; in others the information about the researcher is distributed over multiple pages. Web documents, furthermore, have special properties, that distinguish them from text documents. Web pages are semi-structured documents. Semi-structured means that there is no fixed data structure, which applies to each type of document. The web document itself contains part of its structuring information in the hypertext markup language code. Tags with specific meanings (e.g. title, heading, link, image, table etc.) augment the content elements in the page. Some of the tags define how the content should be rendered by the browser, whereas others define the behaviour of page elements on user interaction. Most of the tags are not required for a valid html page and the combination and order of tags is arbitrary. Also, the visual appearance and content displayed in the browser can be influenced by Cascading Style Sheets (Css) and Java Script functions. Lastly, web documents are directly interconnected with other web documents through links or indirectly through its position in the domain. In addition to the different feature sets, which could be directly extracted from the web document, there is information about the web page on the surrounding web pages and the URL. Those features could also be used for a web page classification.

So, there are on-page features like the text content, which might or might not be augmented with the meaning of the accompanying tag. There is the rendered image of the web page and the images contained in the web page. There are the links leading to internal or external web documents etc. Then, there are features from other web documents in the vicinity of the page, which could be useful, if the web document itself does not contain enough information.

The decision which features should be utilised together with the selection of the machine learning algorithm are basic steps in any supervised machine learning approach. Generally, they are solved iteratively through experimentation. For web documents this general approach was supplemented slightly. The diversity of feature types makes web document classification a natural contender for multi-view learning approaches. In multi-view learning, multiple disjoint sets of features about the same object are used. Each view is modelled separately. The URL, the text content, the rendered web page image etc. each are different views of the same web document. Models trained on the different views are consolidated in the end to produce the prediction. The prediction result of the developed model is the probability for each web document to be a researcher homepage.

With the developed data processing pipeline and machine learning model,

homepages can be identified in a dataset of previously unseen web documents. Then, the desired information from the predicted homepages can be extracted. In this work, the researchers name was extracted, which was also done using a supervised machine learning approach. Therefore, the core steps for any supervised machine learning method, namely the data acquisition, the feature extraction and selection, and the model training and evaluation, are also applied here.

The main tasks for the homepage owner identification are the extraction of person names from the text content of the homepage and the selection of the correct homepage owner name from that list of names. The first task is a named entity recognition (NER) problem. In NER, terms or short phrases, that have a meaning in addition to the sole word meaning and that appear in a unique context, like person names, geo-political entities, organizations, times and dates, monetary values etc. are identified and categorized. The selection of the homepage owner from the extracted list of person names for each web document was achieved with a binary classifier. For the training of the model, simple features were extracted for each person name. The prediction result is the probability for each name to be the homepage owner.

1.2. Implementation and Main Aspects

In this section the implementations and main aspects of this thesis are described. The implementations were mainly written in Python and the following criteria were applied: The steps of the homepage and homepage owner identification pipeline are loosely coupled. The data transfer between the steps was done using flat files. The composition of the components was managed with make files. The overall code base was wrapped in a docker container and unit testing was applied.

Figure 2 shows the workflow during the development process. The data was obtained from Common Crawl [4]. Common Crawl searches the web on a monthly basis and makes the obtained web page data and meta data publicly available. Machine learning features were extracted from the downloaded and filtered data. The resulting base dataset contained text and numeric features. Different subsets of the features were used for the development of the homepage identification model and the researcher name identification model. The latter was only based on web page text features, whereas in the former also numeric features and the URL of the web page were utilised. Samples of the base dataset were drawn and labelled for the development of the two models.

The two models for the homepage identification were developed separately, each with its own data preprocessing, feature selection, model training and model evaluation. These iterative processes were tracked with jupyter notebooks [5]. The data processing steps and machine learning models were assembled using scikit-learn pipelines [6] for ease of use and reproducibility. Different

Figure 2: *An overview of the workflow and the implementations produced during the development process.*

combinations of the individual models were composed and evaluated. With this combined model an F1 score of 68% was achieved on the test data for the homepage identification task.

For the researcher name identification, the person names appearing on the web pages of the training sample and features representing each name were extracted. As before, the preprocessing steps and the machine learning model

were assembled with scikit-learn pipelines. A F1 score of 94% was achieved on the test data for the researcher name identification task.

The best performing data preprocessing pipelines and machine learning models were stored. Wrapper classes using those scikit-learn pipeline objects were written for the sequential execution of the developed data acquisition, the homepage and the researcher name identification. The procedure of building a dataset of predicted homepages, predicted homepage owners and corresponding universities from an input list of arbitrary URLs or domains was composed with make file targets.

The key aspect of this thesis besides the described implementations is the assessment of the developed data preprocessing pipelines and prediction models for the homepage identification and the name extraction. For this purpose, the features used in the three machine learning models were analysed. Also, the common errors produced by the models were evaluated and possible improvements suggested.

A few by-products were created during the thesis. They are listed here for completeness:

- The adjustment of a web search server implementation of the University of Freiburgs' Chair of Algorithms and Data Structures for demonstration purposes. A result dataset of predicted homepages, predicted homepage owners and corresponding universities can be queried by researcher first and last name or the university.
- The implementation of helper scripts for the manual labelling of training data for the different machine learning tasks.
- A comparison of multiple publicly available Named Entity Recognition models.
- Experiments with Co-training to improve the prediction performance on unseen data.

2. Related Work

In this chapter, a short overview is given about work related to the web document classification presented in the thesis. Methods of categorizing web pages only with the URL have been presented by Kan [7] or Baykan et al. [8]. Kan introduces multiple means of segmenting and expanding URL components before training multiple binary SVMs, one for each web page type. Baykan et al. compare different URL to feature mappings and machine learning algorithms in a binary classification setup to determine if a web page belongs to a topic or not. Qi and Davison [9] review web page classification methods based on on-page web page content features and features that are extracted from neighbouring web pages. A co-training approach using URL and web page content features has been proposed by Gollapalli et al. [3]. Co-training is a semi-supervised learning technique which is used to incorporate potentially large amounts of unlabelled data in the model development based on a small amount of labelled data. Onan [10] compares ensemble methods with varying feature selection and base learners. Ensemble machine learning algorithms utilise a collective of many weak learners to obtain a robust, aggregated prediction result.

3. Theoretical Foundation

The key concepts and algorithms used in this work are explained in this chapter. Section 3.1 introduces the general idea behind machine learning with labelled data. In sections 3.2 and 3.3 common data preprocessing steps necessary to apply machine learning to text data are explained. In sections 3.4 and 3.5 two supervised machine learning algorithms for classification are presented. Finally, in section 3.6 problematic effects due to imbalanced class labels are outlined and in section 3.7 the utilised evaluation metrics are defined. The information for this chapter has been taken from [11, 12, 16, 19].

3.1. Supervised Machine Learning

In the field of machine learning, there are three main types of learning algorithms. Those are reinforcement, unsupervised and supervised learning. Reinforcement learning describes the process of learning a sequence of actions or solving a problem by interaction with a real world or virtual environment. Every action taken is evaluated and either rewarded or penalized depending on the effect of the action. Through continuous exploration and evaluation of the possible actions the reward is maximized and beneficial actions are learned.

Unsupervised learning describes the process of finding outliers, novelties and hidden patterns with the goal of learning what is typical, interesting or strange about the data.

Supervised learning could be described as the process of learning from past experience to predict the future. In contrast to unsupervised learning, the data that is usable in a supervised learning context already contains the information about the relationships and patterns, that are aimed to be learned. If this information is not already present in the data, it has to be added by manual labour.

For a collection of data points $\langle (x_i, y_i) \rangle_{i=1}^{N}$, where x_i is the ith data point, y_i the label for the ith data point and the number of data points N, which were collected in the past, a function $h : x \to y$ should be learned, which predicts the label y_{N+1} for a new data point x_{N+1}. A data point x_i is a vector in D dimensional vector space $\mathbb{R}^D$. Each component of a data point is a measurable property of the object that is to be analysed and is called a feature. This could be the number of rooms, and the latitude and longitude of a house in a house price prediction or the RGB values for each pixel in an image for an object detection etc. The label y_i can be continuous ($y_i \in \mathbb{R}$) or discrete (e.g. $y_i \in \{true, false\}$). In the former case, the supervised learning task is denoted as regression. In the latter case it is denoted as classification. In classification, instead of predicting the label explicitly, the conditional probability for the label y_{N+1} given a data point x_{N+1} can also be predicted, i.e. $h(x_{N+1}) = P(y_{N+1}|x_{N+1})$. The past experience $\langle (x_i, y_i) \rangle_{i=1}^{N}$ will also be denoted as training data, the

function $h : x \rightarrow y$ as model or hypothesis and a data point x_i as observation.

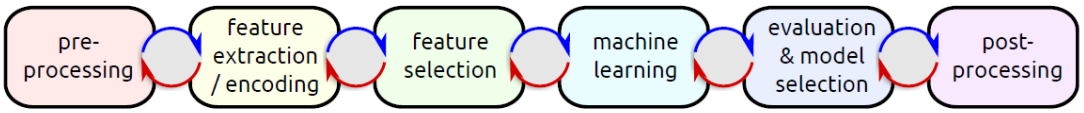

Figure 3: *The supervised machine learning development cycle as presented in the machine learning lecture at the University of Freiburg. [11]*

The development cycle for a supervised learning model as presented in the Machine Learning lecture at the University of Freiburg [11] can be seen in Figure 3.

The first step is the preprocessing. It involves obtaining and labelling of the training data (if no labels are present yet) and the handling of missing values and outliers in the dataset.

During the feature extraction and encoding, the data is prepared for the use in machine learning algorithms. Numeric or at least numerically encoded features can be directly used in machine learning, although it is not uncommon to transform the raw data into more meaningful features before the model training. If the data consist of text, which is the case in this work, features have to be extracted and encoded first. The preparation of raw text data is described in the sections 3.2 and 3.3.

The selection of a subset of the features is necessary, if the data contains features that are detrimental for the prediction performance or the runtime. The selection of useful features is highly dependent on the machine learning algorithm. Some algorithms' time or space complexity scale poorly with increasing number of features. Other algorithms do automatic feature selection and therefore scale well for high dimensional data. An example for such an algorithm is the Random Forest, which will be explained in section 3.4.

The actual machine learning step is comprised of the selection of the learning algorithm, the training of the model and the hyper-parameter tuning. Hyper-parameters are used to set-up the machine learning algorithm before the training. Hyper-parameters are algorithm dependant. For a tree based algorithm, the number of trees or the depth limitation of the trees are two of the algorithm specific hyper-parameters. Finding well performing hyper-parameter settings is a major task in the development of a machine learning model. Often times multiple algorithms are optimized with regard to their hyper-parameters and compared to find the best performing model.

To assess the previously taken steps and to compare the multiple trained models, the prediction results of the models for data not used in the model training is evaluated. There are two phases of evaluation.

The first is the evaluation done during the model development. Different algorithms are compared and the hyper-parameters for the algorithms are tuned

to achieve the best results. For this purpose a sample of the training data is taken before the model training to be used as validation data. A model is then fitted to the training data but applied to the validation data to evaluate the models performance. Iteratively, changes to the preprocessing, feature extraction / selection, machine learning algorithm and hyper-parameters are made and evaluated. In practice, more advanced methods of performance validation then a single validation sample are applied. For example, training and validating a model multiple times with changing training and validation samples (i.e. Cross-Validation) or algorithm specific validation like the out-of-bag error for Random Forests.

The second phase of evaluation is done after the model development. For a machine learning model to be applicable, it is a prerequisite that the model performs well on unseen data. This is called generalization performance. [1] It is common practice to draw a sample right at the beginning of the development, which should at no point be analysed or taken into consideration during the development process. This dataset is called the test set. It will only be used to evaluate the generalization performance of the trained models. Some of the standard metrics to evaluate classification models will be explained in section 3.7.

The last step is the post-processing. It involves deployment of the model, e.g. as a web API or integrated in an existing system, to take action based on the models predictions.

3.2. Preprocessing Natural Language Data for Machine Learning

Developing machine learning models, that take natural language text data as input, requires special preprocessing. In this section, the processing steps for the preparation of text data for machine learning applied during this work are explained.

