

Melanie Neunerdt

This thesis contributes to sequence labeling tasks in the field of Natural Language Processing by introducing novel concepts, models and algorithms for Part-of-Speech (POS) tagging, social media text detection and Web page cleaning.

First, the task of social media text classification in Web pages is addressed, where sequences of Web text segments are classified based on a high-dimensional feature vector. New features motivated by social media text characteristics are introduced and investigated with respect to different classifiers. Two classification problems in the context of social media text classification are treated, (1) the problem of social media text detection and (2) a method for Web page cleaning for social media platforms. A new Web page corpus, particularly designed to train and test the classifiers on representative Web pages is created.

Moreover, a POS tagger for social media texts is developed. The need for a specialized tagger is due to the specific social media text characteristics and the high non-standardization of such texts. Based on these factors, a Markov model tagger with parameter estimation enhancements with respect to social media texts is proposed. Particular focus is put on reliable estimation of non-standardized tokens like out-of-vocabulary words. To that end, methods are proposed to improve the reliability of probability estimation. Moreover, a novel approach mapping unknown tokens to tokens either known from training or tokens which fall into a class represented by regular expressions is presented. Finally, for remaining unknown tokens, semi-supervised auxiliary lexica and adequate estimation from prefix and suffix information is proposed. Furthermore, we propose to combine sparse in-domain social media training data and a newspaper corpus by an oversampling technique which improves POS tagging accuracies significantly. Training and evaluation of the proposed POS tagger is performed on a new manually annotated German social media text corpus. Tagging accuracies are presented and compared to accuracies achieved with state-of-the-art POS taggers.

Finally, we show that the proposed social media text detection and Web cleaning methods, as well as the presented POS tagger can be efficiently used in the context of information retrieval for Web page corpus construction. By applying Web page cleaning and social media text detection to Web page corpora obtained from Web crawlers, the generated corpus can be further refined.

ISBN 978-3-86359-402-2



9 783863 594022

Part-of-Speech Tagging and Detection of Social Media Texts

Melanie Neunerdt



Part-of-Speech Tagging and Detection of Social Media Texts

Von der Fakultät für Elektrotechnik und Informationstechnik
der Rheinisch-Westfälischen Technischen Hochschule Aachen
zur Erlangung des akademischen Grades eines Doktors
der Ingenieurwissenschaften genehmigte Dissertation

vorgelegt von

Master of Science Software Systems Engineering,
Diplom-Mathematikerin (FH)
Melanie Neunerdt

aus Essen

Berichter: Universitätsprofessor Dr. rer. nat. Rudolf Mathar
Universitätsprofessor Dr.-Ing. Torsten Zesch

Tag der mündlichen Prüfung: 9. September 2015

Diese Dissertation ist auf den Internetseiten der Hochschulbibliothek online
verfügbar.

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

Melanie Neunerdt:

Part-of-Speech Tagging and Detection of Social Media Texts

1. Auflage, 2016

Gedruckt auf holz- und säurefreiem Papier, 100% chlorfrei gebleicht.

Apprimus Verlag, Aachen, 2016
Wissenschaftsverlag des Instituts für Industriekommunikation und Fachmedien
an der RWTH Aachen
Steinbachstr. 25, 52074 Aachen
Internet: www.apprimus-verlag.de, E-Mail: info@apprimus-verlag.de

Printed in Germany

ISBN 978-3-86359-402-2

D 82 (Diss. RWTH Aachen University, 2015)

Preface

This thesis was written during my time as a Research Assistant at RWTH Aachen University's Institute for Theoretical Information Technology.

First and foremost, I would like to thank my supervisor, Univ.-Prof. Dr. rer. nat. Rudolf Mathar, for giving me the opportunity to take a very unique part in pursuing my Ph.D degree. I would also like to thank Prof. Mathar for his continuous support and for being an excellent example of fair and practical leadership.

Many thanks to Prof. Dr. Ing. Torsten Zesch for taking the effort to referee this thesis.

A special thankyou goes to Michael Reyer, Gholamreza Alirezaei, Alexander Engels, Bianka Trevisan and Niklas Koep for their very helpful discussions and suggestions, and for proofreading parts of this thesis. Furthermore, I would like to acknowledge the support of my student workers Phillip Vaßen and Simon Ruppel.

I would like to express my deepest gratitude to all my former colleagues at the Institute for Theoretical Information Technology. You helped to create a comfortable and inspiring working environment, every day. Thank you for a good time, I will miss your company.

I am very proud that I had the opportunity to contribute to numerous exciting research projects that were carried out in close collaboration with the Chair of Textlinguistics and Technical Communication at the RWTH Aachen. It is my esteemed honor to thank you for the good time and all adventures.

Ein besonderer Dank gilt meinen Eltern und meinem Freund Marc, die mich über all die Jahre hinweg mit viel Geduld und Zuversicht unterstützt haben.

Finally, I am deeply grateful to all my friends and team mates for their patience and understanding throughout the whole Ph.D journey.

Aachen, Januar 2016

Melanie Neunerdt

Contents

1	Introduction	1
1.1	Social Media Texts	3
1.2	Outline	4
2	Sequence Labeling	5
2.1	Sequence Labeling Problem	5
2.2	Bayesian Decision Rule	6
2.2.1	Viterbi Algorithm	7
2.3	Training Methods and Algorithms	9
2.3.1	Maximum Likelihood Estimation	10
2.4	Feature Normalization and Discretization	12
2.5	Reducing Feature Space Dimension	15
2.5.1	Information Gain and Information Gain Ratio	15
2.5.2	Correlation-based Feature Subset Selection	17
2.6	Classification Methods	17
2.6.1	K -Nearest Neighbor	18
2.6.2	Decision Tree	18
2.6.3	Support Vector Machine	20
2.7	Performance Measures and Evaluation Methods	21
3	Social Media Text Detection and Classification	25
3.1	Related Work	26
3.2	Problem Description	30
3.3	Web Page Annotation	31
3.4	Web Page Features	33
3.4.1	Token-based Features	34
3.4.2	POS-based Features	36
3.4.3	HTML-based Features	37
3.5	Classification Approaches	39
3.5.1	Independent Labeling Model	39
3.5.2	Linear Chain Conditional Random Fields	40
3.6	Web Page Corpora	43
3.7	Evaluation	44
3.7.1	Social Media Text Classification	50
3.7.2	Web Page Cleaning	52
3.8	Conclusions	58

4	POS Tagging for Social Media Texts	61
4.1	Related Work	62
4.2	Tokenization	65
4.3	Extended Annotation Rules	67
4.4	Markov-Model Tagger	68
4.4.1	Token Normalization	69
4.4.2	Lexical Probability Estimation	71
4.4.3	Transition Probability Estimation	76
4.4.4	Learning from Out-Domain Data	78
4.5	Text Corpora	79
4.6	Evaluation	81
4.6.1	Impact of Enhanced Parameter Estimation	82
4.6.2	Impact of In- and Out-domain Training Data	87
4.6.3	POS Tag Confusion and Category Specific Evaluation	94
4.6.4	Comparison to State-of-the-Art Taggers	96
4.6.5	Portability to Other Languages	98
4.7	Conclusions	98
5	Information Retrieval Applications	101
5.1	Topic Detection	101
5.1.1	Experimental Results	102
5.2	Social Media Text Corpus Construction	103
5.2.1	Experimental Results	105
6	Conclusions and Future Work	107
	Acronyms	111
	List of Symbols	113
	Bibliography	117

1 Introduction

The social media aspect of the World Wide Web leads to constantly growing user generated content in the Web. Different types of social media platforms such as blogs, forums as well as news sites allow users to post comments. This kind of consumer-to-consumer communication can be used efficiently to access user opinions for marketing studies or technology acceptance research. Among a variety of different topics, emerging technologies in particular are oftentimes controversially discussed on different social media platforms. An example is mobile communication systems, which are taken for granted in today's use, even though the impact of radio emission on the health of the human body is not yet fully understood. This yields high potential for analysing Web discussions. A beneficial property of social media texts posted on the Web is the fact that the data is natural, authentic, and public. Furthermore, opinions from proponents, opponents as well as from impartial people can be obtained from different Web domains, i.e., different communities. However, the process of extracting and processing such social media texts is a challenging task for two reasons. First, the size of the World Wide Web makes it a challenging task to detect Web pages containing social media texts, which are at the same time topic-relevant. Second, the fact that social media texts are non-structured and non-standardized texts requires adequate *Natural Language Processing* (NLP) methods in order to extract, process and evaluate relevant information.

The work presented in this thesis addresses parts related to these two problems. Particularly, we deal with two different sequence labeling tasks in the context of NLP and propose two novel labeling concepts. First, the task of social media text classification in Web pages is addressed. We consider two related subproblems, first the detection of social media texts in Web pages for corpus construction and second the cleaning of Web pages containing social media texts, where the intention is to separate the main Web page content from, e.g., the template or commercials. Both tasks are solved by decomposing Web pages into a sequence of text segments which are subsequently labeled (classified) with the considered classes.

A variety of approaches dealing with certain text classification in Web pages [64] exists. The closest task to social media text classification is text genre classification in Web pages. However, common approaches like proposed in [45, 40, 34] only consider genre classes like research articles, reviews, reportage or fiction. Social media texts have not been considered so far. Furthermore, existing approaches aim at predicting one genre class for the whole Web page and consequently do not operate on a sequence of Web page segments. By contrast, state-of-the-art Web page cleaning approaches, [37, 18], are solved by sequence labeling, where for each text segment usually a binary decision

between *main content* and *non-content* is made. However, applying these approaches to Web pages containing social media texts results in a significant performance loss [50].

To solve the problem of social media text classification, initially we introduce a new Web page corpus, particularly designed to train and test the classifier based on consecutive Web text segments of a representative set of Web pages from different social media text platforms. We particularly propose to extend existing feature sets for text classification by features related to social media text specific characteristics. Their performance is studied in combination with different state-of-the-art classifiers where each text segment is classified independently. Additionally, we apply *Conditional Random Fields* (CRFs) to solve the sequence labeling problem modeling the dependencies between consecutive Web text segments. CRFs allow for flexible feature function constructions, particularly designed for discrete features. We investigate the applicability of different feature functions related to the considered problems and determine a usable set of feature functions. All approaches are both applied to social media text detection, as well as Web page cleaning.

Second, we deal with the problem of automatic *Part-of-Speech* (POS) tagging for non-standardized social media texts, where the goal is to label a sequence of tokens with adequate part of speech tag classes, for example the word *habe* (have) as *finite verb*. Recent results, [25, 51], have shown that applying state-of-the-art newspaper POS taggers such as the Stanford tagger [83] or TreeTagger [74] to social media texts leads to a significant performance loss. Other approaches dealing with non-standardized texts predominantly deal with the annotation of Twitter messages [24, 60, 68]. However, Twitter messages exhibit very specific characteristics such as hashtags or @-mentions. Following these specifications, approaches are adapted to these characteristics and POS tag sets are extended with Twitter specific tag sets. Hence, applying these methods to more general social media text types is not appropriate and significant tagging improvements are not achievable. Application of these methods to more general social media text types can not improve tagging accuracies. We address these problems and propose a Markov model tagger called *WebTagger* with parameter estimation enhancements for POS annotation of social media texts in general. The main difference to other models lies in the calculation of probability estimates. One central goal is to achieve reliable estimates for non-standardized elements in social media texts like out-of-vocabulary tokens. Therefore, parameter estimation for out-of-vocabulary (unknown) words is adapted in several ways. Furthermore, we propose to combine the sparse in-domain social media training data and a newspaper corpus by an oversampling technique to improve tagging accuracies. In contrast to existing approaches, the standard *Stuttgart Tübinger Tag Set* (STTS) for the German language is used without any extensions. This allows for straightforward integration of the POS tagger into existing NLP pipelines without further modifications. In addition, to the *WebTagger*, a new social media text corpus for training and test purposes is developed which is an indispensable task.

Finally, we show that the proposed social media text detection and Web cleaning methods, as well as the presented POS tagger can be efficiently used in the context of information retrieval for Web page corpus construction. By applying Web page

cleaning and social media text detection to Web page corpora obtained from Web crawlers, the generated corpus can be further refined. Resulting corpora can be used for marketing studies or technology acceptance studies depending on the requirements. Parts of this thesis have already been published in [57, 55, 51, 52, 56, 53, 54] and [50].

1.1 Social Media Texts

It has already been mentioned that special characteristics are inherent in social media text types [55]. Before the two methods are proposed, we motivate our work by the main characteristics which differentiate non-standardized social media texts from newspaper texts. These characteristics are important for two reasons: (1) They give an idea about potential features to be used in order to detect social media text types, i.e., to differentiate such texts from standardized texts, to solve the sequence labeling problem based on text segments. (2) For social media text POS tagging, they reveal the technical challenges that are posed and help to find adequate solutions.

In order to describe the characteristics in more detail we introduce four categories, motivated by the above aspects:

1. *Spoken language character* - The language is borrowed from spoken language and characterized by linguistic irregularities. In German social media texts, verbs are often shortened or merged (e.g., *hab, habs* - *have, have it*), prepositions are merged with articles (e.g., *aufm, überm* - *on the, over the*), articles are shortened (e.g., *ne, nen* - *a, a*), fill and swear words are used (e.g., *verdammst* - *damn*), reflection periods are verbalized by interjections (e.g., *hmm, äh*) or elliptical constructions are used (e.g., *Entschuldigung!* - *Sorry!*). Thus, the language is characterized by a lower degree of standardization or colloquial style compared to newspaper texts.
2. *Dialog form* - A dialogic style characterizes the communication in social media applications. Hence, the use of first and second person singular and plural formulations is predominant. On the other hand, newspaper texts are typically written in third person singular or plural as this type of texts is more descriptive in nature. Moreover, social media texts are dialogic texts where many anaphoric expressions (e.g., *die* - *this, that*) can occur.
3. *Social media language* - The language is characterized by the use of interaction signs such as emoticons (e.g., *:-)*), interaction words (e.g., *lol, rofl*), leetspeak (e.g., *w!k!p3d!4*), word transformations (e.g., *nix* - *nothing, iPhone*), using mixed languages in the same context and references such as URLs and filenames (e.g., www.google.de).
4. *Informal writing style* - The majority of user posts are written in an informal way. Hence, social media texts suffer from spelling errors, typing errors, abbreviations, missing words and incomplete sentence structures (e.g., missing punctuation marks), missing capitalization, character repetitions (e.g., *Helloooo*), and multiple punctuation (e.g., *!?!*, *!!!*).

1.2 Outline

Chapter 2 covers the mathematical preliminaries to the sequence labeling problems considered in this work. This mainly comprises fundamental classification methods in the field of NLP.

In Chapter 3 the task of social media text classification in Web pages is addressed. In a sequence labeling approach Web text segments are classified based on a high-dimensional feature vector. New features motivated by social media text characteristics are introduced and investigated with respect to different classifiers. Particularly, a Conditional Random Field approach with specialized feature functions is proposed to solve the sequence labeling problem. Two classification problems in the context of social media text classification are treated. Firstly, the problem of social media text detection in Web text sequences is addressed. Secondly, a method for Web page cleaning for social media platforms is proposed which enables automated content detection from Web text sequences. A new Web page corpus, particularly designed to train and test the classifiers on representative Web pages containing social media texts is created. The cleaning results achieved on German and English Web pages comprising social media texts are evaluated and discussed.

In Chapter 4 we develop a Part-of-Speech Tagger for social media texts. The need for a specialized tagger is due to the specific social media text characteristics and the high amount of non-standardization of such texts. Based on these factors, a Markov model tagger for social media texts with parameter estimation enhancements is proposed. Particular focus is put on reliable estimation of non-standardized tokens like out-of-vocabulary words. To that end, methods are proposed to improve the reliability of probability estimation. Moreover, a novel approach mapping unknown tokens to tokens either known from training or tokens which fall into a class represented by regular expressions is presented. Finally, for remaining unknown tokens, semi-supervised auxiliary lexica and adequate estimation from prefix and suffix information is proposed. Furthermore, we propose to combine sparse in-domain social media training data and an out-domain newspaper corpus by an oversampling technique which improves POS tagging accuracies significantly. Training and evaluation of the proposed POS tagger is performed on a new manually annotated German social media text corpus. Tagging accuracies are presented and compared to accuracies achieved with state-of-the-art POS taggers.

Before this work is concluded in Chapter 6, Chapter 5 demonstrates the application of the developed methods in the context of information retrieval for Web page corpus construction. By applying Web page cleaning and social media text detection to Web page corpora obtained from Web crawlers, the generated corpora are further refined in two different scenarios. Firstly, the Web page cleaning classifier proposed in Section 3.7.2 is used in order to achieve a topic relevant Web page corpus. Secondly, in the same way social media text detection proposed in Section 3.7.1 is used in order to set up a corpus of Web pages containing social media texts.

2 Sequence Labeling

This chapter gives a brief introduction to sequence labeling tasks in NLP and an overview of some important concepts and related work linked to the task. The methods and results are repeatedly used throughout the subsequent chapters. Before we give an outline of this chapter, some notations are defined. Boldface capital letters indicate matrices or a sequence of vectors and boldface lower case letters are used to denote vectors or sequences. When dealing with randomness, capital letters indicate random variables, capital boldface letters are used for random vectors and random matrices. In the subsequent chapters, the symbols and identifiers are extended with respect to the particular context.

The outline of this chapter is as follows: In Section 2.1 and 2.2, we give an introduction to the sequence labeling problem, give a short overview of the probabilistic models applied in this thesis and introduce the Bayesian approach which is used throughout this work. Subsection 2.2.1 describes the Viterbi algorithm with respect to the task of solving the maximization problem given by a Bayesian approach for a probabilistic sequential model and to predict the sequence of labels for a given sequence of observations. Section 2.3 discusses training methods and related algorithms to estimate the probabilistic model parameters based on a set of training data. Before we give a short overview about feature preprocessing steps in Section 2.4, Section 2.5 describes a number of methods to reduce the feature space dimension in order achieve less complex probabilistic models. In addition to the previously mentioned probabilistic classification approaches, some additional state-of-the-art classifiers are used in this work which are described in Section 2.6. Finally, we list standard performance measurements and evaluation techniques in Section 2.7 which are used to analyse the results achieved by the proposed methods. A very detailed explanation of this chapter's content can be found in [17, 6, 42, 59].

2.1 Sequence Labeling Problem

The need to label sequences arises in many different NLP problems. The sequential characteristic of spoken language and written text is obvious. Sequence labeling is a special type of classification that aims at predicting the associated label sequence $c_1^N = (c_1, \dots, c_N)$ for a given sequence of observations represented by a sequence of feature vectors $\mathbf{x}_1^N = (\mathbf{x}_1, \dots, \mathbf{x}_N)$. In this thesis we aim to label a sequence of words with its associated sequence of POS tags and a sequence of text segments with its corresponding sequence of text type labels, classes respectively. A common approach

is to use a probabilistic model and apply Bayesian decision rule. This results in the optimization problem:

$$\hat{c}_1^N = \arg \max_{c_1^N} P(c_1^N | \mathbf{x}_1^N) \quad (2.1)$$

where $P(c_1^N | \mathbf{x}_1^N)$ is defined as the conditional probability, determined by a probabilistic model. This is a huge optimization problem which in practice is simplified in different ways. In general these simplifications are based on independence assumptions of the observations \mathbf{x}_n or the labels \hat{c}_n . For the two considered sequence labeling problems, we choose models with different complexities and evaluate their usability. Here we give some general information about the models we use throughout this work. Detailed model assumptions are given in the corresponding Sections 4.4 and 3.5.

The easiest way to label sequential data would be to ignore the sequential dependencies completely by solving

$$\hat{c}_n = \arg \max_{c_n} P(c_n | \mathbf{x}_n) \quad (2.2)$$

for each $n = 1, \dots, N$. Such an approach is in general easier to solve, however, it fails to exploit any sequential patterns in the data such as coherences between neighboring observations. Suppose, for instance, that we observe a token in a text and would like to predict its part-of-speech. In addition to the token itself, the context of this token is of significant help in predicting the current part-of-speech.

A simple way to model statistical dependencies in a probabilistic model like equation (2.1) is to consider a *Markov* model which is described in more detail in Section 4.4. The POS tagger *WebTagger* proposed in this work is based on a Markov model.

Another common approach used in recent years are Conditional Random Fields, e.g., in [2] for POS tagging and in [80] for Web page cleaning, which are based on an exponential model combining feature functions to calculate probabilities. Feature functions are used to learn and describe relationships inherent in the training data and then combine these into a model. CRFs are explained in more detail in Section 3.5.2, where the method is applied to the sequence labeling task of Web text segments. Particularly in the field of NLP, good results have been achieved with CRFs for different sequential labeling tasks. Beside the CRF model, the earlier mentioned independent labeling approach is applied to the task of labeling a sequence of Web text segments.

Before a label sequence \hat{c}_1^N can be predicted by solving equation (2.1), the parameters of the probabilistic model need to be estimated. This process is called training. In this thesis we use Maximum Likelihood estimation which is commonly used in classification and explained in more detail in Section 2.3.1. Before that, some general principles and reformulations of Bayesian decision theory are explained in the following section.

2.2 Bayesian Decision Rule

Bayesian decision theory is a crucial statistical approach to the problem of classification. It assumes that the classification problem is given in a probabilistic way and all

probability distributions are known. A Bayesian classifier is based on the Bayesian decision rule, which minimizes the average probability of classification error if all true data probability distributions are known [6]. This is equivalent to selecting the class c with maximum a posteriori probability (conditional probability)

$$\hat{c}_1^N = \arg \max_{c_1^N} P(c_1^N | \mathbf{x}_1^N), \quad (2.3)$$

where \mathbf{x}_1^N is a feature vector, which represents the object to be classified. Probabilistic approaches based on modeling this conditional probability distribution, e.g., the later used CRFs, are referred to as *discriminative* or *conditional models*. According to Bayes' theorem the posteriori probability $P(c_1^N | \mathbf{x}_1^N)$ can be computed from $P(\mathbf{x}_1^N | c_1^N)$ according to:

$$P(c_1^N | \mathbf{x}_1^N) = \frac{P(\mathbf{x}_1^N | c_1^N) P(c_1^N)}{P(\mathbf{x}_1^N)} = \frac{P(c_1^N, \mathbf{x}_1^N)}{P(\mathbf{x}_1^N)}$$

with

$$P(\mathbf{x}_1^N) = \sum_{c_1^N} P(\mathbf{x}_1^N | c_1^N) P(c_1^N). \quad (2.4)$$

Instead of maximizing the class posteriori probability, the joint probability can be maximized. Joint probabilities are generally observable in the training samples and can be estimated from those. Problem (2.3) can therefore be rewritten with the joint probability according to (2.4):

$$\hat{c} = \arg \max_{c_1^N} P(c_1^N | \mathbf{x}_1^N) = \arg \max_{c_1^N} \frac{P(c_1^N, \mathbf{x}_1^N)}{P(\mathbf{x}_1^N)} = \arg \max_{c_1^N} P(c_1^N, \mathbf{x}_1^N) \quad (2.5)$$

Note that $P(\mathbf{x}_1^N)$ is a constant term in the maximization problem and therefore can be omitted. Such approaches which are based on maximizing the joint probability are referred to as *generative model*. Discriminative as well as generative models are used throughout this thesis. A detailed analysis and discussion of the main differences and advantages of the two approaches is given in [87] and [58].

2.2.1 Viterbi Algorithm

The task of solving maximization problems in the form of (2.5) for a given probabilistic sequence model is also referred to as inference problem. The number of possible label sequences grows exponentially with the length of the sequence, hence an exhaustive search is not applicable. A commonly used inference algorithm is the Viterbi algorithm [89] which solves maximization problems based on a Markov model. It searches the space of possible sequences to find the most probable sequence with a computational cost that only grows linearly with the length of the sequence.

In order to explain the Viterbi algorithm in more detail, we consider a k -order Markov model of the form

$$P(c_1^N, \mathbf{x}_1^N) = \prod_{n=1}^N \underbrace{P(c_n | c_{n-k}^{n-1})}_{\text{Transition Prob.}} \underbrace{P(\mathbf{x}_n | c_n)}_{\text{Emission Prob.}}. \quad (2.6)$$

where

$$c_l^p = \begin{cases} (c_l, \dots, c_p) & 1 \leq l \leq p \leq N \\ (c_l = c_{\text{start}}, \dots, c_0 = c_{\text{start}}, \dots, c_p) & l \leq 0 \end{cases}$$

with $l \in \mathbb{Z}$, $n \in \mathbb{N}$, $l \leq n \leq N$ and c_{start} a dummy label added to our set of classes \mathcal{C} , i.e., labels, $\mathcal{C}_s = \mathcal{C} \cup \{c_{\text{start}}\}$. Furthermore, we introduce the simplifying assumptions made in our later proposed Markov model POS tagger in order to describe the algorithm in more detail. Two simplifying assumptions are made in that model. Transition probabilities calculate the probability of one label given its k previous labels (k -order Markov assumption). Emission probabilities are given by the conditional probabilities of an observation \mathbf{x}_n only depending on the given label c_n at position n .

For the mathematical description of the algorithm, we introduce k -tuples of labels $\mathbf{c} = c_1^k$ which represent any sequences of labels with length k . Thereby, we introduce a new Markov model, where k -tuples are considered to be the states with transition probabilities $P(\mathbf{c}' | \mathbf{c})$. Note that if $n - k < 0$ then the longest history available is used for transition probabilities $P(c_n | c_{n-k}^{n-1}) = P(c_n | c_1^{n-1})$. Hence, for the first observation in the sequence, only emission probabilities and for following observations additionally the class history with a maximum length of k , which is known at position n are considered. Emission probabilities are assumed to be position independent and hence it holds that $P(\mathbf{x}_n | c_n) = P(\mathbf{x} | c)$. Algorithm 1 illustrates the Viterbi algorithm. Initially, we introduce a start sequence $\mathbf{c}_{\text{start}} = (c_1 = c_{\text{start}}, \dots, c_k = c_{\text{start}})$ of length k representing the case that no history is available at the starting point of the sequence. The start sequence is set to probability $\delta_1(\mathbf{c}_{\text{start}}) = 1$.

In each iteration, $\delta_{n+1}(\mathbf{c})$ calculates the probability of the most likely path to reach a possible class subsequence \mathbf{c} at position n . $\psi_{n+1}(\mathbf{c})$ determines the preceding subsequence \mathbf{c}' shifted by one, which leads to the highest probability in the current node. $\delta_n(\mathbf{c})$ is the probability of the path which leads to the preceding node. In principle, for each preceding node, the transition probabilities to the current node need to be calculated. However, the Viterbi algorithm works more efficiently and only stores $\delta_{n+1}(\mathbf{c})$ and $\psi_{n+1}(\mathbf{c})$ for each position n . Only these values are used in step $n + 1$. Finally, the end state sequence $\hat{\mathbf{c}}_{N+1}$ is read out and likewise the whole sequence of \hat{c}_1^N .

Note that the Viterbi algorithm is applied to solve the maximization problem based on the Markov model for our POS tagging approach and for the CRF based sequence labeling task of text segment classification. Here, the Viterbi algorithm is described referring to a generative Markov model. Notation looks slightly different when applying the Viterbi algorithm to a CRF model. However, the main idea remains the same.