Tokenization describes the process of splitting a sequence of characters into useful semantic units, called tokens. For example the sentence: "Listen up - there's no war that will end all wars." can be split into:

| Listen | up | - | there's | no | war | that | will | end | all | wars | . |

After the tokenization, different normalization steps are applied. Firstly, every token is changed to have the same case. For many applications it does not

[1] In some sources, methods like Cross-Validation are presented as measures for the generalization performance of the model. This can be misleading. If for example, the validation data is used too excessively during the development it is possible that the model over-fitted the validation data and might perform poorly for new data. Also it can disguise problems with the initial dataset. If the initial data sample is biased or does not contain all of the necessary patterns, validation data drawn from this initial datasets will not be a good measure for the generalization performance.

make a difference if the same word is written in lower or upper case. Secondly, certain tokens are filtered from the dataset. Depending on the task, which is aimed to be solved with the text data, there are tokens that are fruitful for the solution of the task and others that are not. In English for example, words like "a" or "the" appear very often in any kind of text, but they contribute very little to the distinctness of different texts. Those words are called stopwords. The most commonly removed stopwords are the words that appear most frequently in a language. Manually created or corpus-specific stopword lists can also be beneficial. Corpus-specific stopwords are the words appearing most in the set of documents at hand. Additionally, punctuations and special symbols can be removed.

Thirdly, stemming or lemmatization is applied. Both methods aim to reduce words to a common base form. Stemming denotes the process of reducing words to their word stem. For example the words "argue, argued, argues, arguing" are all reduced to "argu". Stemming does not necessarily produce proper words but can be an effective word normalization technique none the less. Lemmatisation denotes the process of grouping inflected forms of a term to their dictionary form, the lemma. For example all conjugated forms of a verb are grouped to their proper base form ("am, are, is" $\rightarrow$ "be").

3.3. Vectorization of Text Data with Tfidf

As machine learning algorithms rely on mathematical operations like the calculation of conditional probabilities, euclidean distances and gradients or data distributions, the data has to be numeric or at least numerically encoded.

When working with text data, it is therefore necessary to apply processing steps that convert each word into a number (or vector) and the text for each observation into a vector of numbers (or a matrix).

One of the basic variants of representing a document as a vector is the calculation of the term frequencies for the given document. Each document is firstly considered a bag-of-words, a collection of the occurring terms, which disregards grammar and ordering, while keeping multiplicity. Each document d is then represented as a sparse vector containing the number of times each term occurred in the document, i.e the term frequency tf. For each term $t \in T$, where T is the set of unique terms from all documents $d \in D$, each document d is represented as

$$d = (tf(t_0, d), ..., tf(t_m, d)), \text{ for } m = |T|.$$

A pitfall of this approach is, that words that occur often in the language at hand, would also appear most frequently in many texts. Then, the words with the highest frequency appear to be the most informative words, but would actually contribute the least to the separability of the vectors. One method to counteract this distorting effect was introduced in section 3.2, the stopwords.

Additionally, the term frequencies can be weighted with the inverse document frequency (idf) [12], which decreases the influence of words that appear in many documents.

For the number of documents N, the inverse document frequency of term t is defined as

$$idf(t) = \log(\frac{N}{\sum_{d:t\in d} 1}).$$

The $idf(t)$ decreases with increasing number of documents, that contain the term. If all of the documents contain the term t the $idf(t)$ is 0.
For a document d the term frequency inverse document frequency $tfidf$ of term t is given by

$$tfidf(t,d) = tf(t,d) \cdot idf(t).$$

With this, each document d in the data set is represented as a sparse vector of $tfidf$ values for each term $t \in T$

$$d = (tfidf(t_0,d), ..., tfidf(t_m,d)), \text{ for } m = |T|.$$

3.4. Random Forest Classifier

One of the machine learning algorithms, which produced the best results in this work, is the Random Forest. Random Forests belong to a type of machine learning methods that are called ensemble methods. Ensemble methods solve a learning task by generating many, possibly weak, prediction models and aggregating their results. The weak models for Random Forests are Classification and Regression Trees (CART), which additionally to the Random Forests have been proposed by Breiman [13, 14].

In this section the theory behind Random Forest Classification is presented. Firstly, the training of individual trees and the criterion for finding the best split at the nodes of the tree are outlined. Then, it is explained how the prediction result for a new data point is achieved. Lastly, bagging [15] and random feature subsets, two means of increasing the variance of the individual models are described. The information for this section is taken from Criminisi and Shotton [16].

For Random Forests, the training of the model translates to the construction of the individual classification trees. A tree is a data structure consisting of a collection of nodes and edges, which are organized in a hierarchical fashion. In case of a binary tree, the inner nodes have exactly two outgoing edges and all nodes except the root node have exactly one incoming edge. Figure 4 shows a binary classification tree for a four class classification task. Beginning at the root node, where all of the training data S_0 is accessible, the data is split into two subsets S_l and S_2 which are associated with their respective child nodes 1

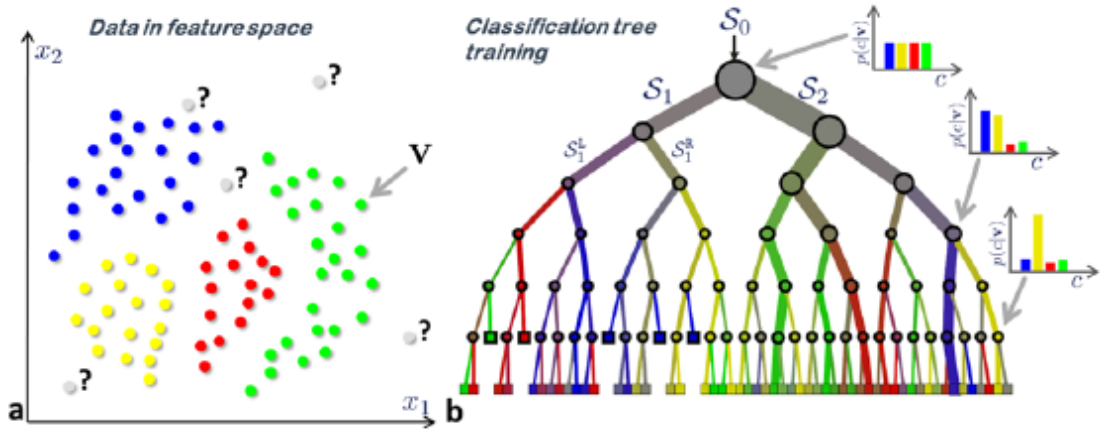

Figure 4: *Training data and tree training for a four class classification toy example as shown in Criminisi and Shotton [16]. a) Shows the training data in its 2 dimensional feature space. The four classes are shown in different colors. b) Shows a binary classification tree, which was trained on the data. S_i denotes the set of data points associated with the i-th node. The distribution of class labels is shown at the right. Traversing down the tree, the class distribution gets less disordered.*

and 2. [2] For each resulting subtree i the associated data S_i is split further until only a single data point is left at the node or a growth restricting condition (e.g. the maximum depth of the tree or the minimum number of data points necessary for a split) is met.

The classification tree is constructed by finding the best split of the data at each node depending on a splitting criterion. A split denotes a selected feature with a value for that feature, at which a decision is made, which separates the data into two subsets. The splitting criterion is such that with each split the disorder in the data regarding the class label distribution is decreased. The best split is then achieved, if it produced the most homogeneous subsets of all the possible splits. An often used splitting criterion is the Entropy and subsequently the Information Gain.

The entropy can be described as a measure of disorder of a system. The entropy H for a datasets S and probability $p(y)$ for class labels $y \in Y$ is defined as

$$H(S) = -\sum_{y \in Y} p(y) * log_2 p(y).$$

If we have, for example, a binary classification task and both labels are evenly present in the dataset, i.e. $p(y = 0) = p(y = 1) = 0.5$, the entropy is

[2] In an implementation of the Random Forest algorithm, the data is not actually split. Only the split features and split values are stored. Imagining the data being split at each node rather supports the intuition for the algorithm.

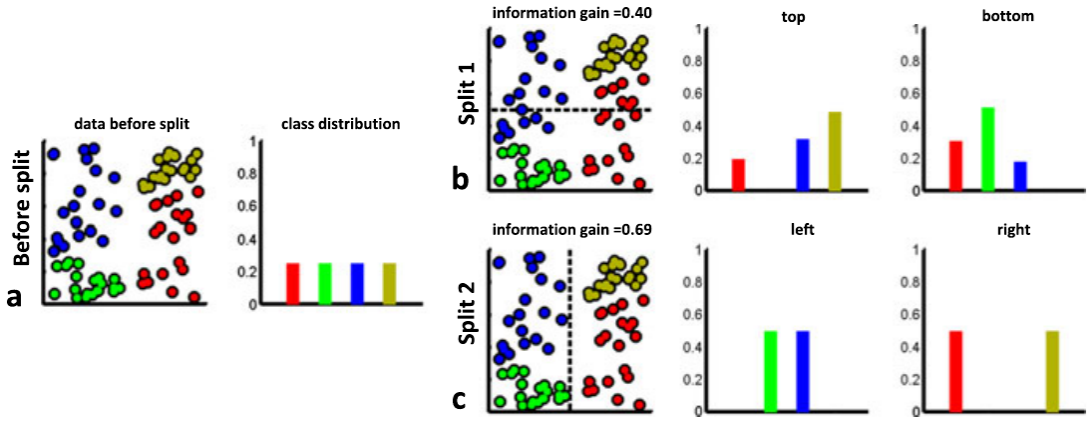

Figure 5: *Information gain for two splits on two different features of a four class classification toy example as shown in Criminisi and Shotton [16]. a) Data and class distribution before the split. b) Class distribution after a split on feature 2 - a horizontal split. c) Class distribution after a split on feature 1 - a vertical split. The vertical split produces less disordered class distributions and is therefore favourable.*

maximal. The further the distribution changes to favour one of the classes, the smaller the entropy gets.

The Information Gain is a measure for the change in entropy from one distribution to another. If a dataset S is split into two subsets S^L and S^R, the information gain is achieved by the split given by

$$I = H(S) - \sum_{i \in L,R} \frac{|S^i|}{|S|} H(S^i),$$

where $|\cdot|$ denotes the cardinality of sets. If the sum of weighted entropy for the subsets S^i is less then the entropy for the data S before the split the information gain is greater than 0 and the split is favourable. During the training of a tree, the best split at each node is found by maximizing the information gain with respect to a set of split features and their possible split values. Figure 5 illustrates the comparison of two different splits. It can easily be imagined, that a further split at the child nodes resulting from the vertical split shown in (c) would separate the classes almost perfectly.

The prediction of the label for a new data point is done by traversing down the tree following a path determined by the previously learned split feature, split value pairs at each node. The leaf node reached represents the result for the given data point. As there are generally multiple data points left in the leaf node, some form of aggregation needs to be applied. For classification trees a majority vote, where the label is returned that appears the most in the leaf node, can be used. Alternatively, the probabilities for the labels, which are derived from the class label distribution in the leaf node, can be returned. The

prediction for the Random Forest can then be derived by averaging over the results of the individual tree results. For the number of trees T, the probability for class label y for the data point x is then given by

$$p(y|x) = \frac{1}{T} \sum_{t=1}^{T} p_t(y|x),$$

where $p_t(y|x)$ denotes the probability of label y given data point x obtained by the t-th tree.

The individual decision trees are high-variance models. I.e. for slight changes in the data, a vastly different tree could be grown. On the other hand, when growing many trees on the same data little will be gained by aggregating their results. Following Breiman, the error of the aggregated model depends on the strength of the individual models and the degree to which the models' errors are uncorrelated. (c.f. [14]). By introducing randomness to the individual models during the training phase the variance of the models is increased and the errors become uncorrelated. Breiman proposed two means of introducing randomness to the forest. Firstly, bagging (bootstrap aggregation) [15], which denotes the training of the individual trees on randomly selected (with replacement) subsets of the training data. Secondly, random feature subsets, which denotes the restriction of the features used to find the best split at each node to only a small random subset of the total features. By introducing randomness to the individual trees, the errors get more uncorrelated, which results in a better generalization of the aggregated model.

Random Forest have three additional favourable properties. Firstly, the maximization of the information gain at each split leads to an automatic feature selection during the model training. Some features will contribute more to decreasing the entropy than others and will therefore be the split features over the features that contribute less. Secondly, using random feature subsets additionally makes the Random Forest training efficient for large numbers of features. Thirdly, with sufficiently large numbers of individual trees, the Random Forest furthermore provides a meaningful hierarchical order of the features based on the amount of which they contributed to the separability of the class labels. This order is called feature importance. The feature importance gives insight into the way in which the model produces its predictions and can also be used for the feature selection.

3.5. Stochastic Gradient Descent Training

The second machine learning algorithm utilised in this work is a support vector machine (SVM) with the modified huber loss function and stochastic gradient descent (SGD) optimization. In this section, this specific algorithm will not be explained in detail. Although this specific variant of SVM performed best,

other linear classifiers (e.g. SVM with hinge loss, Logistic Regression) with SGD training produced similar results. Hence, the basic concepts behind SGD and the intuition for SVMs will be presented. The information for this section is taken from [11, 17, 18].