Algorithm 1 VITERBI k -ORDER MARKOV MODEL(\mathbf{x}_1^N)**Input:** Sequence of feature vectors \mathbf{x}_1^N **Output:** Predicted sequence of labels \hat{c}_1^N

```

 $\delta_1(\mathbf{c}_{\text{start}}) \leftarrow 1$ 
 $\delta_1(\mathbf{c}) \leftarrow 0$  for  $\mathbf{c} \neq \mathbf{c}_{\text{start}}$ 
for each position  $n = 1 \dots N$  do
  for each  $k$ -tuple of labels  $\mathbf{c} \in (\mathcal{C}_s)^k$  do
     $\delta_{n+1}(\mathbf{c}) \leftarrow \max_{\mathbf{c}'=(c'_1, c'_1, \dots, c'_{k-1}), c'_1 \in \mathcal{C}_s} P(c_k | \mathbf{c}') P(\mathbf{x}_n | c_k) \delta_n(\mathbf{c}')$ 
     $\psi_{n+1}(\mathbf{c}) \leftarrow \arg \max_{\mathbf{c}'=(c'_1, c'_1, \dots, c'_{k-1}), c'_1 \in \mathcal{C}_s} P(c_k | \mathbf{c}') P(\mathbf{x}_n | c_k) \delta_n(\mathbf{c}')$ 
  end for
end for
 $\mathbf{c}_{N+1} \leftarrow \arg \max_{\mathbf{c}' \in (\mathcal{C}_s)^k} \delta_{N+1}(\mathbf{c}')$ 
for each position  $n = N \dots 1$  do
   $\mathbf{c}_n \leftarrow \psi_{n+1}(\mathbf{c}_{n+1})$ 
   $\hat{c}_n \leftarrow (\mathbf{c}_{n+1})_k$ 
end for
return  $\hat{c}_1^N$ 

```

2.3 Training Methods and Algorithms

Before a statistical classifier can be used for prediction, e.g., of a POS tag sequence, the probability distributions have to be estimated. This process is called training or learning. In this work, we focus on supervised learning techniques where estimates are determined based on a set of training samples. We introduce our training sequence $\mathcal{TR} = \{(\tilde{\mathbf{x}}_n, \tilde{c}_n) \mid 1 \leq n \leq \tilde{N}\}$, where the true class \tilde{c}_n is known for each object represented by a feature vector $\tilde{\mathbf{x}}_n$. \mathcal{TR} is a set of concatenated single training samples (sequences), e.g., concatenated texts in the context of POS tagging. Hence \tilde{N} denotes the length (size) of the total training corpus. Even if dependencies at tie points of single training sequences are not existent, it can be neglected due to the fact that training sequences are pretty long in the considered problems of word sequences and Web page segment sequences and do not influence classification results significantly. Generally, these training samples are created manually where each object is annotated with the true class by human experts. In the context of NLP, training samples are also referred to as *Gold standard*.

In this thesis we apply two types of supervised learning techniques, parametric approaches, where the classes of density functions are assumed to have a known parametric form and non-parametric approaches. Since the common parametric forms rarely fit the densities in most practical machine learning applications, these assumptions do not lead to reliable estimates. In particular, most of the densities have a single local maximum (unimodal), whereas many practical problems involve multi modal densities. Therefore, non-parametric techniques have been introduced where no distributional assumptions are made about the given features or classes. In contrast to estimating the

parameters of the distribution, we directly estimate the probability distribution, e.g., the conditional probability $P(c | \mathbf{x})$, e.g. in the K -Nearest Neighbor classifier. In the following sections, the parameter estimation methods used in this thesis and related algorithms to solve the resulting optimization problems are explained.

2.3.1 Maximum Likelihood Estimation

Maximum-likelihood estimation (MLE) in general is a method to estimate the parameters of a probability model. It is a common approach for parameter estimation in the training process of a classifier. The principle is to estimate the model's parameter for a given probability model by maximizing the probability of the training samples. Maximum likelihood methods nearly always have good convergence properties as the number of training samples increases. Furthermore, maximum likelihood estimation is very simple compared to other methods.

Consider a classifier for our sequence labeling task which is based on a model for the conditional probability $P(c_1^N | \mathbf{x}_1^N)$. The conditional probability model for $P(c_1^N | \mathbf{x}_1^N)$ is assumed to be known and has a parametric form, e.g. assuming an exponential form as given by the CRF model, see Section 3.5.2. To indicate the dependency of $P(c_1^N | \mathbf{x}_1^N)$ on parameters $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_p)^T$, it is written as $P(c_1^N | \mathbf{x}_1^N, \boldsymbol{\lambda})$. Then the likelihood function is defined as

$$L(\boldsymbol{\lambda}) = P(\tilde{c}_1^{\tilde{N}} | \tilde{\mathbf{x}}_1^{\tilde{N}}, \boldsymbol{\lambda})$$

over the training realizations $(\tilde{\mathbf{x}}_n, \tilde{c}_n)$ for $1 \leq n \leq \tilde{N}$. Ignoring the sequential dependencies of the training samples $\tilde{\mathbf{x}}_n$ and assuming that the training samples are - independent, and identically distributed (i.i.d.)- random variables, allows to multiply the probabilities over all training samples and results in

$$L(\boldsymbol{\lambda}) = \prod_{n=1}^{\tilde{N}} P(\tilde{c}_n | \tilde{\mathbf{x}}_n, \boldsymbol{\lambda}).$$

For analytical reasons, it is usually easier to work with the logarithm of the likelihood, than with the likelihood itself. Since the logarithm is a monotonically increasing function, it does not affect the result of the maximization problem. Hence, in practice the log-likelihood function

$$\ell(\boldsymbol{\lambda}) = \ln \prod_{n=1}^{\tilde{N}} P(\tilde{c}_n | \tilde{\mathbf{x}}_n, \boldsymbol{\lambda}) = \sum_{n=1}^{\tilde{N}} \ln P(\tilde{c}_n | \tilde{\mathbf{x}}_n, \boldsymbol{\lambda})$$

is used to solve the maximization problem. In general the maximum likelihood estimate is by definition the value that maximizes $P(\tilde{c}_1^{\tilde{N}} | \tilde{\mathbf{x}}_1^{\tilde{N}}, \boldsymbol{\lambda})$ over all $\boldsymbol{\lambda} \in \mathbb{R}^p$:

$$\hat{\boldsymbol{\lambda}} = \arg \max_{\boldsymbol{\lambda}} L(\boldsymbol{\lambda}).$$

Thus, a set of necessary conditions for the maximum-likelihood estimate for $\boldsymbol{\lambda}$ is given by a set of p equations

$$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{\lambda}) = \mathbf{0}$$

where ∇ denotes the gradient. For all but the most simple models, the $\hat{\boldsymbol{\lambda}}$ that maximizes $L(\boldsymbol{\lambda})$ cannot be found analytically and the optimization problem needs to be solved numerically. Depending on the form of the probabilistic model different algorithms are used. In the following subsection, the Limited memory BFGS algorithm is introduced which will be used in Section 3.5.2 to solve the maximum likelihood problem based on a CRF probabilistic model.

Limited Memory BFGS

The *Limited Memory BFGS* (L-BFGS) is an algorithm particularly designed to solve non-linear optimization problems and has efficiently been used to solve maximum likelihood estimation in sequence labeling problems. L-BFGS, [59], is based on the popular quasi-Newton algorithm BFGS, named after its inventors Broyden, Fletcher, Goldfarb, and Shanno who published it independently in four publications, [10, 22, 78, 28]. Adapting the original *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) by storing approximations of the Hessian matrices represented in form of a few vectors, only a limited amount of computer memory is used. Due to its linear memory requirement, the L-BFGS is particularly well suited for large-scale optimization problems which made it a popular algorithm for maximum likelihood estimation in classification where a large number of training samples represented by high-dimensional feature vectors are present. This is particularly the case in CRFs, see Section 3.5.2, where L-BFGS is the algorithm of choice.

We consider our log-likelihood function $\ell(\boldsymbol{\lambda})$ to be our objective function to be maximized over $\boldsymbol{\lambda}$. The k -th iteration of the algorithm to find the maximum has the form

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k - \alpha_k \mathbf{H}_k \nabla \ell(\boldsymbol{\lambda}_k),$$

where α_k is the step size determined in each iteration to satisfy the Wolfe condition [93]. This can be done by the backtracking line search approach [1].

The iteration rule is quite similar to the line search Newton method. The key difference is that an approximation \mathbf{H}_k of the true inverse Hessian matrix $(\nabla^2 \ell(\boldsymbol{\lambda}_k))^{-1}$ is used.

Instead of computing $(\nabla^2 \ell(\boldsymbol{\lambda}_k))^{-1}$ at every iteration, the \mathbf{H}_k is updated in each iteration according to

$$\mathbf{H}_{k+1} = \mathbf{V}_k^T \mathbf{H}_k \mathbf{V}_k + \rho_k \mathbf{y}_k \mathbf{s}_k^T$$

where

$$\rho_k = \frac{1}{\mathbf{y}_k^T \mathbf{s}_k}, \quad \mathbf{V}_k = \mathbf{I} - \rho_k \mathbf{y}_k \mathbf{s}_k^T$$

and

$$\mathbf{s}_k = \boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k, \quad \mathbf{y}_k = \nabla \ell(\boldsymbol{\lambda}_{k+1}) - \nabla \ell(\boldsymbol{\lambda}_k)$$

where \mathbf{I} is the identity matrix, $\boldsymbol{\lambda}_k$ is the current iterate and $\nabla\ell(\boldsymbol{\lambda}_k)$ is the gradient evaluated at $\boldsymbol{\lambda}_k$. The main contrast to the general BFGS algorithm is that the costs of storage and manipulating the inverse Hessian approximation \mathbf{H}_k , which will be generally dense, are reduced. A modified version of \mathbf{H}_k is stored implicitly by storing only a certain number m of the vector pairs $\{\mathbf{s}_i, \mathbf{y}_i\}$ for $i = 1, \dots, m$ that are used in the updating formula. The product $\mathbf{H}_k \nabla\ell(\boldsymbol{\lambda}_k)$ can then be obtained by choosing an initial Hessian approximation \mathbf{H}_k^0 , e.g., the identity matrix, and performing products and inner summations based on $\nabla\ell(\boldsymbol{\lambda}_k)$ and the pairs $\{\mathbf{s}_i, \mathbf{y}_i\}$. In contrast to the original BFGS algorithm, in each iteration an initial Hessian approximation \mathbf{H}_k^0 has to be chosen which is allowed to vary from iteration to iteration.

After the new iterate is computed, the oldest vector pair is deleted and replaced by the newest one $\{\mathbf{s}_k, \mathbf{y}_k\}$. Experimental results show that values between $m = 3$ and $m = 20$ produce satisfactory results.

2.4 Feature Normalization and Discretization

In this work, two types of features are considered, discrete features and continuous-valued features, particularly in the context of Web text segment classification (Section 3.4). Depending on the classifier or feature selection algorithm, see Section 2.5, feature normalization and feature discretization methods are applied. In order to describe the following methods, we introduce discrete random variables $\mathbf{X} = (X_1, \dots, X_j, \dots, X_d)$ with $X_j \sim (\tilde{P}(x_i))_{i \in \mathcal{I}_j}$ for all d features, where \mathcal{I}_j represents the index set for the feature values x_i from the training samples. Additionally, we introduce a set of discrete random variables $C \sim (\tilde{P}(c))_{c \in \mathcal{C}}$ supported on the set of classes \mathcal{C} with $C = |\mathcal{C}|$. In contrast to the real distribution P , $\tilde{P}(x_i)$, $\tilde{P}(c)$, $\tilde{P}(c, x_i)$ indicate empirical distribution over the training samples, where $\tilde{P}(c, x_i)$ is the joint distribution.

Feature normalization is basically motivated by the variety of ranges features might have. Hence, some classifiers, e.g., the K -Nearest neighbor classifier which is based on a distance between two feature vectors, do not yield to accurate results. Features with a large range would dominate the distance. Therefore, the range of all features is normalized. A common method is to linearly rescale the range of all features to a fixed interval, e.g. $[0, 1]$. For each feature $j = 1, \dots, d$ the range is rescaled by

$$\tilde{x}_{nj}^{(1)} = \frac{\tilde{x}_{nj} - \tilde{x}_j^{\min}}{\tilde{x}_j^{\max} - \tilde{x}_j^{\min}}$$

with

$$\begin{aligned} \tilde{x}_j^{\min} &= \min_{n=1, \dots, N} \{\tilde{x}_{nj}\} \\ \tilde{x}_j^{\max} &= \max_{n=1, \dots, N} \{\tilde{x}_{nj}\} \end{aligned}$$

so that $\tilde{x}_{nj}^{(1)}$ is the normalized value. In the same way as the feature values \tilde{x}_{nj} for the training samples are normalized, feature values for the unseen test samples x_{nj} have to be normalized. However, for real world classification problems, it is not guaranteed that the transformation applied to the realized values x_{nj} lead to values in the interval of $[0, 1]$ for new unseen test samples. Hence, applying linear transformation could lead to values $x_{nj}^{(1)} < 0$ or $x_{nj}^{(1)} > 1$. Usually, those values are treated as outliers and set to 0 or 1, respectively.

Another linear method is called standardization, i.e., Z -score normalization, where features are transformed in the way to be zero mean with standard deviation 1. Therefore, a translation and scaling is performed according to

$$\tilde{x}_{nj}^{(2)} = \frac{\tilde{x}_{nj} - \tilde{\mu}_j}{\tilde{\sigma}_j}$$

with mean value $\tilde{\mu}_j$ and standard deviation $\tilde{\sigma}_j$ of the j -th feature determined over all realizations in the training samples.

In this work the described feature normalization methods are particularly used in the context of Web text segment classification in Chapter 3, where features with different value ranges are computed. A linear feature transformation is applied, (1) for the K -Nearest-Neighbor classifier, see Section 2.6.1 and, (2) as preprocessing step before features are discretized, as explained in the following section. In addition to that, non-linear transformations are applied to the feature vector from original space into a higher-dimensional space in the context of Support Vector Machines, see Section 2.6.3. Common transformation functions applied in SVMs can be found in [6].

Feature discretization is particularly needed for classification tasks, which involve continuous-valued features. Although classification tasks often involve continuous-valued features, many supervised classification algorithms are developed for learning in discrete feature space. In order to handle continuous-valued features in such algorithms feature discretization methods need to be employed. Some algorithms such as the C4.5 decision tree [66] incorporate the discretization into the learning process. Others, require discretization as preprocessing step. In addition to the algorithmic requirements mentioned above, discretization can improve the classification accuracy and decrease the runtime of some algorithms, as shown by Dougherty et al. [16]. Therefore, discretization is additionally performed on integer-valued features (\mathbb{N}_0) belonging to the class of discrete features, if $|\mathcal{I}_j|$ is in the range of \tilde{N} . By doing so overfitting to the training data can be prevented. Aside from classification algorithms, some feature selection algorithms, e.g., based on an entropy criteria, require feature discretization for continuous features.

The general idea is to discretize the range of any continuous feature into a discrete feature by building intervals. Discretization methods can be categorized into unsupervised and supervised methods. Several discretization methods such as equal width interval binning do not make use of the class labels in the discretization process. Such

methods are referred to as *unsupervised discretization* methods. In contrast, discretization methods that consider the class labels are referred to as *supervised discretization* methods.

In the context of social media text classification, see Section 3.4, we introduce many non-discrete features in order to describe a Web page text segment. Therefore, we apply a supervised discretization method when applying a CRF classifier in order to achieve adequate feature functions, see Section 3.5.2. In this work we use the supervised *Multi-Interval Discretization* (MDL) method proposed by Fayyad and Irani [19]. An initial study has shown that applying unsupervised techniques leads to miserable classification accuracies, when applied to the proposed features for the task of social media text classification. The basic concept of MDL discretization is to recursively determine cut points and thereby split the range of continuous-valued features into multiple intervals. The split criteria is based on the class information entropy of training sample partitions to select bin boundaries. In addition to our feature random variables $X_j \sim (\tilde{P}(x_i))_{i \in \mathcal{I}_j}$, new random variables supported on all possible cut points $t_j \in \mathcal{T}$

$$Y_{jt} = \begin{cases} 0 & X_j < t_j \\ 1 & \text{else} \end{cases}$$

are defined for each feature j . For comparison of different cut points the information gain criterion (transinformation) is used, which is introduced in more detail in the following Section 2.5.1. For each given feature j the cut point is determined by

$$t_j^{\max} = \arg \max_t I(C, Y_{jt})$$

with

$$I(C, Y_{jt}) = H(C) - H(C|Y_{jt}).$$

The t_j^{\max} is selected as binary discretization boundary, over all possible partition boundaries. Note that the set \mathcal{T} of possible cut points results from the range of feature values for X_j over all training samples. Assuming \tilde{N} training samples with distinct values therefore results in $\tilde{N} - 1$ possible cut points.

Repeating this method recursively to both of the partitions induced by t_j^{\max} until some stopping criteria is reached, multiple intervals on the feature j are created. As a stopping criteria the *Minimal Descriptive Length Principle* is used. Hence, recursive partitioning induced by a cut point t_j^{\max} for a set of \tilde{N} training instances stops if

$$I(C, Y_{jt_j^{\max}}) < \frac{\log(\tilde{N} - 1)}{\tilde{N}} + \frac{\Delta(C, Y_{jt_j^{\max}})}{\tilde{N}},$$

with

$$\Delta(C, Y_{jt_j^{\max}}) = \log(3^{|C|} - 2) - (|C|H(C) - |C_0|H(C|Y_{jt_j^{\max}} = 0) - |C_1|H(C|Y_{jt_j^{\max}} = 1))$$

where $|\mathcal{C}_0|, |\mathcal{C}_1|$ are the numbers of classes in the resulting subsets. Since single partitions of the recursive discretization are evaluated independently using this criteria, some areas in the continuous spaces will be partitioned very finely whereas others (relatively low entropy) will be partitioned more coarsely. This is one of the main differences compared to the unsupervised approach.

2.5 Reducing Feature Space Dimension

NLP classification tasks generally allow for a variety of features to be determined. The same holds for our task of labeling a sequence of text segments where each text segment is subject to a variety of characteristics, e.g., number of tokens or position in the Web page. Hence, feature vectors representing such segments are naturally high-dimensional. Dealing with high dimensional feature vectors leads to complex models which in return results in high computational costs. In order to reduce the dimension of the feature space, feature selection or feature transformation algorithms are applied. Feature selection reduces the dimensionality by selecting only a subset of features, whereas feature transformation is a dimension reduction technique based on an approximation in a lower dimension of feature space. Feature dimension reduction techniques are further differentiated into supervised and unsupervised techniques. In contrast to supervised methods, which consider the predictability of a feature concerning particular classes, unsupervised methods only consider the features and their dependencies among themselves. In this work, we make use of two supervised feature selection methods, which are described in the following. A detailed evaluation applying these methods in the context of social media text classification is proposed in Section 3.7.

2.5.1 Information Gain and Information Gain Ratio

The information gain criterion evaluates the usefulness of a feature by measuring the gain ratio with respect to the classes to be differentiated. Hence, it is appropriate for supervised feature selection measures. Features are ranked according to their information gain and by selecting the top $p < d$ ranked features, the dimension d of the feature vector can be reduced to p . An adequate value of p is determined by considering the trade off between model complexity reduction and classification performance. Very small p might lead to a considerable performance loss. Note that the usability of each feature is evaluated independently. Hence, selecting the top ranked features might not lead to the best result if the selected features are highly correlated. Due to that, the information gain criteria is often used for an initial comparison of features instead of feature selection.

In the same way, we evaluate the relevance of the proposed features for social media text classification in Section 3.7. The information gain is calculated based on the empirical distribution of the training data. In general terms, the information gain

measures the change in entropy H from a prior class to a class that takes a feature information as given. In the context of information theory, the information gain is known as transinformation between any two random variables. However, in the context of classification the term information gain has been established since the measure is particularly used for evaluating the usability of a feature with respect to the considered classes based on the empirical distribution of the training data.

Based on the previously introduced notation, the information gain is given by

$$I(C, X_j) = H(C) - H(C | X_j)$$

with

$$H(C) = - \sum_{c \in \mathcal{C}} \tilde{P}(c) \log \tilde{P}(c)$$

and

$$H(C | X_j) = \sum_{i \in \mathcal{I}_j} \tilde{P}(x_i) H(C | X_j = x_i).$$

The information gain between C and X_j is given by

$$I(C, X_j) = \sum_{i \in \mathcal{I}_j} \sum_{c \in \mathcal{C}} \tilde{P}(c, x_i) \log \frac{\tilde{P}(c, x_i)}{\tilde{P}(c) \tilde{P}(x_i)}. \quad (2.7)$$

In practice, classification features have different values and different numbers of values they can attain. The main drawback of the information gain criterion is that it depends on that number of values, i.e., $|\mathcal{I}_j|$. A simple example shows that the criterion is biased towards multi-valued features. One of the features might be a customer's telephone number. This feature has a high mutual information, because it uniquely identifies the customers that are present in the training data. However, this feature does not give any information about new customers and hence would lead to overfitting.

To counteract this problem the *information gain ratio* criterion was introduced. In order to reduce the bias towards multi-valued features, the uncertainty of a feature X_j is taken into account. The higher the uncertainty of a particular feature, the lower the feature is weighted:

$$IR(C, X_j) = \frac{1}{H(X_j)} (H(C) - H(C | X_j)) \quad (2.8)$$

Features with $|\mathcal{I}_j| = 1$ are excluded in advance since $H(X_j) = 0$. Hence, $H(X_j) > 0$ holds for all remaining features. Both criteria are used to evaluate the relevance of the proposed features for social media text classification in Section 3.7.

Beyond general feature evaluation, the two criteria are successfully used in the training process of decision trees, see Section 4.4.3. In the decision tree context, the information gain is frequently preferred where the criterion is used as a decision criterion for building the tree, i.e., to decide what feature to test at which node. The information gain ratio criteria is subsequently used for pruning the tree. It is shown in different

works, e.g., [33, 74], that the information gain ratio promotes the development of un-even trees. An extensive comparison of the two criteria in the context of decision trees can be found in [33]. In order to achieve reliable values for continuous features, feature discretization is required as proposed in the previous section.

2.5.2 Correlation-based Feature Subset Selection

Finally, a feature subset evaluation considering the class predictive ability of each feature (supervised), along with the degree of redundancy/correlation between the features is applied. The *Correlation-based Feature Selection (CFS)* proposed in [29] ranks feature subsets according to a correlation based heuristic evaluation function

$$f(\mathbf{X}, C) = \frac{p \overline{\text{corr}}(\mathbf{X}, C)}{\sqrt{p + p(p-1) \overline{\text{corr}}(\mathbf{X})}}$$

where \mathbf{X} is a $p \times p$ matrix consisting of column-wise random variables $X_1, \dots, X_j, \dots, X_p$ supported on feature subset where $p < d$ is the size of the considered feature subset. C is a random variable supported on the set of classes $c \in \mathcal{C}$. The term

$$\overline{\text{corr}}(\mathbf{X}) = \frac{1}{p^2} \sum_{i,j} \text{corr}(X_i, X_j)$$

denotes the average feature inter correlation between individual features and

$$\overline{\text{corr}}(\mathbf{X}, C) = \frac{1}{p} \sum_i \text{corr}(X_i, C)$$

is the average feature-class correlation. Feature subsets are compared in terms of $|f(\mathbf{X}, C)|$, such that feature subsets with $|f(\mathbf{X}, C)|$ close to zero are ranked higher. The principal idea is to prefer subsets that contain features which are highly correlated with the predictive classes and uncorrelated among themselves.

With the evaluation function defined by $f(\mathbf{X}, C)$, it remains to define the feature subsets. Since dimension reduction is generally applied to high-dimensional feature spaces, the enumeration of all possible feature subsets is exhaustive. Therefore, in practice greedy stepwise forward or backward search methods through the space of feature subsets are used. A comparative study of correlation based feature selection and information gain ratio based methods is presented in [94].

2.6 Classification Methods

The simplest way to solve a sequence labeling problem is to predict the classes for each observation in a sequence independently as in equation (2.2). By doing so, a simple classification task has to be solved where each object is classified independently based on its feature vector. Any supervised classification approach can be applied in this

case. In this work we use the independent labeling approach to solve the sequence labeling problems based on Web page text segments, see Section 3.5.1. Three different state-of-the-art classifiers are applied in this thesis: (1) K -Nearest Neighbor classifier, (2) Decision Tree and (3) Support Vector Machine. The basic concepts of the different classifiers are illustrated in the following.

2.6.1 K -Nearest Neighbor

The general principal of a K -Nearest Neighbor (KNN) approach is to compare the object to be classified with its K nearest training samples and perform a majority decision over the present training sample classes. KNN, first proposed by Cover et al. [13], falls into the category of non-parametric probability estimation approaches. The basic concept of a non-parametric approach is to make only few assumptions about the form of the distribution, but rather work with methods based on frequency counts. More precisely, KNN is a type of instance-based learning or lazy learning where new test instances are classified by direct comparison with instances seen in training. The disadvantage of lazy learning include the large memory requirement to store the entire training set. Another disadvantage of the KNN classifier is that lazy learning methods are usually slower in classification which is mitigated by faster training phase. In general, a KNN classifier is most useful for large training sample sets with few classification features.

As one would expect from the name of the classifier, this rule classifies by assigning the label most frequently represented among the K nearest training samples. Nearest training samples are determined by calculating the distance to all training samples $\mathcal{TR} = \{(\tilde{\mathbf{x}}_n, \tilde{c}_n) \mid 1 \leq n \leq N\}$, with feature vectors $\tilde{\mathbf{x}}_n \in \mathbb{R}^d$ and class labels \tilde{c}_n . Let $y_{(i)} = \|\mathbf{x} - \tilde{\mathbf{x}}_i\|$, then we identify the smallest K elements by the sequence $(y_{(1)}, \dots, y_{(K)})$ with $K < N$. Based on the sequences, the KNN estimates the conditional probability by

$$P(c \mid \mathbf{x}) = \frac{|\{n \mid 1 \leq n \leq K, c = \tilde{c}_{i_n}\}|}{K}$$

where i_1, \dots, i_K are the indices of the K smallest elements. The derivation to this estimate can be found in [6] and [17].

The computational complexity of the KNN algorithm both in space (storage of training samples) and time (classification) is important to mention. In the most naive approach, the search for classifying a test sample for, e.g., $K = 1$, is $\mathcal{O}(Nd)$, with a d -dimensional feature vector. Different algorithms have been proposed to reduce the computational costs in nearest-neighbor searches, see [4, 49].

2.6.2 Decision Tree

Like KNN classifiers, decision trees belong to the class of non-parametric approaches. In this work, we consider the C4.5 decision tree algorithm from [66] which is commonly

used in the context of Natural Language Processing. In the context of POS tagging a slightly modified binary version is used for the estimation of transition probabilities. C4.5 is build up on the ID3 (Iterative Dichotomiser 3) algorithm from [65] with very crucial extensions for real life classification tasks (1) allowing continuous-valued features in addition to discrete features, (2) suggesting an adequate pruning technique to avoid overfitting and (3) allowing for classification with missing features.