A core element of the training phase of many supervised machine learning algorithms, (e.g. SVM, Logistic Regression, Neural Nets) is the iterative minimization of the error made by the model. This requires some kind of measure for the comparison of predicted and actual value for each observation. Let $\langle (x_i, y_i) \rangle_{i=1}^{N}$ be the training dataset with data points $x_i \in \mathbb{R}^D$ and actual labels y_i. A function $h : x \to y$ should now be learned, which can predict the label y_{N+1} for a unseen data point x_{N+1}. At each training step the error for the current model has to be calculated. This is done with a so called loss function. Let $l(h_\phi(x), y)$ be the loss function, which computes the aggregated deviation of the predicted labels $h_\phi(x)$ and the actual labels y. The expression h_ϕ denotes that the model is dependent on a set of model parameters ϕ. Improving the model then translates to finding the model parameters ϕ, which decrease the loss for all data points x, i.e.

$$argmin_\phi l(h_\phi(x), y).$$

A simple example is a linear regression with the mean squared error. The model is a linear function $h(x) = w^t x + b$, where w is the slope and b the y-intercept. The loss is given by

$$l(h_{w,b}(x), y) = \frac{1}{N} \sum_{i=1}^{N} (h(x_i) - y_i)^2.$$

To find the best fitting hyperplane the loss function has to be minimized with respect to w and b, i.e.

$$argmin_{w,b} \frac{1}{N} \sum_{i=1}^{N} (h(x_i) - y_i)^2.$$

Some of those optimization problems can be solved analytically, but in the cases where this is not possible or computationally too expensive iterative optimization strategies are applied.

A common algorithm for iterative optimization is gradient descent. The gradient of a function (denoted with ∇f) is the vector of partial derivatives of f, where the first component of the vector is the partial derivative with respect to the first variable. The second component is the partial derivative with respect to the second variable etc, i.e. $\nabla f = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ...)^T$. The intuition of the gradient is the same as for the first derivative of functions with only one variable. Its the slope of the tangent line to the graph of the function $f(x)$ at the point x. So the gradient of a function points in the direction of steepest ascent for a

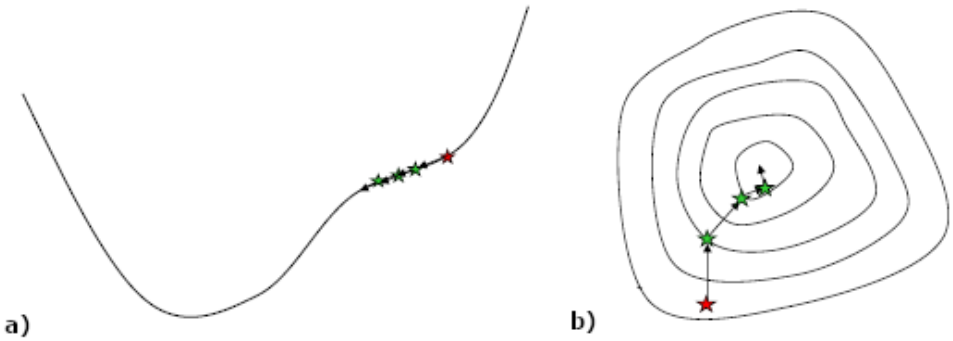

Figure 6: *Iterative minimization with gradient descent as shown in the Optimization lecture at the University of Freiburg [17]. Start point of the gradient descent in red. Each step of gradient descent in green. The arrows show the direction of change. a) Gradient descent for a single variable function. b) Gradient descent for a two variable function.*

multi-variate function. In each step of gradient descent the negative gradient of the loss function is calculated, which gives the direction of steepest descent. The new parameter values ϕ are obtained by making a step in that direction. For the direction of change d^k, the parameters ϕ^k and the step size τ^k at step k, one step of gradient descent is given by

$$d^k := -\nabla l(h_{\phi^k}(x), y)$$
$$\phi^{k+1} := \phi^k + \tau^k * d^k.$$

This step is repeated until the algorithm converges to a local minimum. [3] This will gradually find parameters ϕ with minimal loss. Figure 6 illustrates the iterative minimization with gradient descent.

A downside of gradient descent is, that it scales poorly with high number of data points N. In each step of gradient descent the gradient of the loss function has to be calculated, which accumulates the loss for the full training dataset. A more efficient optimization algorithm is stochastic gradient descent (SGD). Instead of calculating the next parameters ϕ^{k+1} based on the full datasets, SGD calculates the gradient descent step based on one randomly selected data point (or a small batch of data points) at a time and then repeats the step on the next data point (or batch). After randomly shuffling the training data one run of SGD is given by:

[3] This is a very rough description of gradient descent optimization. Aspects not covered here are the necessary conditions, the calculation of optimal step sizes and improvements of the basic approach.

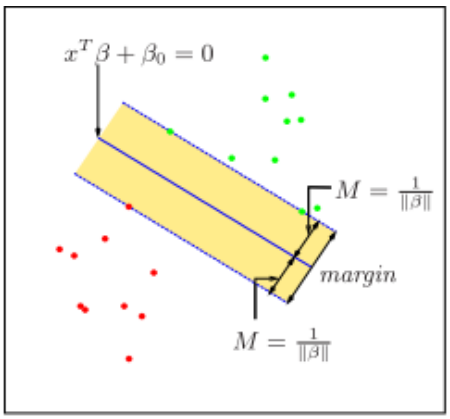

Figure 7: *Support Vector Classifier on two dimensional, two class classification toy example as shown in [19]. The two classes are drawn in red and green respectively. A separating decision function is given with $h_\beta(x) = x^T\beta + \beta_0 = 0$ which maximises the margin M.*

for i:= 1, ..., N {

$$d^k := -\nabla l(h_{\phi^k}(x_i), y_i)$$

$$\phi^{k+1} := \phi^k + \tau^k * d^k$$

}.

After one run of this loop every data point of the training data has been utilised to minimize the loss, which in some cases can be enough to be sufficiently close to the minimum. If not, this step is repeated.

One of the machine learning algorithms, which can be optimized with SGD, is the Support Vector Machine (SVM). The intuition for SVMs is as follows. Given a binary classification task with training data $\langle (x_i, y_i) \rangle_{i=1}^N$, with data points $x_i \in \mathbb{R}^D$ and labels $y_i \in \{-1, 1\}$. Find a hyperplane, which separates the two classes, while maximising the distance (called margin) between the closest data points for each class. Figure 7 illustrates such a hyperplane. The hyperplane is defined with $h_{\beta,\beta_0}(x) = x^T\beta + \beta_0 = 0$. The direction and length of β controls the orientation of the decision hyperplane. The hyperplane can be computed by solving an optimization problem. The margin should be maximised with the constraints that all data points are classified correctly and have a minimum distance of 1 to the hyperplane.

$$argmin_{\beta,\beta_0} \frac{1}{2}||\beta||^2,$$
$$\text{subject to } y_i h(x_i) >= 1, \text{ for all i = 1, ..., N .}$$

The vector β is orthogonal to the decision hyperplane. Minimizing β maximises the margin. Data points x_i, which are directly on the margin have

a distance of 1 to the decision hyperplane. These are the so called support vectors, which suffice to describe the hyperplane; hence the name support vector machine. An efficient way of solving this optimization problem for large datasets and sparse datasets is stochastic gradient descent. A reformulation of the problem with the hinge loss function, i.e.

$$l(h(x_i), y_i) = (1 - h(x_i)y_i)_+,$$

gives the objective function for a linear SVM [4],

$$argmin_\beta \frac{1}{2}||\beta||^2 + \frac{1}{N}\sum_{i=1}^{N}(1 - h(x_i)y_i)_+.$$

Other variants of SVM with different loss functions can also be optimized with gradient descent. The best performing variant for the web page classification is a SVM with modified huber loss function and SGD optimization. The modified huber loss is defined by

$$l(h(x), y) = \begin{cases} max(0, 1 - yh(x))^2 & \text{for } yh(x) >= -1 \\ -4yh(x) & \text{otherwise.} \end{cases}$$

3.6. Imbalanced Class Labels

When developing methods for the classification of web pages, it must be considered, that the different types of web pages are not necessarily represented equally often in the data. This is particularly true, if a binary classification is aimed for, where only one type of web page is aimed to be separated from all other types of web pages, as it is done in this thesis. It was observed, that only between 1% and 5% of the manually labelled web pages in the current dataset are scientist homepages. This imbalance of class labels in the data requires precaution, when developing and evaluating machine learning models. For example, a machine learning model could learn that simply predicting the majority class for all observations results in a nearly perfect accuracy score.

To avoid this, metrics like precision, recall, confusion matrix, f1 score and precision recall curve should be used over accuracy and receiver operating characteristic (ROC) curve. They allow for a better evaluation of the model performance regarding the minority class.

When developing a model, there are multiple basic methods to counteract the detrimental effect caused by imbalanced data. Firstly, machine learning algorithms can be used, which allow for the application of weights to each observation of the dataset. The minority class could be weighted more strongly,

[4] In practice, the linear SVM has to be optimized with the subgradient, as the hinge loss is not differentiable.

resulting in a bigger impact of errors of minority class observations on the trained model. Secondly, re-sampling techniques can be utilised to obtain a more balanced dataset. In an early experimental iteration of this work, undersampling, oversampling with Synthetic Minority Over-sampling Technique (SMOTE), and the use of the imbalanced dataset with class weights were compared. Undersampling denotes the reduction of the dataset size by randomly selecting only as many majority class observations as there are minority class observations. This is a very simple approach, which additionally speeds up the training process, but with the main disadvantage of loosing a large portion of data. Oversampling denotes the process of generating more samples of the minority class. SMOTE achieves this by considering the k-nearest neighbours for a minority class observation and generating a synthetic data point in their vicinity.

Although oversampling with SMOTE produced better results in some of the early experiments, the effect was not consistent and was accompanied by a major increase in model training time [5]. Training the models with imbalanced data and class weights did not increase the prediction performance significantly enough. Therefore, the simpler and faster undersampling method was applied in this work.

3.7. Metrics

In this section, some of the standard metrics for binary classification are explained. Let $h : x \to y$ be a binary classification model, i.e. $y \in \{0, 1\}$. Let y_i be the true label and $\hat{y}_i$ the label predicted by the model for observation i. The model made an error, if $y_i \neq \hat{y}_i$. There are two kinds of errors and two kinds of correct prediction outcomes.

For binary classification results, the following evaluation metrics can be calculated with respect to one of the classes (in this case class 1), denoted with "positive":

True Positive : $y = \hat{y} = 1$.
A positive observation has been correctly classified.

True Negative : $y = \hat{y} = 0$.
A negative observation has been correctly classified.

False Positive : $y = 0 \wedge \hat{y} = 1$.
A negative observation has been falsely classified as positive.

[5] Training time here includes the preprocessing of the data. Only at a later stage of the development, when the preprocessing steps and parameters were fixed, the model training was separated from the data preprocessing.

False Negative : $y = 1 \wedge \hat{y} = 0$.
A positive observation has been falsely classified as negative.

Let TP, TN, FP, FN be the count of the true positive, true negative, false positive and false negative prediction results, respectively.

Precision : $\frac{TP}{TP+FP}$
The proportion of true positive out of the total number of positive predictions. The precision, also called the positive predict value, gives a measure for the quality of the positive predictions made by the model.

Recall : $\frac{TP}{TP+FN}$
The proportion of true positive out of the total number of actually positive observations. The recall, also called the hit rate, measures how well the model is suited to predict the positive class.

F1 Score : $2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$
The F1 Score is the harmonic mean of precision and recall. Both, precision and recall contribute evenly to the F1 Score.

4. Web Page Data Source

The initial step in the development of a machine learning model is the acquisition of suitable datasets. In scientific work, often times well known and tested datasets for a given machine learning problem are used. They allow the comparison of the proposed methods and algorithms with existing work.

In this work, the application of the developed machine learning models is in focus. It was therefore important to use only actual and unaltered data. The source of the used data is Common Crawl [4]. Common Crawl is a non profit organization, which makes html data, meta data and text extractions from web pages publicly available. Common Crawl obtains the data by crawling the web on a monthly basis. In 2018, the size of the uncompressed content of a monthly crawl reached 215 to 270 TiB and consisted of 2.4 to 3.4 billion individual web pages [4]. The Common Crawl data is stored in the Amazon Web Services as part of their Public Datasets Program [20]. To access the data, the Amazon Web Services storage location of the searched URLs or all of the URLs from a domain have to be found. Common Crawl provides a server for this purpose, the Common Crawl Index Server. The index server can be queried via a web page or directly through an API. Alternatively, the 300 index files for a monthly crawl can be downloaded and processed locally. The size of a single uncompressed index file is around 5 GB.