In principle, the decision tree is built recursively in the training phase according to the training samples. The result is a decision tree where each non-terminal node represents a test on a particular feature. The decision tree can subsequently be used to classify new test samples \mathbf{x} by passing through the tree and reading out the class label, i.e., the estimated class distribution $\hat{P}(c | \mathbf{x})$ at the resulting terminal node. In order to choose the test (feature) for a node l , the information gain criterion is applied by

$$\hat{j} = \arg \max_{j \in \mathcal{X}^{(l)}} I(C, X_j)$$

with $I(C, X_j)$ as in equation (2.7), C a random variable supported on the set of classes \mathcal{C} , $X_j \sim (\hat{P}(x_i))_{i \in \mathcal{I}_j}$ a random variable supported on feature j and $\mathcal{X}^{(l)}$ the set of possible tests at node l . At the root node we initialize

$$\mathcal{X}^{(1)} = \mathcal{X}$$

with \mathcal{X} representing the set of all features. After determining \hat{j} for a node l , the corresponding set of possible features is updated by

$$\mathcal{X}^{(l_i)} = \mathcal{X}^l \setminus \{X_{\hat{j}}\}$$

for all child nodes $i \in \mathcal{I}_{\hat{j}}$, i.e., for all values of feature \hat{j} a child node is created. The training data set valid for the used node is partitioned with respect to the values $\mathcal{I}_{\hat{j}}$.

The information gain criterion was adopted from the original ID3 algorithm. Although it gives good classification results it has a strong bias towards tests with many outcomes. In order to counteract this problem, C4.5 proposes to use the information gain ratio criterion, see equation (2.8), as alternative criterion. This problem has already been addressed in more detail in the context of feature selection in Section 2.5.

Furthermore, C4.5 addresses the task of handling different feature types. It proposes two methods to deal with discrete (nominal) features and one for continuous (real-valued) features. For the discrete feature case, the standard test works with one outcome and branch for each possible value of the corresponding feature. A more complex test, often applied to avoid overfitting is to further cluster the feature values into a smaller number of groups where each group is represented by one value rather than considering the exact feature value. If the feature takes on continuous values, a binary test with outcomes $x_j \leq z$ and $x_j > z$ with a threshold z is performed. The additional challenge here is to determine an appropriate threshold/cut point z . The problem of determining these thresholds is equivalent to the feature discretization of continuous features treated in Section 2.4. Therefore, the training samples are sorted

according to the values of feature x_j which results in an ordered finite sequence of values v_1, \dots, v_{N_j} with $N_j < |\mathcal{I}_j|$. Any threshold lying between v_i and v_{i+1} divides training samples into two groups according to that features threshold, resulting in $N_j - 1$ possible splits. Similar to the other tests, the best threshold is determined by choosing the test (threshold) with maximum information gain (information gain ratio).

Another important issue in the context of decision trees is overfitting. A decision tree that correctly classifies every training sample might not be as good as a smaller tree that does not fit all the training samples. In general, this problem is counteracted by allowing binary splits only or applying pruning to the tree. C4.5 includes a pruning method based on estimating the error rate of every subtree and replacing the subtree by a terminal node if the estimated error of the new terminal node is lower.

Finally, Quinlan [66] introduces another constraint which proved to be useful in practice: For any split, at least two of the subsets must contain a number of α training samples. Experiments have shown that $\alpha = 2$ is an adequate value in many problems. In Section 4.4.3, we use a slightly modified version of the C4.5 algorithm for the estimation of transition probabilities in the context of POS tagging. Modifications are described in that section.

2.6.3 Support Vector Machine

Support Vector Machines (SVM) fall into the category of linear discriminant models. In contrast to probabilistic models, a discriminant function without underlying probabilistic model is used for classification. Note that the optimization problem to be solved is different from the one presented in equation (2.2). Instead of assuming that the form of the underlying probability distribution is known, in a discriminant approach the proper form of the discriminant function is assumed to be known and the training samples are used to estimate the parameters of the classifier.

The main idea of an SVM is to preprocess the feature vectors representing the objects in a higher dimensional space, typically much higher than the original feature space, in order to allow for a linear separation. Using an appropriate non linear mapping $\phi(\cdot)$ (kernel function) to a sufficiently higher dimensional space, the instances of two classes can be separated by a hyperplane. This is also known as kernel trick. An overview of different kernel functions is given in [17, 6]. Furthermore, the goal in training a SVM is to find the separating hyperplane with the largest margin in order to reach a high generalization of the classifier.

The general idea is to find a linear model of the form

$$g(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b \tag{2.9}$$

where $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^p$ with $d < p$ denotes a fixed feature space transformation function, \mathbf{w} is a normal vector and $b \in \mathbb{R}$ is a bias parameter and a new test object \mathbf{x} is classified according to the sign of $g(\mathbf{x})$. Consider a 2-class problem with a set of \tilde{N} training data

pairs $\mathcal{TR} = \{(\tilde{\mathbf{x}}_n, \tilde{c}_n) \mid 1 \leq n \leq \tilde{N}\}$ where $\tilde{\mathbf{x}}_n$ is a feature vector with corresponding label $\tilde{c}_n \in \{-1, 1\}$. Assuming that $\phi(\cdot)$ has been chosen appropriately, so that the training data is linearly separable in the higher dimensional feature space, then there exists at least one choice of the parameters such that a function of the form (2.9) fulfills $g(\tilde{\mathbf{x}}) > 0$ for training data pairs $(\tilde{\mathbf{x}}_n, 1)$ and $g(\tilde{\mathbf{x}}) < 0$ for training data pairs $(\tilde{\mathbf{x}}_n, -1)$.

However, there may be more than one solution that separates the classes exactly. As mentioned before, the idea is to find a hyperplane, which maximizes the margin to the training samples. To that end, we introduce the distance from a hyperplane to a given (transformed) point $\phi(\mathbf{x})$

$$\frac{\tilde{c}_n g(\tilde{\mathbf{x}}_n)}{\|\mathbf{w}\|} = \frac{\tilde{c}_n (\mathbf{w}^T \phi(\tilde{\mathbf{x}}_n) + b)}{\|\mathbf{w}\|}.$$

The margin is given by the perpendicular distance to the closest point \mathbf{x}_n from the training set. Thus the maximum margin solution is solved by the optimization problem

$$\arg \max_{\mathbf{w}, b} \frac{1}{\|\mathbf{w}\|} \min_n [\tilde{c}_n (\mathbf{w}^T \phi(\tilde{\mathbf{x}}_n) + b)]$$

This optimization problem can be converted into a less complex constrained optimization problem, which can be solved by using the method of Lagrange multipliers. A detailed description can be found in [87].

Originally, the SVM was developed to solve a two-class problem. In practice, we often have to deal with $C \geq 2$ classes. Different methods have been proposed to combine multiple two-class SVM in order to build a multi-class classifier. A commonly used approach is to construct C separate SVMs, in which each SVM separates the data of class c from the remaining $C - 1$ classes [88]. This approach is also known as *one-versus-the-rest* approach. A detailed treatment of SVMs can be found in [87, 11].

2.7 Performance Measures and Evaluation Methods

The classification problems treated in this thesis, are so-called multi-class problems, where each object is assigned to one, and only one of several non-overlapping classes. In order to measure the performance of such classifiers, common statistical measures are used. In the following the performance measures used throughout this thesis are briefly described. A more detailed description and analysis can be found in [79].

The correctness per class $c^j \in \mathcal{C}$ for classes $j = 1, \dots, J$ of such classifiers can be assessed by computing the number of correctly detected class samples of class c^j (true positive TP^j), the number of correctly recognized samples that do not belong to class c^j (true negative TN^j), and either samples that were incorrectly assigned to class c^j (false positive FP^j) or were not detected as c^j class sample (false negative, FN^j). FP^j and FN^j are also known as *Type I* and *Type II* errors known from statistical hypothesis testing. Note that in a multi-class problem, the correctness per class is

determined by treating all remaining classes as \bar{c}^j . Based on TP^j , TN^j , FP^j , FN^j different evaluation measures can be calculated. Note that the considered class c^j is referred to as positive class and any others as negative class.

Table 2.1 presents the most commonly used measures for multi-class problems. The

Measure	Formula	Evaluation focus
Accuracy	$\frac{1}{J} \sum_{j=1}^J \frac{TP^j + TN^j}{TP^j + FP^j + TN^j + FN^j}$	The average per-class effectiveness of a classifier
Precision (PR_j)	$\frac{TP^j}{TP^j + FP^j}$	Class agreement of the labels \bar{c}^j with the classifier labels \hat{c}
Recall (RE_j)	$\frac{TP^j}{TP^j + FN^j}$	Effectiveness of a classifier to identify labels \hat{c}
F-Score (F_{β_j})	$\frac{(\beta^2 + 1)PR_j RE_j}{\beta^2 PR_j + RE_j}$	Weighted average of precision PR_j and recall RE_j
Specificity	$\frac{TN^j}{TN^j + FP^j}$	Effectiveness of a classifier to identify labels which are not c^j

Table 2.1: Performance measures for a multi-class problem with classes $c^j \in \mathcal{C}$.

average accuracy is the most general evaluation measure which calculates the ratio between correctly classified instances and the total number of instances. However, the main drawback is that no insights into the classification performance of specific classes c^j are gained and it is not sensitive to unbalanced a priori class distributions. E.g., consider the case of a two class problem where the class probabilities are significantly different, the average accuracy can be significantly higher even if the class with the lower probability is detected particularly bad. A more detailed per-class evaluation is given by the precision and recall measures. Particularly in the field of information retrieval, e.g., for the result of a Web Crawler, see Section 5.2.1, the precision is important. In the context of information retrieval, it measures the retrieved documents that are relevant to the find. Recall, also known as sensitivity or true positive rate (TP rate), denotes the fraction of instances of a particular class c^j that are successfully detected. The F_{β} -Score can be interpreted as a weighted average of precision and recall. The balanced case with $\beta = 1$ is the harmonic mean. By choosing different values for β , the weights can be adapted. All previously described performance measures can also be calculated from the confusion matrix by building the ratios between summation of the columns and rows of the matrix. As the name indicates, it visualizes the confusion between two classes. Each column of the matrix represents the instances in their predicted class, while each row represents the instances in their true class.

Note that the average classification quality over all classes $c^j \in \mathcal{C}$ is in general assessed in two ways: One way is to average over the same measures calculated for c^1, \dots, c^J , e.g., the average over precisions PR_j , (macro-averaging) or the sum of counts TP^j , FN^j , TN^j , FP^j to obtain cumulative TP , FN , TN , FP and then calculate the performance measure, e.g., the precision, (micro-averaging). Since macro-averaging treats all classes equally while micro averaging favors classes with higher frequencies, we use macro-averaging in this thesis. Generally, each classifier is tested on multiple randomly collected combinations of training and test sets. A particular approach is to use a k -fold cross validation where the training data is separated into k equally sized subsets by random selection. In each run the classifier is trained on $k - 1$

subsets and tested on the remaining one. Both sequence labeling problems are basically evaluated in form of cross validations, see Section 3.7 and Section 4.6. All performance measurements are depicted consistently in percent.

Significance tests

In order to show that improvements achieved by a classifier compared to another are significant, statistical significance tests are needed. Diettrich et al. review and evaluate five statistical tests for comparing supervised classification learning algorithms in [15]. In a cross validation evaluation statistical tests for comparison of more than one algorithms on multiple trials (resampling) are needed. Furthermore, it has to be considered that the test sets in each trial of cross validation are indeed independent but the training sets are overlapping and hence dependent. The resampled paired t -test is an adequate significance test for a series of trials where in each trial the available training samples are randomly divided into training and test sets. However, considering the dependence between the training samples in a cross validation the corrected resampled paired t -test has been proposed in [48]. Note that the variance in the corrected t statistic may still be underestimated which may result in large t values.

Consider two classification algorithms A and B which are trained and tested via k -fold cross validation. The training data are generated by combining $k - 1$ subsets in each trial and hence are overlapping random samples. Let f_i^A (f_i^B) be the observed proportion of misclassified test samples by algorithm A(B) during trial i . Assuming that the k differences $f_i = (f_i^A - f_i^B)$, $i = 1, \dots, k$ were drawn independently from a normal distribution, the resampled paired t -test can be applied:

$$t = \frac{\bar{f}\sqrt{k}}{\sqrt{\left(\frac{1}{k-1}\right) \sum_{i=1}^k (f_i - \bar{f})^2}},$$

where $\bar{f} = \frac{1}{k} \sum_{i=1}^k f^{(i)}$ is the mean difference. However, the dependence of the training samples has to be considered. Therefore, the formula is adapted to the corrected resampled paired t -test according to

$$\tilde{t} = \frac{\bar{f}\sqrt{k}}{\sqrt{\left(\frac{1}{k} + \frac{k_2}{k_1}\right) \sum_{i=1}^k (f_i - \bar{f})^2}},$$

where k_1 is the mean fraction of data used for training and k_2 is the mean fraction of data used for testing. Under the null hypothesis this statistic has a Student's t distribution with $k - 1$ degrees of freedom. In this work significance tests are applied when comparing our approaches for social media text POS tagging, see Section 4.6.4 and Web page cleaning, see Section 3.7.2 to state-of-the-art methods.

3 Social Media Text Detection and Classification

Following different interests Not only linguists working in the field of Natural Language Processing are evermore interested in acquiring social media text corpora. A common example is the request of a topic-specific social media text corpus in order to perform an empirical or manual study on topic-specific user opinions or even perform automatic sentiment analysis. Founded on that request the problem of automatic social media text corpus construction has to be solved. Therefore, the problem of social media text classification in Web pages is processed in this work. To the best of our knowledge this problem has not yet been addressed by researchers so far. Strong related research topics as text genre classification and subjectivity detection, which have been treated in different research papers. However both objectives are not exactly solving our problem. The differences are explained in more detail in the *Related Work* Section 3.1. We pursue two closely related objectives with inherent social media text classification in this chapter.

First, we address simple social media text (comment) detection in Web pages. We propose a method solving a 2-class problem, which differentiates social media texts from other texts in a Web page. Applying this classifier could serve to filter a corpus by selecting only Web pages containing social media texts. The construction of such corpora is addressed in more detail in Section 5.2.1, where the here proposed methods are applied. In addition to a 2-class problem we solve a 7-class problem, where relevant meta informations, such as the title or the posting time are differentiated as additional classes. This very fine-grained classification would procure useful corpus information allowing for a more detailed analysis.