The data for a web domain is stored distributively over many segments in the Amazon Web Services without any particular order. Therefore, even when downloading all of the web pages for a domain, many segments have to be accessed, as each segment contains only a few of the desired web pages. For example, the 127557 pages of the California Institute of Technology web domain found in 3 crawls of 2018 are distributed over 86026 segments. Fortunately, it is possible to utilise the offset and length information, which is part of the index server query result to only download the desired bytes from a segment for each of the web pages.

The data can be obtained in three formats. The WARC files [21] contain the raw crawl data. This includes meta data about the crawl process, the http header and the html data for the searched web page. The WAT files contain meta data about the records stored in the WARC format, whereas the WET data only contains the extracted plain text for a web page. For the experiments conducted in this work, WARC formatted data served as starting point.

5. Web Page Classification

A machine learning model for the identification of researcher homepages was developed with the acquired WARC data . This model estimates the probability for a web document to be a homepage. To achieve this, a multi view learning approach was used. Multi view learning consists of methods, which aim to improve the generalization performance of prediction models by utilising two or more disjoint feature sets regarding the same prediction object. This work was inspired by the co-training approach proposed by Gollapalli et al. [3].

Co-training belongs to the sub-field of semi-supervised learning, where unlabelled data is utilised to improve models previously trained on labelled data. In co-training, two or more models are trained separately on different views. In an iterative process, each of the models would predict samples of unseen data and high confidence predictions would be used to augment the initial two views. For example with the AddCross scheme, confident predictions of model 1 would be added to view 2 and vice versa. Each model would be retrained with its respective view in every iteration. Gollapalli et al. state the following intuition: If the model based on view 1 is able to predict at least one unlabelled observation confidently, which the model based on view 2 can not, co-training should improve the models performance over the iterations. In this work, an improvement of the models through co-training was not achieved. Nonetheless, a major increase in prediction performance could be achieved by a simple combination of two individual models trained on two disjoint feature sets. The first model was based on URL path features. The second model was based on text content and numeric features extracted from the html code for a page. The models for each view were trained and evaluated separately. The final, best performing classification model was obtained by evaluating different combinations of URL and page content models. The two models were combined by multiplying the prediction probabilities of the individual models. An adjustment of the decision threshold for the combined model improved the performance further. A significant increase in generalization capability was observed with the combined model in comparison to the individual prediction models.

In section 5.1 it is explained, how labelled training and testing data was obtained. In sections 5.2 and 5.3 the development of the two individual models is described. This involves the applied feature extraction and encoding, as well as the selected machine learning algorithm. The results for the individual and the combined models are presented in section 5.4. In the subsections of 5.5 additional findings regarding the feature extraction approach, the common prediction errors of the models and improvements of the approach are discussed.

5.1. Sampling

5.1.1 Training Data

Labelled training data for the web page classification task was obtained from three sources.

Firstly a small subset of the 4 Universities Dataset provided by the World Wide Knowledge Base project of the Carnegie Mellon University was used [22]. This dataset contains 8,282 web pages from the universities of Cornell, Texas, Washington and Wisconsin, which were manually classified into one of seven categories. Of those categories the URLs for "student" and "staff" were extracted and the corresponding WARC files contained in the nine monthly crawls of January to September of 2018 were downloaded via Common Crawl. As for all of the WARC files downloaded via Common Crawl the files were filtered, only accepting web pages that were reachable at the time of the crawl and contained at least a portion of English text. Of the 1778 web pages in the "student" and "staff" category only 52 remained after the filtering. This was due to the fact that the 4 Universities Dataset was raised in 1997 and most of the web pages do not exist any more. Considering the fast pace in which web content changes, it was assumed that the actuality of the training data has a major impact on the generalization capabilities of the developed machine learning models. Therefore, the loss of almost all the labelled data of the webkb dataset was accepted.

Secondly, data from the Computer Sciences Bibliography (dblp) [23] founded by the University of Trier was utilised. Dblp collects bibliography information on major computer science publications and provides free access to information about 4.4 million publications, published by more than 2.2 million authors. The raw dblp data can be downloaded in a single xml file with a compressed size of around 400 MB. The data set was searched for information about the homepage of the authors. From the resulting data, web pages, which are not part of a university domain, like social networks, code repositories or library domains were removed. With the obtained homepage URLs, Common Crawl was queried and after the filtering for reachability and language, 14473 WARC entries from the July, August, and September crawls of 2018 were downloaded.

Thirdly, 2130 homepages were labelled manually. On the basis of WARC data of the July, August, and September crawls of 2018 from the universities of Freiburg, Munich, Stanford and the media faculty of the MIT homepage URLs were extracted and the WARC data was downloaded from Common Crawl.

Additional filtering was applied to remove erroneous entries and entries without any web page text. The remaining 13670 examples for homepages were augmented with roughly the same amount of randomly selected data from the rest of the crawled data. The final training dataset consisted of 27197 entries.

5.1.2 Test Data

One of the main questions to answer, when developing machine learning models, is how the model performs for data, which has not been used during the model training. This is called generalization. To get an unbiased evaluation of the models prediction capabilities, it is common practice to take a sample from the data right at the beginning of the development and neither analyse nor utilise it in any form during the development. This, so called, test dataset is only used for the evaluation of the final model. In this work, a different approach was applied to generate the test dataset. The training data was obtained only from a few different sources. Therefore, it can not be assumed, that the training data represents the real world conditions concerning the web page classification extensively enough to test the generalization on a sample of that data. For this reason, the test dataset was raised separately. From the July, August, and September crawls of 2018, the WARC entries of six different university domains not used for the training dataset were downloaded. The universities are: California Institute of Technology, Princeton University, University of York (GB), University of Stuttgart, University of Hamburg and the University of Applied Sciences Upper Austria. From the WARC data of each university a random sample of 250 entries was drawn and manually labelled. The test data generated in this manner, consists of 1500 web pages, of which 86 are homepages. The test data is therefore highly imbalanced. This approach of obtaining the test data was chosen to get as close as possible to the actual application environment of the final classification model.

5.2. URL Based Classifier

In this section, the approach for the first of the two individual web page classification models is described. In the area of web page classification it has been shown that a prediction model, which is solely based on URL features, can produce good results [3, 7]. Often times, URL based models come to use, when classification speed is of importance; for example in a focused web crawler, which should quickly decide if a web page is relevant before it even fetches the page content. URLs contain both structural information by representing the location of the web page in the world wide web and also textual information. Even if no filtering, summation or encoding of the URL elements is done, the number of features, i.e. the number of unique words, is small (in comparison to web page content features), as most URLs are quite short. This allows for fast model training, evaluation and application. The major drawback is that some URLs simply do not contain enough information for the classification of the web pages.

	URL
(1)	https://ee.stanford.edu/research/control-and-optimization research, hyphenatedword
(2)	https://news.stanford.edu/2018/02/08/avoiding-blackouts-100-renewable-energy/ number, number, number, hyphenatedword
(3)	https://www.inf.uni-hamburg.de/en/inst/ab/hci/news/rse15.html nondict, nondict, nondict, nondict, news, alphanumeric
(4)	http://abi.inf.uni-tuebingen.de/People/krueger people, nondict
(5)	http://people.ucas.ac.cn/∼zhangxiaopeng?language=en tildenondict, querykeylanguage, queryvaluenondict

Table 1: *Example URLs with the extracted features in the line after each URL*

5.2.1 URL Surface Patterns

The extraction of URL based features, which can be used in the training of a machine learning model, was done on the basis of Gollapalli et al. [3]. They reason that in URL strings conventions can be observed, which are indicative of the web page type. When examining the example URLs in Table 1 it can be seen, that certain words like "research" or "news" in (1) and (3) or numeric patterns in (2) make it highly unlikely, that those URLs lead to a scientist homepage. Example (4) on the other hand, is more likely to be a homepage as it contains the keyword "people" followed by a person's name.

Gollapalli et al. [3] propose the encoding of the URL components with surface patterns such as the presence of hyphenated or underscored words, alphanumeric, numeric or long words (i.e. words with more than 30 characters), or the symbol ∼. By doing this, they aim to filter out certain kind of web pages like course pages, announcements, calendars and auto-generated content. They further propose the use of a term dictionary, i.e. the collection of unigrams and bigrams, which appeared more than three times in the URLs of the training data. Terms, which do not exist in the term dictionary and are also not found in WordNet [24], a lexical database for English, are also encoded. Additionally they add a special case, which is designed to capture scientist homepage URLs often found in computer science departments. Those URLs contain the researchers name prefixed by the tilde symbol. For example, in "http://www.robotics.stanford.edu/∼ang" the term containing the tilde symbol followed by an abbreviation for Andrew Ng would be encoded as "tildenondict".

In this work, the following steps were applied. Firstly, the domain and subdomain components of the URLs were removed, as they would introduce a strong bias with regard to the universities and the faculties present in the training data. Secondly, the surface patterns were captured. In contrast to Gollapalli et al. [3], URL path and URL query components were encoded

separately. The path components were captured as described above. For the query components, only the presence of alphanumeric, numeric or non dictionary terms was captured and an identification prefix for the query key and the query value was added. This was done to account for the observation, that the query components often modify the contents of the page but are not used to load new pages. For example, the query string "?language=en" denotes, that the page content should be displayed in English, whereas an URL path ending with "/language/en" could also indicate a new page, which has something to do with the English language. Without the proposed distinction, the features for both URLs would be the same. (I.e. "... , language, nondict") With the proposed distinction the URL containing the query is represented as "... , querykeylanguage, queryvaluenondict".

Also, the constraints on the term dictionary were loosened up. Instead of filtering uni- and bigrams by their number of appearance, only unigrams with more than three occurrences were added to the term dictionary. This was done with the intention not to loose any information about the sequence of the URL terms at this point. Allowing for n-grams was instead done after the preprocessing.

Following the feature engineering, preprocessing steps were applied to the URL features, which are common in natural language processing. The URLs were tokenized. It was made sure that all tokens are lower case and do not contain any white spaces. Additionally, stemming was applied.

Lastly, $tfidf$ vectorization was applied to uni- and bigrams built from the URL features. Corpus-specific stopwords were removed, by omitting all features, which appear in more than half of the URLs. The hyper-parameters of the feature engineering, the preprocessing and the vectorization were optimized with a grid search [25].

By extracting URL features in this manner, a few beneficial properties can be achieved. The number of features is reduced substantially. The URLs in the training dataset contained 28307 unique terms. After the feature engineering and preprocessing 2595 feature terms remained, which resulted in a final 8386 features after the $tfidf$ vectorization. So the training data consisted of 27197 URLs each transformed into a 8386 dimensional sparse vector of weighted term frequencies.

5.2.2 Machine Learning Method - Random Forest Classifier

For the URL based classification, Random Forest and linear models with Stochastic Gradient Descent learning were compared. Scikit-learns Random Forest Classifier [26] and SGD Classifier [27] were used. The best classification performance on the test dataset could be achieved with the Random Forest Classifier. Mostly the default parameters of the scikit-learn implementation were used. Only the number of estimators was set to 1000. Different hyper-parameter

settings were tested via grid search [25] without improving the prediction performance.

5.3. Page Content Based Classifier

The second homepage classification model uses the web page text and numeric features, which describe structural properties of the web page. It was observed that many researcher homepages are comprised of a certain set of content elements, like the contact information, the researchers academic background and current activities, a listing of publications, the persons affiliations or an image of the researcher. The variety in the composition of those elements is very high, as some homepages only consist of the researchers image and the contact information, whereas others contain any number of those elements or have the information scattered over multiple pages. Despite this difficulty, it is assumed that basic text and numeric features constructed to capture those content elements, are sufficient for the classification.

5.3.1 Page Content and Structural Features

To obtain the features for each web page the text content of the title tag, the h1 tag, and the remaining visible text of each web page were extracted from the RAW-html. The title and h1 texts were tokenized, stopwords and punctuations were removed and prefixed with an identifier for their respective html tag. It is assumed that the information about the text source can be quite beneficial for the classification model. The remaining visible text of each web page was filtered by the length of the text segments. Text segments that contain less than 7 words were disregarded. Initially this was meant to reduce the noise in the text data by removing web page components, such as navigation. Through experimentation, it was found that removing longer segments (length > 3) produces better prediction results. Stopwords, punctuations and terms, which could not be found in Wordnet [24], were also removed. Finally, the remaining text for each page was tokenized and lemmatized. The processed title, h1 and page texts were concatenated, uni-, bi-, and trigrams were constructed and vectorized via $tfidf$. The number of features produced by $tfidf$ was limited to 20000. Of the 110771 unique terms (before any preprocessing) 20000 features were generated. Experiments with multiple parameters settings for the preprocessing were conducted. The preprocessing procedure described above produced the best prediction results on the validation dataset.

In addition to the text based features, six simple numeric features were extracted from the html data for each web page. Those are the number of tables on the page, the number of links referencing an external web location, the number of links referencing an internal location, the number of images on the page, the number of person names in the html title tag and the number of

person names in the h1 tag. When counting the number of images, size and ratio limitations were applied to disregard irrelevant image types, like banners and icons. The purpose of the number of tables, links and images is to rule out certain non homepage types. A web page, for example, that has many tables or images is unlikely to be a homepage. The extraction of the person names was done with a pre-trained named entity recognition model ("en_core_web_lg") from the spacy library [28]. It was observed that the title tag on homepages often times contains the researcher name. The number of person names is therefore assumed to be a strong indicator for homepages. Still, it has to be noted that the named entity recognition in itself is a complex task. The person name extraction is certainly not flawless and introduces an error to the training data. That being said, the benefit of adding the numerical features could be verified through experimentation. Finally, numeric features were normalized with the scikit-learn MinMaxScaler [29] and concatenated with the vectorized text features. Each of the 27197 web pages is then represented by a sparse vector of length 20006.

5.3.2 Machine Learning Method - Support Vector Machine

For the page content based web page classification, Random Forest and linear models with Stochastic Gradient Descent learning were compared. The best classification performance on the test dataset could be achieved with a Support Vector Machine (SVM) using the modified huber loss function [27]. Multiple hyper-parameter settings were tested via grid search, but the default parameters performed the best.

5.4. Results

In this section, the results for the URL based, the page content based, and the combined model for the classification of homepages are presented (Table 2). For the combined model results, the estimated probabilities of the two individual models were multiplied, i.e.

$$P_{combined}(y|x) = P_{url}(y|x) * P_{page}(y|x).$$

The prediction performance is high for the individual as well as the combined model for the validation data. The URL feature based model has a F1 Score of 92%, the page content feature based model 96% and the combined model 97%.

On the test data, the individual models show low precision with 18% and 12% for the URL based and the page content based model, respectively. A significant increase in prediction performance could be achieved by combining the two individual classification models. The precision for the prediction of homepages was increased to 68% for the combined model. The recall of the

		Validation Data			Test Data		
Model	Label	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Page Content	0	0.97	0.94	0.96	1	0.58	0.73
	1	0.94	0.97	0.96	0.12	0.97	0.22
Url	0	0.91	0.91	0.91	0.99	0.77	0.86
	1	0.92	0.92	0.92	0.18	0.84	0.29
Combined	0	0.95	1	0.97	0.98	0.98	0.98
	1	1	0.94	0.97	0.68	0.69	0.68

Table 2: *Results of the Web Page classification for the URL feature based, page content feature based and the combined model for the validation and the test dataset. The combined model is the product of the prediction probabilities of the two individual models. The decision threshold for the combined model was increased to 0.62. Label 1 denotes homepages, label 0 non homepages.*

prediction of homepages decreased from 97% and 84% to 69%, whereas the F1 score increased from 22% and 29% for the page content and URL based model to 68% for the combined model. The test dataset is highly imbalanced as it contains 1414 non homepages and 86 homepages. By combining the two individual models, the number of false positives dropped from 595 and 331 to 28 and the number of false negatives increased from 3 and 14 to 27. The increase in precision therefore has a significantly bigger impact on the quality of the resulting dataset then the decrease in recall.

5.5. Evaluation and Discussion

In this section, additional findings are discussed. In section 5.5.1 and 5.5.2 the feature extraction of the two individual models is evaluated. The most frequent terms and the most important features are assessed. In section 5.5.3 common prediction errors produced by the models are analysed and potential improvements proposed. In section 5.5.4 the improvements achieved by combining the two individual models are examined.

5.5.1 URL Model Features: Evaluation

In this section, the relationship between term frequency and separability of web page types for the URL based model is examined. Firstly, it is checked, if the frequent terms seem fruitful for the separation of web pages. Secondly the features, which contributed most to the predictions with the Random Forest, are examined.

Figure 8 shows the fifteen most frequent terms in the preprocessed training data for the URL based model, separated by homepages and other web pages.

The most frequent term for the homepages is "tildenondict", which corresponds with Gollapalli et al.'s [3] observation and is not surprising as most of the training data comes from the Computer Science Bibliography [23]. The second most common term is "nondict". Although a lot of different terms are summarized by this feature it appeared almost two times more often in homepage URLs than in other web pages.

This corresponds with the observation, that many homepage URLs contain the name of the scientist or at least an abbreviation of that name. It is assumed that other terms, which would also not be in the term dictionary but appear in non homepages, often contain underscores, hyphens or numbers. This could be the case in auto-generated URLs or web locations where many similar pages can be found, like library or news pages. The frequency of keywords like "peopl", "profil" and "staff" for homepages and "index" and "event" for other web pages support the assumption of separability of web pages by URL naming conventions.

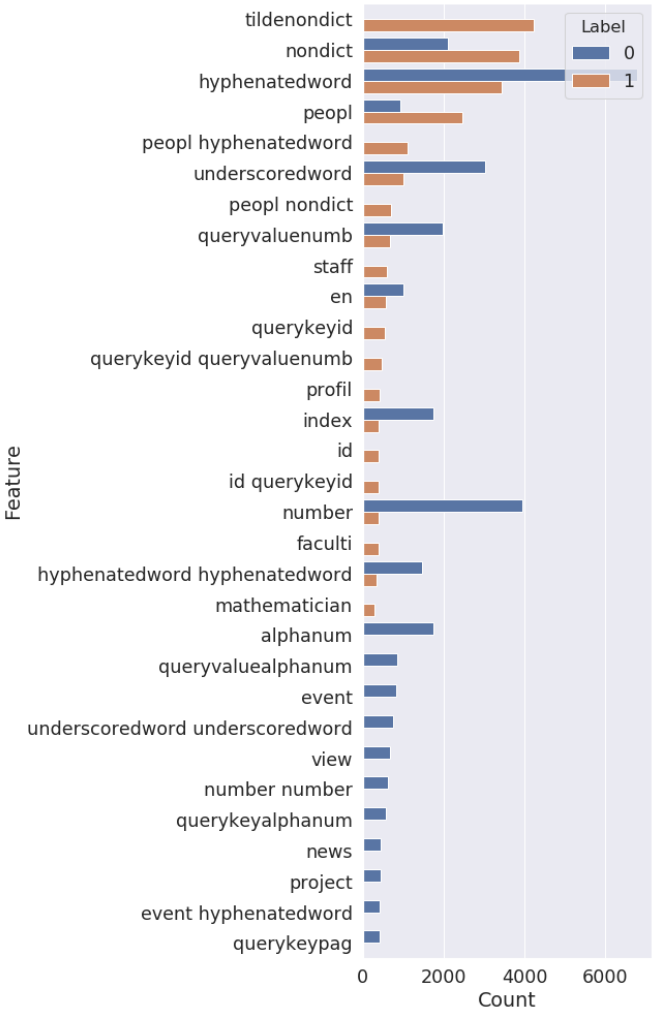

Figure 8: *Top 10 term frequencies for preprocessed URL features by label. Most frequent features in scientist homepages are in orange. Most frequent features in non-homepages in blue.*

The frequency of other keywords, such as "en", "id" and "view" do not seem to be very fruitful for the web page classification as they could appear in either context. The surface patterns, hyphenated words, underscored words, and numbers as query value appeared often for both web page types, but are more prominent in non homepage URLs. More informative seems to be the presence of numbers and alphanumeric terms in the path or alphanumeric terms in the query. They appear about 10 times more often in non homepage URLs than in homepage URLs. It was also found that certain bigrams seemed to be valuable for the web page classification task. Those are the keyword "people" followed by either a

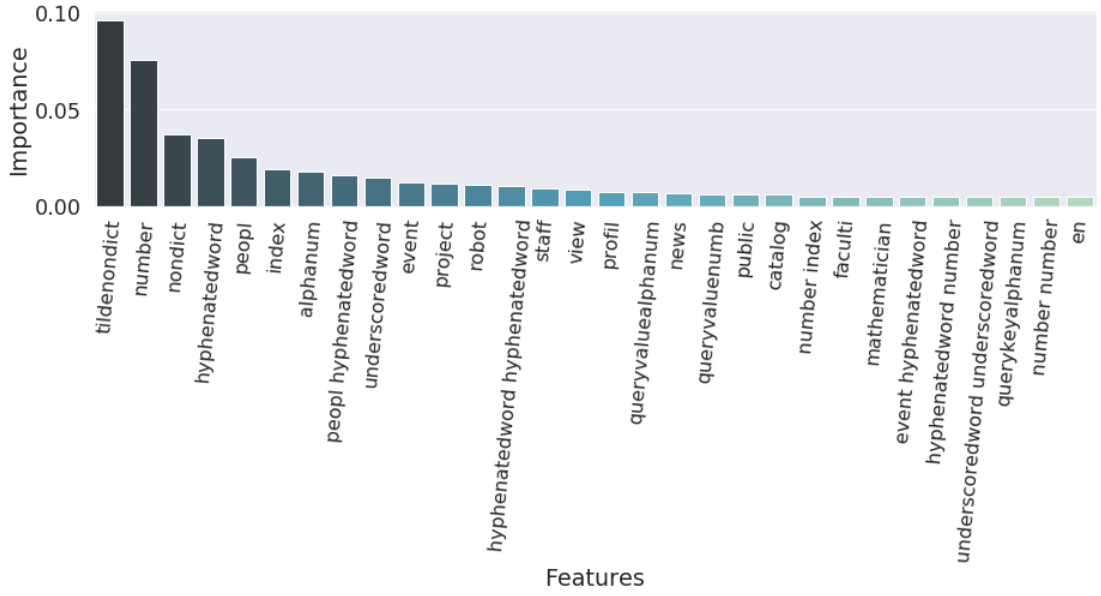

Figure 9: *Feature importance as produced by the Random Forest Classifier for the URL based web page classification model. The thirty most important features are shown.*

hyphenated or non dictionary word in homepage URLs and the succession of two numbers, two hyphenated words or two underscored words in the URL path of non homepages.

An examination of the feature importance measure produced by the Random Forest Classifier verified that the trained model picked up the intended relationship of web page type and URL term frequency. The feature importance can be seen in Figure 9. 20 of the 23 most frequent features are in the top 31 of important features. The remaining eleven important features contain keywords, such as "project", "catalog", "news" and "robot" (The robot.txt in the root directory of a website contains instructions for web crawlers.) and bigrams like "event" followed by a hyphenated word or a number followed by the term "index". The first 30 of the important features explain almost 50% of the importance measure, though almost 2000 of the 8336 features are needed to explain 95% of the importance measure.

5.5.2 Page Content Features: Evaluation

In this section, the relationship between the term frequency and the separability of web page types for the model based on page content is examined analogous to the previous section. Support Vector Machines, however, do not inherently provide a measure for the feature importance. Therefore the feature importance for the page content features is discussed based on the mutual information metric. Mutual information is, besides other metrics, used to identify important features before the model training, e.g. for feature selection.