Second, the task of Web page cleaning is addressed. It is one of the most essential tasks in Web corpus construction. The intention is to separate the main content from navigational elements, templates, and advertisements, often referred to as *boilerplate*. Several approaches have been introduced to solve the problem of Web page cleaning, i.e., boilerplate detection. However, existing approaches do rather study the results achieved on pages containing social media texts. This is particularly reasoned by the fact that existing training corpora suffer from low numbers of Web pages containing social media texts (comments), e.g., the *L3S-GN1 corpus* introduced in [37], which contains only 1% comments. In this thesis, we particularly enhance Web page cleaning applied to pages containing social media texts.

Both tasks are solved in the way that at first a Web page segmentation is performed and second classification is conducted based on these segments. The resulting problem is a sequence labeling problem, where a sequence of segments has to be classified into one out of two classes. In both cases the social media text characteristics differing from characteristics coherent in standardized texts help for classification, hence, the same feature vectors are used. Furthermore, we introduce a new Web page training corpus, particularly designed to train and test the two classifiers on Web pages containing social media texts.

The outline of this chapter is as follows. We summarize related work considering the two sequence labeling problems in Section 3.1. In Section 3.2, we give a mathematical description of the considered sequence labeling tasks. Section 3.3 describes the annotation principals for Web pages. In Section 3.4 we propose token-, POS- and HTML-based features, which serve for the representation of a text segment. Section 3.5 introduces different classification approaches to solve the sequence labeling problems. Sections 3.6 and 3.7 introduce a new manually annotated training Web page corpus and discuss experimental results achieved with different approaches. A final conclusion is given in Section 3.8.

3.1 Related Work

The following section is differentiated into two parts, first related work to social media text classification is presented and second a short overview about state-of-the-art Web page cleaning algorithms is given. However, both approaches and their related work are strongly related in terms of text classification.

Social media text classification is a specific task of text classification, which is not directly known from literature. However, a number of strongly related works exists. The two main factors that characterize a text are, its content and its style. Both of which are interesting aspects for automatic text classification purposes. Classification approaches dealing with the texts content, mainly refer to the topic of a text. Whereas approaches considering the style of a text address a variety of objectives, e.g., Web text genre classification or subjectivity classification. Both text classification tasks particularly differ in their selection of classification features. Topic classification basically operates on token-level features such as n-gram language models, whereas genre classification often employs additional stylistic features, e.g., the mean sentence length. The task of social media text classification in Web pages is more related to text style classification approaches. In particular it's related to Web text genre classification and subjectivity classification.

Before state-of-the-art approaches dealing with these two problems are listed in the following, we would like to mention some fundamental work, see [77, 94, 44, 23] dealing with text classification in more general. Even if in all approaches rather thematic than style classification is addressed approaches and selected features are taken up in related

works. Beside a number of different classification approaches, the mentioned papers give an overview of feature selection algorithms, which are evaluated and compared. Lastly, Sebastiani et al. [77] discuss the main machine learning approaches to the task of text classification. In particular, three different problems, namely document representation, classifier construction and classifier evaluation are discussed.

With the increasing amount of Web information, much research dealing with Web document classification according to various criteria has been performed in recent years. One focus is the task of Web text genre classification.

Meyer zu Eissen et al. [45] differentiate between eight genres following a user study on genre class usefulness. Moreover, they examine various feature sets attempting to combine different kinds of information. They differentiate four feature types, style-related features, e.g. HTML tags, closed word/token features, e.g. dates or currency symbols, text statistics, e.g. letter digit frequencies and POS features, e.g. noun or verb frequencies. Using discriminant analysis, they report 70% average classification accuracy over a ten-fold cross-validation.

Lim et al. [41] also evaluate different feature sets and focus on the usefulness of information found in different parts of the Web pages, e.g. the body or title. In addition to HTML-related and textual features some URL-related features, e.g. depth of URL, are proposed for genre classification. Based on a corpus of 15 genres they indicate that the *main body* and *anchor text* features are the most effective with an classification accuracy of 75%.

Kessler et al. [34] identify four feature categories: structural features, e.g. POS tag frequencies, lexical features, e.g., accuracies of addressing terms, character level features, e.g. question mark frequencies, and derivative features, e.g. the average sentence length. For classification they use logistic regression and different neural networks that combine 55 features. Based on a corpus with 5 different genres a maximum accuracy of 75% is achieved.

Qi et al. [64] review Web classification approaches with respect to its features and algorithms. They particularly investigate Web-specific features and their usability for different Web page classification tasks, e.g., sentiment classification and subject classification. Beside, on-page features, directly located on the page to be classified, they investigate the usability of features of neighbors, which are found on the pages related in some way.

In [40] they distinguish genre-related features from topic-related features. To this end, they use a corpus annotated for both genre and topic. Considering the genre and topic class, they use the term frequency (tf) and term's document frequency (df) as features. Closely related terms to the topic of the Web page are eliminated. Using a set of seven genres they report mean precision and recall values of 86.7% and 86.6% over all genres.

The performance of different POS-related features is studied in [20, 71]. In [20] they propose to use POS histograms over a sliding text window as features. Compared to

the results achieved by a classifier working with POS trigram features a significant performance increase is reported.

Stamatatos et al. performed a series of feature evaluations for automatic text genre classification. In their first research papers [46, 81] they perform genre classification based on newspaper texts. In [46] they differentiate five newspaper genres according to their style: public affairs, scientific, journalistic, everyday communication and literary. Different style markers based on token- and POS-level, e.g., number of word per sentence or verb-noun ratios are introduced and evaluated. In [81] they propose a text genre detection approach based on a linear discriminant classifier, using common word frequencies. Four new newspaper genre classes are introduced adapted to the new training and test corpus. In contrast to other approaches the most frequent words are determined based on the entire written language (represented by the British National Corpus) in contrast to use the training corpus itself. Like in [34] results show that punctuations mark frequencies play an important role for classification.

In [82] they present their first work dealing with text genre classification applied to Web documents. By means of existing NLP tools, features on a more structural level, e.g., the ratio of noun phrases to the total number of chunks are used. They achieve better results on a Web page corpus with 11 different genre classes compared to a pure token-level approach.

Indeed, the mentioned Web text genre classification approaches differ in the classes they consider, however the number of classes is greater than five. The problem of social media text classification we deal with is more a detection problem (2-class) problem so that features can be selected more specifically. Furthermore, the considered genre classes are often related to newspaper genres and do not consider any social media text specific genre class.

In addition to Web text genre classification, social media text classification is strongly related to subjectivity classification. Most works dealing with subjectivity classification pursue the goal to improve information extraction systems. In [21] they focus on the task of identifying whether a news article from a Web page reports objectively or presents the authors opinion. Three different feature types, bag of words, part-of-speech tags and stylistic text statistics are investigated for this task of classification. In this work, they focus on the classifiers transferability to corpora dealing with different topics than in the training corpus. Results show that particularly, features based on POS statistics result in a more general classification model for this task.

Wiebe et al. intensify their work on sentence-level subjectivity classification. In [91] they propose to classify the subjectivity of a sentence according to the presence or absence of particular adjectives. Using a word clustering according to distributional similarity the feature space is extended with additional adjectives for classification. These features are further refined with the addition of lexical semantic features of adjectives. In a 10-fold cross validation they show that the extended feature space leads higher precision rates.

The work in [92] comprises a huge study on different features and their suitability for subjectivity classification. Sentence-level classification is performed, but also ex-

tended to document-level. Applying a KNN classifier using n-gram features based on a combination of POS tags and tokens, particularly improves the precision to detect opinionated documents.

A subjectivity classifier for improving information extraction results is proposed in [70]. In contrast to other approaches they concentrate on filtering out subjective documents from a data corpus about terrorist events. Their approach applies a rule-based classifier to an unlabeled corpus to create training data, which is used to train a Naive Bayes classifier at sentence level. In addition to precision improvement their results show that subjectivity filtering is a good complement to topic-based filtering for information extraction systems.

To the best of our knowledge subjectivity classification research particularly for German is not addressed so far. Since the proposed approaches use a number of language specific features, i.e., lexica, the transferability to German needs to be tested. Furthermore, most approaches are performed on sentence level, rather than a whole text segment. Beside these mentioned aspects two more differences compared to subjectivity classification in the existing approaches exist. First, HTML-based features, give additional structural informations, e.g., the position in Web page and format characteristics. Css-style classes are often related to their meaning and hence are frequently a good hint to the text type of the particular Web page segment. Second, particularly social media text characteristics, which are not present in standardized texts, e.g., emoticons, multiple punctuations, see Chapter 1.1, which serve as additional features for social media text classification.

Web page cleaning is strongly related to Web text classification in general, but differs in the way that classification is based on Web page segments and the specific 2-class problem is solved. However, classification approaches and the features used are overlapping.

Automatically extracting the main content from Web pages has been well studied and a wide range of methods has been proposed. Common alternative terms for Web page cleaning are content extraction or boilerplate detection. Although Web page cleaning is a very crucial step in the construction of Web corpora, only relatively little literature can be found in this area. The *CleanEval* shared task and competition, [3], is one of few works, which particularly aims at cleaning arbitrary Web pages with the goal of preparing Web data for the use as a corpus. The *shared task* competitors basically apply different classification algorithms based on a variety of classification features [3]. For instance, NCleaner, [18], extracts content by deleting boilerplate with regular expressions and uses n-gram language models to separate content segments from non-content segments. Spousta et al., [43, 80], employ Conditional Random Fields based on multiple features and thereby treat the problem of boilerplate detection as a sequence labeling task.

Kohlschütter et al., [37], analyse a representative set of features used by the approaches proposed by the *CleanEval* competitors. In addition to a boilerplate detection algorithm working with a small set of shallow text features, they introduce a new Web

page training corpus called *L3S-GN1 corpus*. Furthermore, they extend their approach with handcrafted rules for comment detection (11 indicator strings like, e.g., *User comments*), but achieve lower precision. Both corpora, *CleanEval* and *L3S-GN1*, contain only a little amount of comment pages, which are annotated as part of the main content. Kohlschütter et al., [37], even explicitly annotate comments as separate class. Nevertheless, their publication reveals few insights about the efficiency of extracting comments as main content.

3.2 Problem Description

In the following, we first mathematically describe the two related sequence labeling problems, which only differ in the classes, which have to be predicted. A Web page p is segmented into a sequence of M_p text blocks (segments), where each block is represented by a d -dimensional feature vector

$$(\mathbf{b}_m^p)^T \in \mathcal{X} = \mathcal{B}_1 \times \mathcal{B}_2 \times \dots \times \mathcal{B}_d.$$

A detailed description about single feature ranges \mathcal{B}_j for $j = 1, \dots, d$ is given in Section 3.4.

The aim is to predict the to $\mathbf{B}^p = (\mathbf{b}_1^p, \dots, \mathbf{b}_{M_p}^p)$ associated class sequence

$$\mathbf{c}^p = (c_1^p, \dots, c_{M_p}^p),$$

with $c_m^p \in \mathcal{C}$ and $C = |\mathcal{C}|$ for $m = 1, \dots, M_p$.

In the task of Web page relevance detection, we consider a 2-class problem, where we differentiate between the classes COMMENT (social media text) and NON-COMMENT. Additionally, we consider a 7-class problem where the classes COMMENT, ARTICLE, USER, TITLE, TIME, META and OTHER are differentiated. We assume, that the more fine-grained classification can improve classification accuracy on the COMMENT class, due to similar sequences of classes, e.g., a user name (USER) followed by a TITLE followed by a COMMENT, which are coherent in Web pages containing social media texts. Furthermore, this detailed differentiation between meta informations belonging to the posted comment, could further enrich a social media text corpus and thereby increase corpus quality. Application of such classifiers for social media text Web page corpus construction is discussed in Chapter 5. Here, the focus lies on solving the sequence labeling problem.

Considering the second problem of Web page cleaning, the set of text classes \mathcal{C} comprises CONTENT and BOILERPLATE. According to the classification result, the Web page is cleaned in a very simple way, where all segments classified as BOILERPLATE are discarded.

The following steps are performed equally for both problems with respect to the considered classes. As a feature vector a combination of selected features proposed for text genre classification and Web page cleaning from [41, 45, 37] is used. The resulting

combination of features is complemented by some new features, which are especially motivated by social media text characteristics. Altogether, three types of features are used, token-, POS- and HTML-based. A detailed description of the feature set is given in Section 3.4.

For Web page segmentation, we use the segmentation from Kohlschütters boilerpipe tool. Web pages p are segmented into atomic *text blocks (TB)*, represented by \mathbf{b}_m^p , by a simple split at each HTML tag, except for the $\langle a \rangle$ tag. The result is a very fine-grained segmentation, which might result in splitted articles or comments. To counteract this problem, we apply the same block fusion algorithm delivered with Kohlschütters *Article Extractor* and achieve a sequence of fused text blocks represented by

$$\mathbf{S}^p = (\mathbf{s}_1^p, \dots, \mathbf{s}_{N_p}^p),$$

with $N_p \leq M_p$, where feature vectors \mathbf{s}_n^p are calculated analogously to \mathbf{b}_m^p . The fused text blocks are later referred to *Article Segments (AS)*.

The resulting sequence labeling task is solved by the optimization problem

$$\hat{\mathbf{c}}^p = \arg \max_{\mathbf{c}^p} P(\mathbf{S}^p, \mathbf{c}^p) \quad (3.1)$$

where the label sequence $\hat{\mathbf{c}}^p$ is determined that maximizes the joint probability $P(\mathbf{S}^p, \mathbf{c}^p)$. Note that we propose solutions to both segmentation results AS and TB. The optimization problem for the TB segmentation is formulated and solved in the same way. This is a huge optimization problem, which needs to be simplified in practice. In Section 3.5 two approaches, which particularly differ in the degree of simplifying assumptions and hence the complexity of the model are proposed.