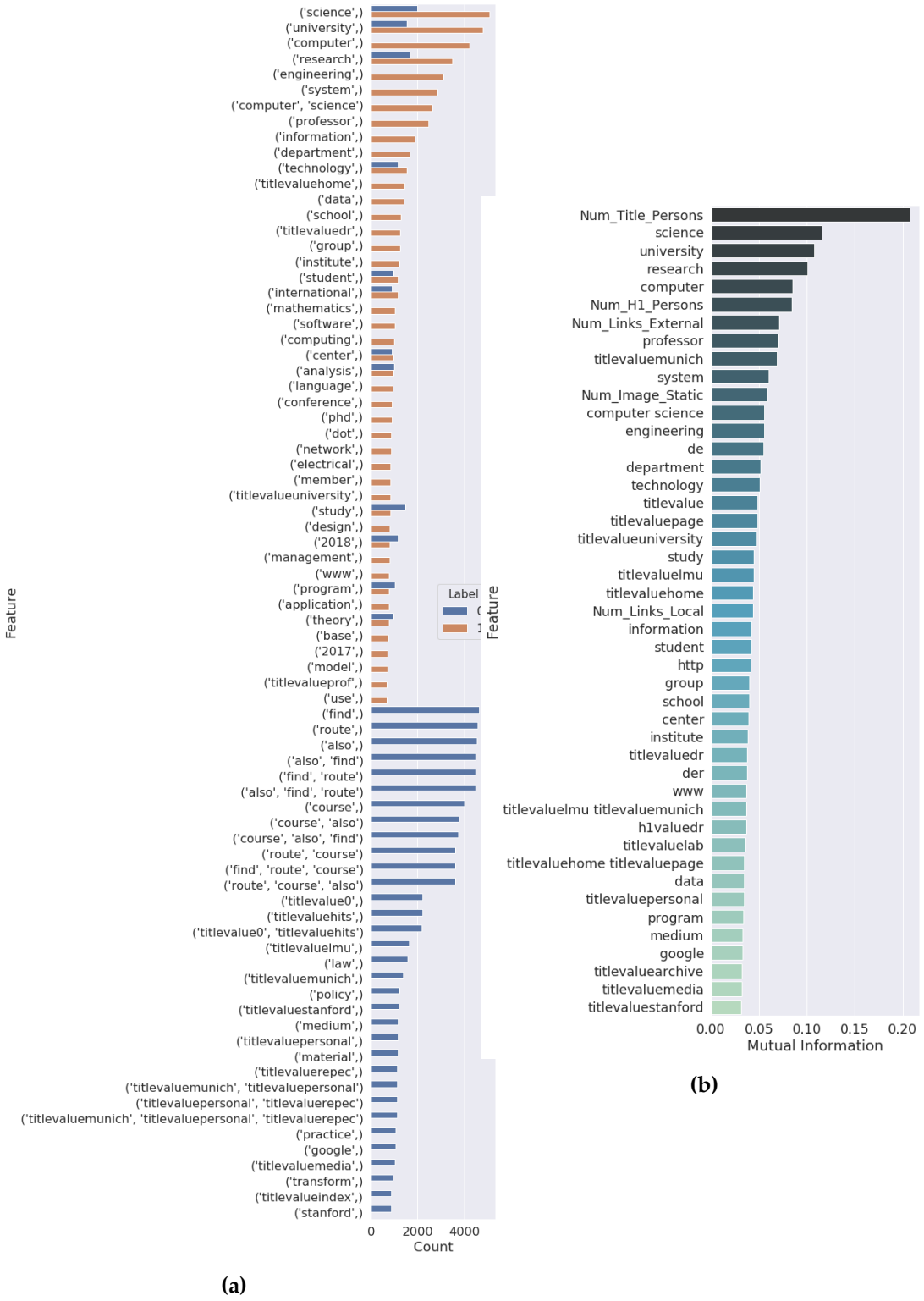

Figure 10: *a) The 45 most frequent, preprocessed page content features by label. Most frequent features in homepages in orange; most frequent features in non-homepages in blue. b) The 45 most important features based on mutual information.*

Examining the term frequencies for the web page content features, shown in Figure 10a, two problematic effects of the proposed data extraction and preprocessing were observed. Firstly, a bias towards university domains and the computer science field was seen. Terms like "university", "computer science", "research", and "engineering" appeared within the 10 most frequent terms for homepages. The bias towards university domains was expected and unproblematic, as the web page classification task was constrained to only university web pages in the first place. But the bias towards the computer science field was the result of the selection of data sources for the training data. The majority of training data was extracted utilising the Computer Science Bibliography (dblp). Secondly, most of the frequent terms seem to be not fruitful for the web page classification. For example, "system", "information", "data", and "international" for homepages, and "medium", "titlevaluepersonal", and "find" for non homepages.

The inspection of the feature importance, shown in Figure 10b, also revealed a bias towards university domains and the computer science field. Out of the first 15 important features six were related to those areas. Furthermore, there seems to be an additional bias towards the University of Munich. Although only a small subset of the training data was obtained from the University of Munich domain, the feature importance measure indicated that the presence of the terms "munich" or "lmu" in the title was important for separating the two web page types. This is clearly not the case. Also, multiple other terms ("system", "www", "de", "http") were found in the important features, which are very unlikely to actually impact the separability of web page types.

Besides those problematic aspects, there were also indicators for the usability of the simple relationship of term frequency and web page type. There was a significant separation of terms, which often appeared in homepages and terms, which often appeared in other web pages. Also key terms like "professor", "titlevaluehome", and academic titles in the title or h1 tag strongly indicate a homepage, whereas uni-, bi- and trigrams containing the term "course" or the terms "material", "titlevalueindex", and "titlevaluelab" indicate other web page types. The simple numeric features seem to work as intended. The presence of a person's name in the title or h1 tag were ranked first and sixth, the number of external and internal links seventh and 23rd, and the number of images 11th in the most important features. The first 40 most important features explained 9% of the feature importance measure. To explain 95% of the feature importance measure, 14388 features were needed.

5.5.3 Improvements of the Web Page Classification Models

In this section, common errors of the models are examined and possible improvements for the data acquisition and the feature extraction for the individual models are proposed.

	URL	$P_{url}(y=1\|x)$	$P_{page}(y=1\|x)$	Error Type
(1)	http://www-users.cs.york.ac.uk/~susan/sf/dani/PS_019.htm tildenondict, nondict, nondict, underscoredword	.73	.14	false positive
(2)	https://www.ifm.uni-hamburg.de/en/datenschutz.html en, nondict	1	.79	
(3)	https://www.york.ac.uk/economics/our-people/staff-profiles/john-hutton/ economics, hyphenatedword, hyphenatedword, hyphenatedword	.37	.88	false negative
(4)	http://carvermead.caltech.edu/research.html research	.58	.74	

Table 3: *Selection of URLs of falsely classified web pages. Features are shown in the second row for each URL. $P_{url}(y=1|x)$ is the prediction probability produced by the URL based model for a web page being a homepage given its feature vector x. (1) was falsely classified by the URL model only, (2), (3), (4) were falsely classified by the combined model.*

During the development of the models, the hyper-parameter tuning was done using either a validation subset of the training data or cross validation as part of the grid search [25]. The prediction performance of the models was consistently high for the validation data. In addition, no overfitting, i.e. a decrease in training error without a decrease in validation error, was observed. This indicates that the engineered features and chosen machine learning methods are suitable for the attempted binary web page classification.

The generalization capabilities, on the other hand, were sub-par. The most obvious reason is that the training dataset does not contain enough of the relationships and patterns necessary. Only a small subset of the training data was manually selected and labelled and the majority of the training data was obtained from the Computer Science Bibliography [23]. This introduced a bias towards the computer science field to the training dataset. A common measure to improve the generalization is the use of more training data. This would increase the amount of relevant and not yet seen patterns in the data and diminish the effect of the bias. Ideally, gold standard data sets for the web page classification task would be utilized, or a substantial amount of data from many different universities, cultures and faculties would be manually selected and labelled.

Another measure to improve the generalization is the addition of new features, which capture new patterns in the data, which are relevant for the web page classification. The evaluation of the errors produced by the URL, the page based, and the combined model gives insight about how the addition of new features could increase the precision. Table 3 shows example errors made by the URL based model. A frequently occurring error was the classification of a non homepage as homepage, if the tildenondict feature was present. In the majority of those cases, the tildenondict element was followed by one or more URL path or query elements. Homepage URLs however often end with the tildenondict feature. URL (1) in Table 3 is such a case, where the URL

Figure 11: *Common error of the page content based model. The title of the page contains a person name.*

based model misclassified a book review web page as homepage. Prof. Susan Stepneys' actual homepage is at "https://www-users.cs.york.ac.uk/~susan. It is assumed that the lack of sequence information, beside the use of bigrams, leads to this kind of error. New features representing the beginning and end element of the URL could reduce those errors.

An example for another error, which resulted from the handling of different languages during the feature engineering, is shown in Table 3 (2). In this work, only web pages with at least one English portion of text were used. However, data from multiple German universities were utilised. With the proposed feature engineering, any German word appearing in the URL is encoded as nondict, disregarding its actual meaning. This introduces noise to the training data in two ways. Firstly, the keyword "en", which was often present in URL routes for the English variant of a web page, was falsely learned by the model as being relevant for the classification. Secondly, the nondict feature, which was designed to capture named entities and abbreviations, gets distorted by the addition of non English words, which actually have a proper meaning. Adding the keyword "en" to the stopwords, translating all terms to English before the feature engineering or only using data from English universities could reduce this kind of error.

Table 3 (3) shows a similar error type. The meaning of the URL route terms was disregarded by the encoding as hyphenated words. This could be circumvented by encoding the individual words of the hyphenated terms. In this case keywords like "people", "staff", "profiles" followed by a nondict would strongly indicate a person related web page.

An examples for the limit of the URL based model is shown in Table 3 (4). The URL leads to the homepage of Prof. Carver Mead but the URL route

does not contain relevant information. It could be argued that the subdomain elements of the URL should be included as feature, as in this case the Professors name is present as subdomain element. But it was found that in many cases the subdomains hold information about the faculty, the scientific area or the laboratory (which sometimes was denoted by a person name). The former is assumed not to improve the separability as only web pages from university domains were used. The latter two might even introduce more noise. The bias towards the computer science field could be reinforced by adding the scientific area. Adding non person entities, which have person names would add a misleading relationship to the data.

When examining common errors of the page based model, it is concluded that a more elaborate feature engineering for the text content of web pages is necessary to increase the precision of the classification. The simple features utilised in this work perform pretty well for typical homepages (high recall), but because of their simplicity only allow for a rough classification (very low precision). The page based model misclassified non-homepages as homepages, which only roughly resembled a homepage. Index pages for scientific authors, for example, were confused as homepages. Fig. 11 shows such an index page. It is assumed that the small amount of text content paired with the presence of a person name in the title tag leads to the misclassification. The presence of a person name in the title is a strong indicator for a homepage only if combined with homepage specific web page attributes.

An improvement could be achieved by substantially expanding the stopwords used in the preprocessing of the page content features. As seen in section 5.5.2, many unigrams appeared in the important features although it is unlikely that they actually contribute to the separability of the two web page types. A more elaborate method could be topic modelling applied to the page content features. Gollapalli et al. [2] propose Latent Dirichlet Allocation (LDA) as part of the feature engineering. In LDA, a document can be regarded as a collection of topics and the presence of each term in the document contributes to one or the other topic. The topic distribution of a web page would then be used as features for the classifier.

5.5.4 Combined Model: Prediction Probabilities

The prediction results of the combined model were achieved by multiplying the prediction probabilities of the URL based and the page content based model. The prediction probability can be interpreted as a measure for the confidence with which a prediction was made.[6] By multiplying the prediction probabilities

[6]For binary classification the prediction probabilities with respect to class 1 are equal to the converse prediction probabilities with respect to class 0. Here, only the prediction probabilities with respect to the classification of homepages are examined. Prediction probability values close to 0 or 1 a therefore considered confident, values close to 0.5 are considered uncertain.

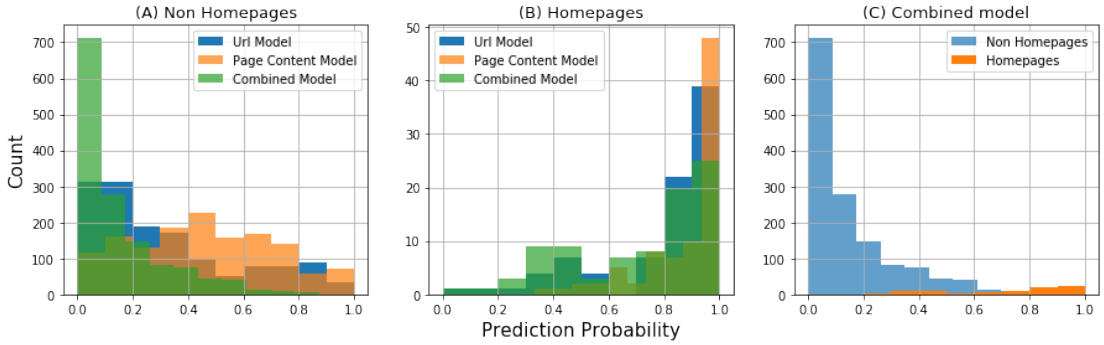

Figure 12: *The prediction probabilities of the web page classification of the test dataset. The test dataset is highly imbalanced. The decision threshold was adjusted to 0.62. Figure (A) shows the prediction probabilities for non homepages. Figure (B) shows the prediction probabilities for homepages. Figure (C) shows the prediction probabilities for the combined model coloured by predicted label.*

only the homepage predictions, which are confidently made in both models, stay confident predictions in the combined model. Any insecurity of either of the models further reduces the confidence value for the combined model. The classification of the web page type was done using a threshold value for the prediction probabilities. For a binary classification, the decision threshold defaults to 0.5, but was set to 0.62 for the combined model, which increased the prediction performance. So, a web page was classified as homepage, if the prediction probability was greater than or equal to 0.62 and classified as non homepage if the prediction probability was less than 0.62.

Figure 12 shows the prediction probability values produced by the URL based, the page content based, and the combined model for the test dataset. Figure (A) shows the prediction probability distribution for known non homepages. Prediction values greater than 0.62 are therefore prediction errors. The URL based model confidently predicted (prediction probability $<= 0.2$) around 44% of web pages as non homepages. The page content based model only predicted 20% of non-homepages confidently. Its distribution of prediction probabilities shows that the majority of predictions are around the 0.5 value and can therefore be considered uncertain. The distribution for the combined model shows the previously stated decrease in classification error, as well as an increase in the number of confident non homepage predictions. Here 74% of predictions were done with a prediction probability of $<= 0.2$.