3.3 Web Page Annotation

Depending on the task it is more or less challenging to develop an annotation scheme and to decide, which annotation tool to use. In the context of Web page content annotation the main questions are (1) is the annotation performed based on the visual representation of the Web page or directly based on the source code and (2) is the annotation performed based on the resulting segments or before the segmentation. In this work a visual self-implemented annotation tool called *AnnotationHelper* is provided, which supports direct annotation in the visual representation of the Web browser *Firefox*. This approach is much more simple for the assessors, since they can directly assign classes according to their visual perception. Annotating based on the representation of the source code is in general much more confusing and time consuming. Following that decision, the annotation is performed directly on the Web page without any given segmentation. The main advantage is, that the data is segmentation independent and can still be used for different approaches. However, an adequate

mapping from the annotated Web pages to the resulting segments is needed. This mapping and associated difficulties are explained in more detail later in this section.

In general the preparing steps and annotation works as follows: The Web pages are stored locally and supplemented by an additional CSS header to implement background coloring of the annotated blocks/areas on the Web page (preprocessing step). For example, the CONTENT is colored in turquoise, social media texts, i.e., COMMENT, are labeled in yellow. This color-scheme simplifies the manual annotation process and allows to check at a glance the Web pages structure and the annotation progress, see Figure 3.5. For the training (gold standard) corpus, two annotators manually classify the selected Web pages offline using the DOM inspector function of the Web browser Firefox. The assessors annotate offline to prevent reloading of additional dynamic content by JavaScript. The tool follows and bases on the manipulation of HTML tags and includes the corresponding CSS class name. Marking the area to be annotated, with right click in the Web page opening Firefox' inspector function, the corresponding HTML code can be edited by the tool. AnnotationHelper simplifies the annotation process: Copying the corresponding HTML code, the class can be selected via clicking on the related button provided by the tool. Then, the code is pasted at the same position with adding the corresponding CSS class that colors the block on the Web page as defined in the header (preprocessing). According to the given CSS classes the annotation can be read out for further processing.

The annotation process is performed in to levels: (1) coarse-grained for Web page cleaning and (2) fine-grained annotation for social media text detection. Figure 3.5 shows the coarse-grained annotation, Figure 3.1 shows the fine-grained annotation of an example Web pages. (1) For the coarse-grained annotation, three classes are distinguished: CONTENT, BOILERPLATE and COMMENT_AREA. Note that only the first two classes are considered for the purpose of training the Web cleaner. However, since we are particularly interested in evaluating the accuracy achieved on Web page parts, where comments are posted, we additionally annotate such areas as COMMENT_AREA. The class CONTENT (turquoise) labels the whole textual content of a Web page including the (main) article, social media texts/comments (yellow), meta-data and functionalities referring to the article as well as the comments like ratings, the reply-function, navigational elements, keywords, information about the author and publication date and location. Within the CONTENT area, all comments, the entry form to reply on posts and all meta-information belonging to the comment-functionality, e.g., number of comments, authors' avatars, are characterized as COMMENT_AREA. All areas which are not marked by the annotator, are automatically assigned to the BOILERPLATE class. These areas comprise basically the Web pages template including all menu items, but also advertisement banners between article or social media text blocks or article teasers placed between article and social media texts to motivate the user for further reading of related articles.

(2) For the fine-grained annotation, six classes are distinguished: COMMENT, ARTICLE, TITLE, TIME, META. Note that all non-annotated areas are summarized into the class OTHER. For the purpose of social media text detection a 2-class problem

is solved, where the only relevant class is the COMMENT class. All differently and non-annotated areas are summarized to be NON-COMMENT in this case.

The COMMENT areas cover all social media text parts, such as blog posts, forum posts, comments posted to an article or any form of discussion between users written . ARTICLE covers all text parts that do relate to the main article's content. The main title and all subheadings within the article, as well as the titles on top of the COMMENT_AREA or repeating headings between the comments are annotated as TITLE. The class TIME covers all exact date and time information referring to the article's publication date as well as the comments' publication dates. In addition, all dates about the user such as the date of registration or the date of the last editing are annotated as TIME. The class META covers all meta information related to the article or the comments such as the number of comments or the related keywords. The class USER covers all user names or the authors' pseudonyms. In addition, particles such as *von (from)* or *geschrieben von (written by)*, *bearbeitet von (edited from)* are also annotated with the class USER.

3.4 Web Page Features

The number of potential features for any kind of text classification is huge and results in a large variety of proposed features known from literature, [41, 45, 37, 92, 21]. Pure token-based approaches like bag-of-words models can result in a huge feature vector dimensions. The potential feature space increases for the task of classifying text segments in Web pages, where additional HTML-based features and features based on predecessor and successor segments are available. Searching for a topic independent approach for social media text detection claims for carefully selecting token-based features, in order to avoid overfitting to the topic/content of the considered training data. Inconsiderately using token-level features may provide skewed results that describe a particular topic only. Study text at the functional level rather at the textual level claims for features at a higher, topic- and language independent level. We believe that a topic independent solution for social media text detection, requires a good trade-off between token-based and higher level features.

In the following, 245 features of three different types, token-based, POS-based and structural features based on the HTML source code are introduced. It is generally expected that the combination of several features from different types can be used to identify text segments as COMMENT, NON-COMMENT,CONTENT,BOILERPLATE. In our approach we combine modified features proposed for text genre classification [41, 45], with features for Web page cleaning [37] and introduce some new features for the particular task of social media text detection. The newly introduced features distribute over all three feature types. Feature suggestions are motivated by investigating the language in and style of social media texts, see Section 1.1, as well as the structure of Web pages like news sites, blogs and forums. The basic idea is to identify significant differences considering both aspects.

3.4.1 Token-based Features

Token-based features are easily accessible, without any text preprocessing. However, in order to develop a topic independent solution, token-level features need to be carefully selected. It might explode to a thousand of features, which make the classifier prone to overfitting to the particular content or even content topic. In our approach we extend several frequency count features by social media text related features. We believe that the additional features particularly serve to differentiate social media text segments from other text segments. Table 3.1 gives an overview of the different types of token-based features including 149 features in total. The table is structured column wise, feature names are followed by possible feature ranges and a short description of the feature type. A detailed description follows.

Feature	Range	Description
TO1-TO10	$\in \mathbb{R}_{\geq 0}/\mathbb{N}_0$	Overall Token/Character based frequencies
CA1-CA6	$\in \mathbb{R}_{\geq 0}/\mathbb{N}_0/\{0, 1\}$	Capitalization related frequencies
DT1-DT2	$\in \mathbb{N}_0$	Date/Time related tokens
TE1	$\in \mathbb{R}_{\geq 0}$	Mean Text Density
TE2	$\in \mathbb{R}_{\geq 0}$	Ratio between TE1 of previous segment and TE1
SO1-SO6	$\in \mathbb{R}_{\geq 0}/\mathbb{N}_0$	Social media text characteristics
SU1-SU2	$\in \mathbb{N}_0$	Subjectivity related features
SE1-SE2	$\in \mathbb{R}_{\geq 0}/\mathbb{N}_0$	Frequency of sentiment related words
PU1-PU17	$\in \mathbb{R}_{\geq 0}/\mathbb{N}_0/\{0, 1\}$	Punctuation mark related features
LC1-LC50	$\in \mathbb{N}_0$	Frequencies of top 50 CONTENT words
LC51	$\in \mathbb{N}_0$	Total frequency of top 50 CONTENT words
LC52-LC102	$\in \mathbb{N}_0$	Frequencies of top 50 FUNCTION words
LC103	$\in \mathbb{N}_0$	Total frequency of top 50 FUNCTION words
US1	$\in \mathbb{N}_0$	Frequency of USUAL words $\geq 1,000$
US2	$\in \mathbb{N}_0$	Frequency of UNUSUAL words with frequency of 1

Table 3.1: Frequency and ratio features based on tokens.

Initially, we adopt some general features from other text classification approaches and complement them by some task specific features represented by TO1-TO10, the number of characters, tokens, special characters, alphabetic characters, digits and token combinations of alphabetic characters and digits is calculated. Additionally, the mean token length and number of differing tokens/words in the segment, which is equivalent to the lexicon size of the text, is determined. Based on that, we measure the vocabulary richness in more detail by the ratio between the lexicon size and the number of tokens.

CA1-CA6 comprise features related to capitalization, which is generally considered with less importance in social media text compared to standardized texts. Another issue is, that capital letters are used to emphasize someones expression in social media texts. We differentiate between capitalized words and words completely written

in capital letters. Additionally, a binary feature is introduced to mark if a text segment starts with a capital letter. All tokens starting with an uncapitalized letter are counted and the ratio between such tokens and the capitalized tokens is determined. Furthermore, the same ratio is calculated based on character level.

The posting time or date is a common information given to each article or social media text. Therefore, DT1 calculates the occurrence of common date strings by regular expressions in a binary way, e.g. 10.11.99 or 10/11/99. The feature, DT2 additionally considers time related strings such as 10:50 and counts the frequency of such date and time related elements in the segment. This helps to detect common structures like the posting date typically located above a comment or an article. Beside the link density in a Web page a text density measure has been proposed for Web page cleaning by [37], originally developed for segmentation. The idea is to identify similar text densities in segments of the same class. This feature TE1 counts the number of tokens in a text segment divided by the number of lines after word-wrapping the text at a fixed column width, here 80 characters. Due to the side-effect of having an incompletely filled last line after wrapping, the latter is not taken into account, unless it is the only line in the segment. Furthermore, the ratio between TE1 of the previous segment and current text segment is determined.

In Section 1.1 the different types of social media text characteristics have been shown in a very detailed way. The following features are developed according to that characteristics. One of the most essential features is the number of emoticons counted in a text segment. Related to the dialog form the occurrence of complimentary closes, e.g. *Hey* or *Hallo*, is counted, as well as the occurrences of addressing forms like *Herr* or *Frau*. Using multiple letters in a word, e.g. *Haaaaallo* is very unique for social media data and is introduced as additional feature. Frequencies are counted by means of regular expressions. It is generally common to relate to other relevant comments in a comment itself by giving the corresponding URL, which is additionally counted. Finally, a counter of eight social media text related words (CA6), *lol*, *ne*, *nein*, *danke*, *bitte*, *jo*, *man*, *so*, *super*, *nichtmal* is used. The list of words is compiled from the particular training text segments assigned to the COMMENT class. We manually select the words considering their frequencies in such segments.

Strongly related to the previous features are subjectivity related features and sentiment related features. We differentiate two subjectivity feature types. First frequency counts of personal and possessive pronouns in the singular form, e.g., *ich* or *dir*, are summed up. Second the plural forms, e.g., *ihr* or *wir* are counted. Furthermore, some sentiment related features are defined. Adjectives in social media texts are inherently connected with evaluative judgments. Hence, frequency counts of positive and negative orientated adjectives are a good feature for differentiation. The *SentiWS* word list proposed in [69] is used for frequency counts. Supplementary, a weighted frequency is used as feature by multiplying with the polarity weight.

Punctuation mark related features PU1-PU16 comprise frequency counts of coherent tokens representing a punctuation mark in the given text segment, but at the same time frequencies of punctuation mark relevant characters. Full stops are differentiated

from, question marks, exclamation marks, commas, colons and semicolons in single counts and are complemented by a total count on character and token level. Additionally to the standard frequency counts, we count social media specific occurrences of multiple punctuation marks, e.g. !!!, !?! and put this in ratio to the total count of punctuation mark related tokens. Finally, we introduce a binary feature, which stores the information if a text segment is terminated with a final punctuation mark. This could be a useful criteria since social media texts are often completed with the users name in contrast to articles, which are terminated by a final punctuation mark.

Features based on lexical information are most commonly used for any type of text classification. Counts of function words as well as counts of content words play an important role in a sentence, hence we want to evaluate their usability for our approaches. In order to achieve a reliable list, the top fifty content words (open class words) including nouns, verbs, adjectives, interjections, and most adverbs are extracted from the training corpus. In the same way the list of function words is constructed, including prepositions, pronouns, auxiliary verbs, conjunctions, grammatical articles or particles. In total fifty frequency counts and one total counter are used as feature for both word types. Note that for the list generation the part-of-speech information is needed, which is taken by applying *WebTagger*. Additionally, all usual words with a frequency higher 1,000 and unusual words occurring only once in the training corpus are extracted. Based on these lists, two features, US1 and US2 are introduced, building the sum over the single word frequencies.

3.4.2 POS-based Features

POS analysis assigns the word of a sentence according to their part-of-speech and/or syntactical function. Using POS informations requires automatic POS tagging (higher level feature), which increases computational complexity. However, existing text classification approaches have shown, that POS information can increase classification accuracies. Differences in POS tag statistics for newspaper texts compared to social media texts reinforce the usability of such features for our purposes.