Figure (B) shows the prediction probabilities for known homepages. The URL based and the page content based model show similar prediction probability distributions. Of the predictions, 71% and 75% are made with a confidence of greater than 80% by the URL based model and the page content based model, respectively. The distribution for the combined model shows a decrease in confident homepage predictions and the previously stated increase in classification

error. Hence, only 52% of homepage predictions are done with a confidence value of greater than or equal to 80%. Because the test dataset was highly imbalanced and contains only 86 homepages, the shown distributions should only be interpreted as tendencies.

Figure (C) shows the prediction probabilities for the combined model coloured by the known web page type. The decrease in classification error achieved by the adjustment of the decision threshold to 0.62 can be observed. Also it can be seen that, although there is a significant amount of uncertainty left in the predictions, there are no confidently made prediction errors.

6. Homepage Owner Identification

The step following the identification of researcher homepages is the extraction of information from the identified web documents. Exemplary, the researcher's name was extracted. To identify the homepage owner, firstly, all person names appearing on the web page were extracted. Secondly, a classifier was trained that predicts the probability of a name belonging to the homepage owner. The first subtask is a special case of a Named Entity Recognition (NER) problem, the second is formulated as binary classification task. The machine learning algorithm used for the second subtask was a Random Forest Classifier with mostly default hyperparameters. Only the number of estimators was set to 30 and the minimum number of samples required for a node to be a leaf node was set to 15. Those values were found through experimentation.

For the first subtask, a publicly available NER model was used. To find the best performing NER model for the person name extraction from web page text segments, a comparison was conducted.

The results of that comparison are presented in section 6.1. The sampling and preprocessing of the data for the homepage owner identification is described in section 6.2. The results are presented in section 6.3, which is followed by the evaluation of the developed model in section 6.4.

6.1. Named Entity Recognition Model Comparison

To get a rough idea which of the publicly available NER models is likely to perform best in the person name extraction, a simple comparison was done for three NER models of the spacy library [30] (en_core_web_sm, en_core_web_md, en_core_web_lg), the NER model of the nltk library [31] and the Stanford Named Entity Recognizer [32]. To generate testing data, lists of frequent first and last names were obtained from the US Census Bureau [33]. 750 male and female first names with varying length and number of abbreviations were randomly generated and suffixed with a randomly selected surname. This produces names like "Aretha Randee Bagaoisan", "Ora G. J. Mattock" and "X. Laine".

In the first step, it was evaluated how well those randomly generated names by themselves are recognized by the NER models. Also basic means of combining the results of two models were evaluated. Those are the union and intersection of results of two models. It is expected that the union of results has a higher recall and the intersection a higher precision than the individual models. The worst performing individual model came from the nltk library achieving an accuracy of 42% and an F1 score of 59%. The Stanford Named Entity Recognizer performed best in extracting randomly generated names with an accuracy of 96% and an F1 score of 98%.

Based on the assumption that the person name that appears in the title

	Test Sentence	True Name
(1)	Homepage of C. Ignacio Baierl	C. Ignacio Baierl
(2)	Prof. Dr. Eric Aulder | The Insight Centre for Data Analytics	Eric Aulder
(3)	Brigida J. C. Heilig | TAMU	Brigida J. C. Heilig
(4)	Overview | Staff | nbrooks	-
(5)	The Gordon Lab	-

Table 4: *Example sentences for the NER model comparison. Academic titles should be disregarded. (4) and (5) are negative examples. No person name should be found here.*

or header tags of a homepage is most likely the homepage owners name, the second comparison was done constructing test sentences similar to homepage titles and headings. Thirty-one test sentences were constructed and augmented with 50 random person names each. Example test sentences are shown in Table 4. Table 5 shows the performance metrics for a selection of models. The Stanford NE Recognizer performed best. The Stanford model ranks first in F1 score (91%) followed by the union of the Stanford and medium spacy model (F1 : 86%).

	Stanford	stanford ∪ spacy_md	spacy_md	spacy_lg	stanford ∩ nltk	spacy_sm	nltk
F1 Score	0.91	0.86	0.78	0.68	0.67	0.61	0.59
Precision	0.86	0.77	0.71	0.58	0.99	0.54	0.46
Recall	0.96	0.98	0.87	0.85	0.5	0.71	0.86

Table 5: *A selection of results of the NER model comparison for test sentences similar to homepage titles and headings.*

As this experimental setup is very basic and was only meant to get an idea of the general performance of some of the publicly available NER models, small differences in any of the metrics are not significant. All the individual models beside the Stanford NE Recognizer are found to be in the middle or even the end of the list of compared models. The combination of models lead to higher recall (for union) and higher precision (for intersection) with a loss in the other metrics. Therefore, the Stanford Named Entity Recognizer was selected for the preprocessing step of person name extraction.

6.2. Sampling and Preprocessing

The source of the training and test datasets for the scientist identification was the same data used for the web page classification task. Training data was obtained by randomly selecting 1705 web pages of the web page classification training dataset. The test dataset was obtained by randomly selecting 83 web pages from the web page classification test dataset. Both datasets were preprocessed in the same way. The preprocessing is described next.

Firstly, the person names were extracted for each web page utilising the Stanford Named Entity Recognizer Server (sner) module [34]. The sner variant is substantially faster than the nltk python interface.

Secondly, different spelling variations of a name, in terms of number and abbreviation of first-names, were merged favouring the longest name. For example, the names "John A. Doe", "John Doe", "A. Doe", "J A Doe" would all be merged to "John A. Doe". This obviously is a very basic approach only meant to capture some of the different naming conventions found in publication lists. Different name order, short forms of names or nicknames were not covered. For the 1705 web pages in the training dataset, 36123 person names were extracted. For the 83 web pages of the test dataset 2106 person names were extracted. One of the extracted and merged person names per page was manually labelled as homepage owner. So it was not attempted to find the full, written out name of the scientist. A prediction is rather considered correct, if the predicted name is equal to the manually labelled, longest name variant appearing on the web page.

Thirdly, three categorical and four numerical features were extracted for each name. Those are the flags for the appearance of the name in the title, and any h1 or h2 tag. The numeric features are the total number of occurrence of the name, the count of the name in the first half of the page, the count of the name in the first third of the page and the number of parts a name consists of.

The categorical features are inspired by Gollapalli et al [2]. They propose the simple heuristic that the person name that appears first on the web page is actually the name of the homepage owner. Although they report only 30% exact matches, they argue that the Jaccard similarity [7] of the extracted and actual names of 80% shows that most extraction errors are due to name spelling variations.

In this work, instead of relying on the order of appearance of names on a web page, the source tag of the name was utilised. It was observed that the order of names on the rendered web page is not necessarily the same as the order of names in the html code. This could be the case when a navigation element of the web page lists the names of the scientist. In such cases the name order heuristic would only rely on the title of the page.

The numeric features were designed to support the classification if none of the person names for a page appear in the title or the h1 tag. It was observed that on homepages, that contain publication lists, the homepage owner's name appears more often than the co-researchers' names. Also a name that appears often on the page but not often at the top is unlikely to be the homepage owner. Most scientist names on homepages in the training data consist of two parts, a first and a last name. The number of parts a name consists of was added to

[7] The Jaccard similarity measures similarities between sets. The Jaccard similarity $J(A, B)$ given sets A and B is defined as: $J(A, B) = \frac{A \cap B}{A \cup B}$.

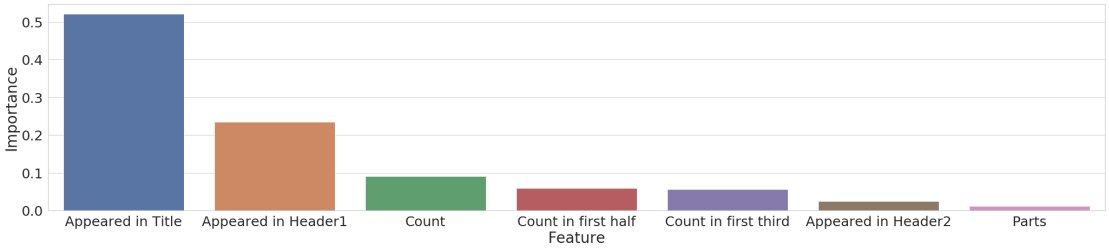

Figure 13: *Shows the feature importance for the homepage owner identification model as produced by the Random Forest Classifier.*

rule out name extraction and name merging errors. Names consisting only of one or a lot of parts are likely to be a preprocessing errors.

Finally, the three name count features were standardized by calculating the z-scores [8]. As the range of values of the number of occurrences for different web pages is very large, the standardization was done for each URL separately. In this manner, the relation of the values for each web page are kept while still producing similar value ranges for the total dataset.

Figure 13 shows the feature importance measure produced by the Random Forest Classifier. It can be clearly seen, that the majority of the predictive capability of the trained model comes from the two categorical features. Those first two features already explain 74% of the importance measure. To explain 95% of the importance measure, all three name count features are needed.

6.3. Results

The results for a baseline heuristic and two feature subsets for the homepage owner identification model are presented in this section.

	Validation Data			Test Data		
Model	Precision	Recall	F1 Score	Precision	Recall	F1 Score
(1)	0.93	0.84	0.88	0.95	0.92	0.93
(2)	0.93	0.92	0.92	0.95	0.93	0.94
(3)	0.96	0.91	0.94	0.95	0.93	0.94

Table 6: *The performance of homepage owner identification models. (1) is the baseline heuristic, (2) is a model with the four most important features (appearance in title and h1 tag, total name count and name count in the first half of the web page), (3) is the model with the full feature set.*

It has previously been stated that the presence of a person name in the title tag is a strong indicator, that the name belongs to the homepage owner. A

[8] The z-score z_i, given observation x_i, the sample mean μ and the sample standard deviation σ is defined as: $z_i = \frac{x_i - \mu}{\sigma}$

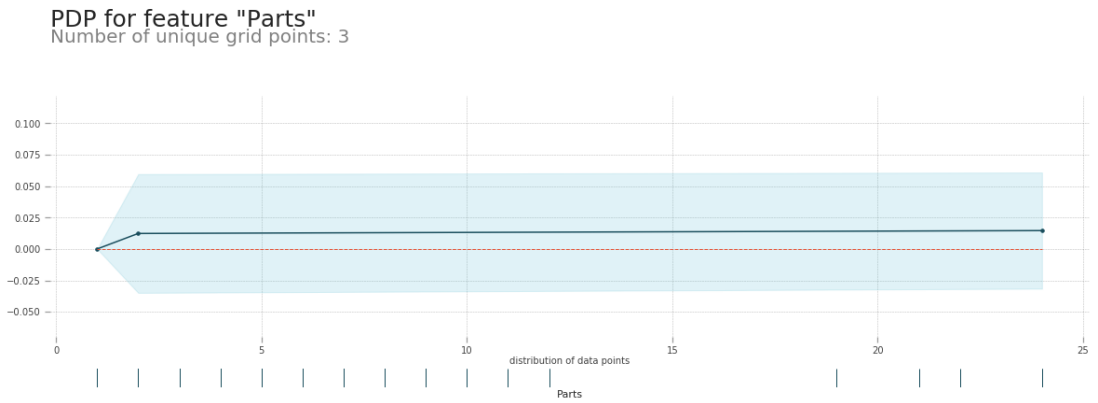

Figure 14: *Shows the partial dependency plot for the "number of name parts" feature of the homepage owner identification model. Below the x-axis the distribution is shown.*

simple baseline heuristic using this observation was utilised to evaluated the effect of additional features. For the baseline, a person name was predicted to be the homepage owner's name if it appeared in the title. Table 6 shows the performance measures for the baseline and two sets of features. Table 6 (1) shows the results of the baseline heuristic. It already achieved 88% F1 score for the validation data. The model (2) is using the four most important features, i.e. appearance in title and h1 tag, total name count and name count in the first half of the web page. It has a 4 percentage point higher F1 score.

With the model using all seven features (3) an increase in precision of 3 percentage points could be achieved, resulting in a F1 score of 94%. However the recall is slightly reduced, if compared to model (2).

The results for the test data are added to Table 6 to indicate, that the built models generalize well. Due to the fact, that the test dataset is quite small and small differences in the performance metrics can therefore not be considered significant, it is not used to compare the models.

6.4. Evaluation and Discussion

Most interestingly, the even more simple heuristic than the one proposed by Gollapalli et al. [2] performs very well. Although the html code for similar looking web pages as well as the usage of certain tags for certain page elements can differ quite a lot, as not every developer follows the same coding standard, the convention of writing the persons name in the title tag of homepages seems to be widely held. Using only the information about the appearance of the name in the title, 84% of homepage owners could be identified, while only producing 7% false positive errors. Obviously, this approach is dependent on the quality of the name extraction.

By adding a few simple features the prediction performance could be in-

Figure 15: *Classification results of the homepage owner prediction by confidence intervals. A random sample was drawn for each of the confidence intervals and manually classified. Correct classifications in bin 1, person identification model errors in bin 0, homepage classification errors in bin -1.*

creased. Although some of the features are listed with very low importance for the classification done by the Random Forest as seen in Figure 13 (i.e. the name count in the first third of the page, the name appearance in the h2 tag and the number of parts a name consists of), they still improve the precision by 3 percentage points. When examining the partial dependency plot (Figure 14) for the number of parts a name consists of, an increase in probability for the name to be the homepage owners name is observed for the transition from a name consisting of one to two parts. This feature seem to rule out a particular error. If a name only consists of one part it is less likely for the name to be the homepage owners name.

The examination of the errors produced by the classifier give hints for improvements of the model. Of the 15 false positive errors around 80% are due to name extraction and name merging errors. For the name extraction errors, either no person at all (e.g. "Carnegie Mellon", "-RRB-", "GeNeura Team") or only part of a name were extracted. The latter was found in cases, where special characters appeared in between the name components. In "Robert (“Bob”) Elliot Kahn" the names "Robert" and "Elliot Kahn" were extracted separately.

For the merging errors three types could be found. Firstly, nicknames are not handled (e.g. "Ted" denotes the same person as "Theodore"). Secondly, different spelling for umlauts or the presence / absence of accents is not unified. Thirdly, different ordering of names is not handled. Improvements of the name merging approach could be achieved through the equalization of the spelling before the comparison of the names and the addition of comparison cases for name ordering and nicknames.

When applying the person identification model to homepages predicted by the web page classifier, falsely classified web pages will have a detrimental effect on the performance of the person identification. In a sample of predicted homepage owners the errors for different intervals of person identification confidence were examined. Figure 15 shows the prediction results for the 10-30% and the 30-50% confidence interval for that sample. In the bottom interval

72% of homepage owner predictions are false and 86% of the errors are due to homepage classification errors. In the 30-50% confidence interval only 34% of homepage owner predictions were false, but the proportion of homepage classification errors is still high with 69%. Instead of the default value of 0.5 for the decision threshold, it was reduced to 0.3. So in the application of the model, the person name with the highest confidence value for each homepage is selected, omitting predictions with less than 30% prediction probability. In this manner, a substantial amount of correct person names could be added to the resulting data, while introducing an acceptable amount of error. With further improvements of the homepage classifier, that threshold could further be reduced.

7. Runtime

In this work, the focus was the development and analysis of a machine learning approach for the information extraction from web documents. Optimizing the runtime was therefore postponed to a later phase, when the developed method should be applied to a web size corpus. Until now, python was used for most of the components, as it allows for fast development and prototyping. To get an idea of the runtime of the components and where optimization might be most impactful, rough runtime estimates are given in this section. The development machine was a AMD FX(tm)-8150 Eight-Core Processor at 3.6 GHz. It has 32 GB of Ram. When possible, parallel processing was utilised. The download of the storage locations and the WARC data was bound by the download rate allowed by the Common Crawl and Amazon Web Services servers. Querying the Common Crawl index server to get the storage locations of the WARC data in AWS for three different crawls took 1 hour for 5k entries. Downloading the WARC data from AWS for 400k web pages took 3 hours [9]. Extracting the features for the homepage identification model for the 400k WARC files took 8 hours and the application of the model to the extracted features 2 hours. The majority of the runtime of the model application was the preprocessing of the features. The actual prediction with the model takes merely minutes. In an optimized pipeline, the feature extraction and preprocessing could be done at the same time and should be implemented in a more efficient programming language. Extracting the features for the researcher identification, namely the person name extraction with the Stanford NER server, the merging of different name spelling, and the collection of metrics for each name took 1 hour for 6k web pages. The prediction (including the preprocessing) of the correct homepage owner on the 300k names extracted from those 6k web pages then took 20 minutes.

[9] A vastly different download time has be observed in one of the data acquisition attempts. A lot of server side connection termination and name resolution errors were the reason for continuous retry attempts. Those where handled with exponential back-off. Downloading 80k web pages then took 1 day and 15 hours.

8. Conclusion

8.1. Summary

In this thesis a machine learning approach for the identification of researcher homepages and the extraction of the researcher name was investigated. A F1 score of 68% could be achieved for the identification of homepages and a F1 score of 94% for the identification of the researcher name for the respective test data. The html data for the training of the models and the application of the models in the homepage and researcher identification pipeline was obtained from Common Crawl. Common Crawl provides easy access to web page html and meta data through web APIs. As the data is stored in the Amazon Web Services, it can be also accessed with cloud compute, enabling scalability to big data approaches. The model trained for the identification of homepages performed well, considering the issues with the training data sampling. The good performance on the validation data indicates, that the two view approach, the choice of features and machine learning algorithms are well suited for the homepage identification task. The examination of the researcher name identification revealed i.a., that the convention of writing the researcher name in the title of the homepage is widely applied. A heuristic based on that observation was refined with a machine learning setup to improve the name extraction significantly.

8.2. Future Work

The poor generalization capability of the individual homepage identification models leads to prioritizing possible improvements to the individual models in any future work. The pitfalls and suggested improvements of the training data sampling and the individual models feature extraction and selection have been discussed in detail in the sections 5.5.1, 5.5.2 and 5.5.3. Therefore, necessary improvements regarding a future work will be only summarized in this section. After that, potential next steps for the homepage identification and expansions on the information extraction aspect of this work will be outlined.

To develop a classifier that performs well in a real world scenario, the quality of the training data has to be improved. This can be achieved by acquisition and evaluation of gold standard datasets and the manual selection and labelling of web pages. The aim should be an unbiased dataset which covers a sufficient amount of various homepage types and ideally counter-examples, that are similar to homepages.

As for the individual models, the URL based model could be improved by adding information about the beginning and end elements of a URL and by better handling of non-English languages and composite terms. Kan [7] proposes methods of segmentation of composite terms and also expansion of

abbreviations, which could be added to the feature set presented in this work. The already discussed improvements for the page content based model are the expansion of the stopwords used in the preprocessing and topic modelling with Latent Dirichlet Allocation [2]. As in both cases the feature sets are very large, feature selection techniques should be utilised. By removing less important features, a reduction in noise in the data can be achieved. This is more important for non-tree-based machine learning methods as they do not inherently select important features. If the improvements lead to a classification model that performs sufficiently well on unseen real world data, further steps, which address the special properties of web page data are applicable. One of the special properties is the change of conventions, coding or design standards and tools for web development over time and subsequently the change in the data representing web page types. Another one is the accessibility of a vast amount of unlabelled data. Under such circumstances semi-supervised methods are often applied. Gollapalli et al. [3] have shown that co-training is well suited for the web page classification. Combining the improvements to the prediction probability estimates proposed by Tanha et al. [35] with a co-training approach could be the next step.

As for the homepage owner identification, improvements could be achieved with better name extraction and name merging procedures. Further work could be done in expanding the information extraction. Beside the name of the researcher, multiple other topics are of interest. For example the research field, the contact information, the image of the person, the members of the researchers team etc.

9. Acknowledgments

Firstly, I want to thank my supervisor Prof. Dr. Hannah Bast for the guidance, continuous assistance, as well as the numerous suggestions provided throughout this thesis.
I further want to thank my colleague Nils Flaschel. Our many discussions always left me with new ideas and opportunities for improvement.
Last but not least, I want to thank my family for the unconditional support and the family cohesion I experienced in the difficult times we had.

10. References

[1] Internet-Live-Stats, *Internet live stats: Total number of websites*, http://www.internetlivestats.com/total-number-of-websites/, Accessed: 2019-02.

[2] S. D. Gollapalli, C. L. Giles, P. Mitra, and C. Caragea, "On identifying academic homepages for digital libraries", in *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, ACM, 2011, pp. 123–132.

[3] S. D. Gollapalli, C. Caragea, P. Mitra, and C. L. Giles, "Improving researcher homepage classification with unlabeled data", *ACM Transactions on the Web*, vol. 9, no. 4, pp. 1–32, 2015, ISSN: 15591131. DOI: 10.1145/2767135.

[4] Common Crawl, *Common crawl*, https://commoncrawl.org/, Accessed: 2019-02.

[5] The Jupyter Project, *Jupyter notebook*, https://jupyter.org/, Accessed: 2019-02.

[6] Scikit Learn, *Pipeline*, https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html, Accessed: 2019-02.

[7] M.-Y. Kan, *Web Page Categorization without the Web Page*. New York, NY: ACM, 2004, ISBN: 1581139128. [Online]. Available: http://dl.acm.org/citation.cfm?id=1013367.

[8] E. Baykan, M. Henzinger, L. Marian, and I. Weber, "A comprehensive study of features and algorithms for url-based topic classification", *ACM Transactions on the Web*, vol. 5, no. 3, pp. 1–29, 2011, ISSN: 15591131. DOI: 10.1145/1993053.1993057.

[9] X. Qi and B. D. Davison, "Web page classification:features and algorithms", *ACM Computing Surveys*, vol. 41, no. 2, pp. 1–31, 2009, ISSN: 03600300. DOI: 10.1145/1459352.1459357.

[10] A. Onan, "Classifier and feature set ensembles for web page classification", *Journal of Information Science*, vol. 42, no. 2, pp. 150–165, 2016, ISSN: 0165-5515. DOI: 10.1177/0165551515591724.

[11] J. Boedecker, F. Hutter, and M. Tangermann, *Machine learning*, Albert Ludwigs University Freiburg Lecture, 2017.

[12] C. Manning, P. Raghavan, and H. Schütze, "Introduction to information retrieval", *Natural Language Engineering*, vol. 16, no. 1, pp. 117–120, 2010.

[13] L. Breiman, J. H. Friedmann, R. A. Olshen, and C. J. Stone, *Classification and regression trees*. CRC Press, New York, 1999.

[14] L. Breiman, "Random forests", *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[15] L. Breiman, "Bagging predictors", *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[16] A. Criminisi and J. Shotton, *Decision forests for computer vision and medical image analysis*. Springer Science & Business Media, 2013.

[17] B. Thomas, *Optimization*, Albert Ludwigs University Freiburg Lecture, 2016.

[18] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms", in *Proceedings of the twenty-first international conference on Machine learning*, ACM, 2004, p. 116.

[19] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, 10. Springer series in statistics New York, 2001, vol. 1.

[20] Amazon Web Services, *Common crawl registry of open data in aws*, `https://registry.opendata.aws/commoncrawl/`, Accessed: 2019-02.

[21] WARC, *The web archive format*, `https://iipc.github.io/warc-specifications/specifications/warc-format/warc-1.1/`, Accessed: 2019-02.

[22] Webkb, *World wide knowledge base - 4 universities dataset*, `https://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/`, Accessed: 2018-11.

[23] dblp, *Computer science bibliography*, `https://dblp.org/`, Accessed: 2018-11.

[24] G. A. Miller, "Wordnet: A lexical database for english", *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[25] Scikit Learn, *Grid search*, `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`, Accessed: 2019-02.

[26] Scikit Learn, *Random forest classifier*, `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`, Accessed: 2019-02.

[27] Scikit Learn, *Stochastic gradient descent classifier*, `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html`, Accessed: 2019-02.

[28] Spacy, *Industrial-strength natural language processing*, `https://spacy.io/`, Accessed: 2018-11.

[29] Scikit Learn, *Min-max scaler*, `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html`, Accessed: 2019-02.

[30] Spacy, *Entity recogniser*, https://spacy.io/api/entityrecognizer/, Accessed: 2018-11.

[31] Nltk, *Named entity chunker*, https://www.nltk.org/api/nltk.chunk.html, Accessed: 2018-11.

[32] Stanford Natural Language Processing Group, *Name entitiy recognizer*, https://nlp.stanford.edu/software/CRF-NER.html, Accessed: 2018-11.

[33] United States Census Bureau, *List of frequent names*, https://www.census.gov/topics/population/genealogy/data/1990_census/1990_census_namefiles.html, Accessed: 2019-01.

[34] Pypi, *Name entitiy recognizer server*, https://pypi.org/project/sner/, Accessed: 2018-11.

[35] J. Tanha, M. van Someren, and H. Afsarmanesh, "Semi-supervised self-training for decision tree classifiers", *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 1, pp. 355–370, 2017, ISSN: 1868-8071. DOI: 10.1007/s13042-015-0328-7.