We employ *WebTagger* proposed in Chapter 4, which leads to particularly high accuracies for social media texts and performs adequately on standard texts. Hence, it is suitable for tagging the text segments of the present Web page corpus. Table 3.2 depicts the 33 POS-based features, which are explained in more detail in the following. Based on the automatically POS tagged training data corpus a list of the 15 most frequent POS tags is extracted. 15 features ST1-ST15 are calculated from that list counting the frequencies of such POS tags in the segment. Additionally, the frequency of POS tags indicating the imperative form of a verb are summed up and added as feature ST16. The imperative is a frequent instrument used in social media dialogs, e.g., question and answering forums, but rather not occurring in articles or any other text parts of a Web page. The features ST17-ST18 are basically introduced as typical social media text characteristics, which comprise the frequency of answer particles, e.g., *ja*, *nein*, *danke*, *bitte* and frequency counts of interjections, e.g. *ach*, *naja*, *lol* in

Feature	Range	Description
ST1-ST15	$\in \mathbb{N}_0$	Frequencies of top 15 STTS POS classes
ST16	$\in \mathbb{N}_0$	Frequency of imperative verbs
ST17	$\in \mathbb{N}_0$	Frequency of answer particles
ST18	$\in \mathbb{N}_0$	Frequency of interjections
ST19	$\in \mathbb{N}_0$	STTS POS class variety
PO1-PO5	$\in \mathbb{N}_0$	Frequencies of morphological POS classes
RA1-RA3	$\in \mathbb{R}_{\geq 0}$	Ratios between different POS classes
SE1-SE6	$\in \mathbb{R}_{\geq 0}/\mathbb{N}_0$	Sentence related frequencies/ratios

Table 3.2: Frequency and Ratio features based on POS information.

the text segment. ST19 simply sums up the different occurring STTS POS tags and thereby measures the POS tag variety in a text segment. In addition to features based on the fine-grained STTS POS annotation considering the morphosyntax we use counts based on the morphological information exclusively. Features PO1-PO5, comprise frequency counts of verbs, pronouns, adjectives, nouns, and adverbs. Features RA1-RA3 are adopted from many text genre classification approaches. They calculate ratios between morphological POS classes, the verb-noun ratio, the adjective-noun ratio and adverb-noun ratio. Note that POS tag informations are used to determine the end of sentence and hence features related to that are assigned to that feature category. Use POS information here, is more exact considering social media texts, where, e.g., a sentence is terminated by an emoticon, which is detected by *WebTagger* correctly.

The sentence related features S1-S6 comprise basically length informations. Apart from the mean sentence length in tokens, the median, maximum and minimum token length of a sentence is calculated. Long sentences, predominantly exist in standardized articles whereas sentences in social media texts are rather expected to be short. Elliptical constructions, i.e. one word sentences, are exclusive for social media texts. Additionally, the total number of sentences for a segment is calculated, so that the total amount of sentence related features results in five. Finally, we calculate the number of conjunctions per sentence, which is a common feature in text genre classification.

3.4.3 HTML-based Features

HTML-based features are the cornerstone off many Web page classification tasks. In this work we propose 63 structural features based on the HTML source code, a combination of HTML tag based, CSS class name and link based features. An overall list considering the different types is given in Table 3.3. Initially, absolute and relative segment position (P1,P2) features are introduced. They give essential structural information. A class is normally located in the same area of a Web page. For instance, BOILERPLATE segments are rather located at the beginning or end of a Web page

Feature	Range	Description
P1	$\in \mathbb{N}_0$	Absolute Web page position
P2	$\in [0, 1]$	Relative Web page position
L11-L14	$\in \{0, 1\} / \mathbb{R}_{\geq 0}$	Link related features
CS1-CS33	$\in \{0, 1\}$	CSS class names related features
HT1-HT22	$\in \{0, 1\}$	HTML tag related features
LE1	$\in \mathbb{R}_{\geq 0}$	DOM level related features

Table 3.3: Frequency and ratio features based on HTML informations.

in contrast to CONTENT segments. COMMENT segments are usually located at the end of a Web page but still followed by some BOILERPLATE segments.

L11-L14 comprise four link related features, which are crucial for Web page cleaning. In addition to the frequency of links in a segment a binary feature is introduced, which marks if the segment itself is a link. This serves to differentiate BOILERPLATE segments, which frequently fulfill that criteria. If a segment is a link itself or contains some links, the link density feature is different from zero. It is the ratio of tokens in a link environment(`<a> . . . <a\>`) divided by the total number of tokens. In order to measure the transition from the previous segment, the ratio between the link density in the previous and current segment is calculated as additional feature.

Whereas previous structural features are well-chosen for BOILERPLATE detection, the following features are rather useful for detecting comments and their surrounding elements, e.g. the title or user name. These features measure the occurrence of common CSS class names or common HTML tags itself. Binary features of 22 common CSS class names for comments, 5 for time related information and 3 for user related classes are introduced. A feature value is equal to one, if the class name contains the considered name/string. The 22 features are a mix of more general class names, e.g. *post* or *kommentar* and specific class names extracted from the training data, e.g. *postingtext* or *commentbody*. The time and user related features calculate the existence of typical names like *date, year, author* or *user*. In addition to the single CSS classname related feature, we introduce one overall binary feature per type, comments, time and user, which measures the existence of one of the mentioned class names. Furthermore, we use 22 HTML tag based, binary features, which measure the occurrence of a special tag. The list of HTML tags used is a set up of tags which indicate different functionalities of the particular segment, e.g. heading tags like *h1, h5* or commenting specific tags like *form*.

An HTML document is given in a tree structure. We assume that the position in a tree structure could be a good indication to differentiate the here considered classes. Therefore, we introduce the HTML tag depth of the segment to be classified as feature, which simply measures the depth in the DOM tree (LE1).

3.5 Classification Approaches

In order to solve the sequence labeling problems, we compare two different approaches. We successively reduce our independent assumptions, which at the same time increases the complexity of our classification models. Starting with a simple independent classification for each text segment in a Web page in Section 3.5.1, we evaluate three classification methods based on the described feature set. In order to integrate the dependencies from surrounding text segments, the feature vector is extended by the features of predecessor and successor text segments. Furthermore, we apply a Conditional Random Field classifier in Section 3.5.2, which is a more complex model integrating dependencies of the predictions and features from predecessor and successor text segments.

3.5.1 Independent Labeling Model

Initially, we make very strong independence assumptions: (1) The classes \hat{c}_n^p are predicted independently from predictions \hat{c}_r^p for $r \neq n$. (2) We assume that the class for a given text segment \mathbf{s}_n^p at position n only depends on some - here k - preceding and succeeding text segments. Hence, the optimization problem from formula (3.1) is reformulated into

$$\hat{c}_n^p = \arg \max_{c_n^p} P(\mathbf{S}^p, c_n^p) \quad (3.2)$$

for $n = 1, \dots, N$ optimization problems. In this approach, we apply different classifiers, which operate on a single feature vector as input. Therefore, we represent our segment \mathbf{s}_n^p by a feature vector

$$\mathbf{s}_{nk}^p = ((\mathbf{s}_{n-k}^p)^T, \dots, (\mathbf{s}_n^p)^T, \dots, (\mathbf{s}_{n+k}^p)^T)^T.$$

Note that, for the first and last segments of each Web page p , there are not enough predecessor and successor segments available. In such cases the number of considered predecessor and successor segments is reduced to the maximal possible amount. Hence, the problem can be reformulated to

$$P(\mathbf{s}_{n-k}^p, \dots, \mathbf{s}_n^p, \dots, \mathbf{s}_{n+k}^p, c_n^p) = P(\mathbf{s}_{nk}^p, c_n^p).$$

According to the Bayes' theorem we can rewrite the joint probability by

$$P(\mathbf{s}_{nk}^p, c_n^p) = P(c_n^p | \mathbf{s}_{nk}^p) P(\mathbf{s}_{nk}^p).$$

Since the probability of a segment $P(\mathbf{s}_{nk}^p)$ does not change with the label sequence and is a constant considering the optimization task in equation (3.2), solving

$$\hat{c}_n^p = \arg \max_{c_n^p} P(c_n^p | \mathbf{s}_{nk}^p) \quad (3.3)$$

leads to the same solution, i.e., classification result. Furthermore, we assume all probabilities to be position and Web page independent. Three different models are used to solve the optimization problem given in equation (3.3), a K -Nearest Neighbor (KNN) algorithm, a decision tree (C4.5), and a Support Vector Machine (SVM). A detailed description about the different methods can be found in Section 2.6. Note that a SVM is a classifier based on discriminant function, hence a slightly different optimization problem is solved, see Section 2.6.3.

3.5.2 Linear Chain Conditional Random Fields

Additionally, we apply a Linear Chain Conditional Random Field to the sequence labeling task. Conditional Random fields are more and more used in order to build probabilistic models for sequence labeling tasks in NLP. The main advantage compared to the independent labeling model proposed in Section 3.5.1 is that strong independence assumptions are relaxed in such models. CRFs allow for modeling the dependencies of consecutive text segments and their predicted classes.

CRFs, first introduced in [39], are probabilistic models, where the sequence of text classes \mathbf{c}^p is predicted, dependent on the whole sequence of observed text segments \mathbf{S}^p . They fall into the category of discriminative probabilistic models, where the conditional probability $P(\mathbf{c}^p | \mathbf{S}^p)$ is modeled directly, which is sufficient for classification. The main advantage of a discriminative model is that no modeling effort of observation probabilities $P(\mathbf{x})$ need to be expanded, since they do not affect the classification because they are constant with respect to the maximization problem. Furthermore, the conditional probability of the text class sequence can depend on arbitrary, non-independent features of the sequence of observations. Additionally, the probability of a transition between classes may depend not only on the current observation, but also on past and future observations. In comparison, generative models, e.g., Markov models, must make very strong independence assumptions on the observations, e.g., conditional independence given the classes, to achieve tractability. Note that for simplicity we replace $P(\mathbf{c}^p | \mathbf{S}^p)$ by $P(\mathbf{c} | \mathbf{S})$.

For our approach, we implement a Markov model like linear chain CRF. It is based on the model described in [39], however we use additional feature functions, which is essential for the resulting model. The models assumptions are that dependencies of \mathbf{c}^p , conditioned on \mathbf{S}^p form a linear chain. Therefore, by the fundamental theorem of random fields, see [32], the conditional probability distribution has the form

$$P_{\theta}(\mathbf{c} | \mathbf{S}) = \frac{1}{Z(\mathbf{S})} \exp \left(\sum_{n,p} \lambda_p t_p(c_n, c_{n-1}, \mathbf{S}) + \sum_{n,q} \mu_q e_q(c_n, \mathbf{S}) \right)$$

with an instance-specific normalization function

$$Z(\mathbf{S}) = \sum_{\mathbf{c} \in \mathcal{C}^M} \exp \left(\sum_{n,p} \lambda_p t_p(c_n, c_{n-1}, \mathbf{S}) + \sum_{n,q} \mu_q e_q(c_n, \mathbf{S}) \right)$$

where $\theta = (\lambda_1, \dots, \lambda_R, \mu_1, \dots, \mu_Q) \in \mathbb{R}^{r+q}$ are the parameters of the distribution. Feature functions t_r and e_q are assumed to be given and fixed. Their specific construction is described in more detail in the course of Section 3.5.2. Parameters λ_r and μ_q correspond to usual transition $P(c | c')$ and emission probabilities $P(\mathbf{s} | c)$ in a Markov model. Since, the parameters are not required to be log probabilities, it is no longer guaranteed that the probability distribution sums up to 1, unless we use the normalization constant $Z(\mathbf{S})$ for a particular \mathbf{S} to ensure P_θ to be a probability distribution.

Before the CRF can be used for classification the parameter estimation problem of determining θ needs to be solved. The parameter estimation is based on a set of training Web pages

$$\mathcal{TR} = \left\{ \left(\tilde{\mathbf{S}}^p, \tilde{\mathbf{c}}^p \right) \mid 1 \leq p \leq P \right\}.$$

As described in previous sections, we solve this by a maximum likelihood approach. In order to maximize the log-likelihood objective function

$$\hat{\theta} = \arg \max_{\theta} \ell(\theta)$$

with

$$\ell(\theta) = \sum_{p=1}^P \log P_\theta(\mathbf{c}^p \mid \mathbf{S}^p)$$

we apply the Limited Memory BFGS algorithm as described in Section 2.3.1.

Computing the likelihoods additionally requires an inference algorithm 2.2.1. Inference algorithms are employed in two cases for CRFs. First, the gradient based training requires computing marginal distributions $P(c_n, c_{n-1} \mid \mathbf{S})$ and computing the likelihood function requires $Z(\mathbf{S})$. Second, to apply the classifier and label an unseen sequence of text segments, the most likely (Viterbi) labeling $\hat{\mathbf{c}} = \arg \max_{\mathbf{c}} P(\mathbf{c} \mid \mathbf{S})$ needs to be computed. In case of a linear chain CRF both inference tasks can be performed efficiently and exactly by variants of the standard dynamic-programming algorithms for Markov models. Here, we use the to CRF adapted Viterbi algorithm, see Section 2.2.1.

Feature Functions

In general for a linear chain CRF, each feature function can be any real-valued function of the form $f(c_n, c_{n-1}, \mathbf{S})$. However in practice, boolean feature functions have been proven to be successful in the context of different applications [39, 12, 36, 43]. For our approach we construct a first-order Markov model like CRF with transition related feature functions and boolean emission related feature functions as proposed in [39]. Emission related feature functions based on the current observation are particularly extended by feature functions based on the observed predecessor and successor text segments.

Initially, we adopt the same model proposed by Lafferty, [39], and use one transition like feature function

$$t(c_n, c_{n-1}, \mathbf{S}) = \mathbb{1}_{\{c\}}(c_n) \mathbb{1}_{\{c'\}}(c_{n-1}) \quad (3.4)$$

for each transition pair (c, c') and additional emission like feature functions

$$e(c_n, c_{n-1}, \mathbf{S}) = \mathbb{1}_{\{c\}}(c_n) \mathbb{1}_{\{l_j\}}(s_{nj}) \quad (3.5)$$

for all class-observation pairs (c, l_j) , where $l_j \in \mathcal{S}_j$ is a realization of the feature j . The sets \mathcal{S}_j with $|\mathcal{S}_j| = L_j$ are defined as the set of all training data realizations of each feature $j = 1, \dots, d$. In total this results in $R = C^2$ transition related feature functions t and $Q = \sum_j CL_j$ emission related feature functions e .

For our approach we perform modifications to emission like feature functions in two ways. Initially, we perform modifications, which are referring to the feature vector extensions by k preceding and succeeding segment features for independent predictions, as proposed in Section 3.5.1. In addition to the feature functions proposed in equation (3.4) we create emission related feature functions for features of preceding segments

$$\begin{aligned} e(c_n, c_{n-1}, \mathbf{S}) &= \mathbb{1}_{\{c\}}(c_n) \mathbb{1}_{\{l_j\}}(s_{(n-1)j}) \\ &\vdots \\ e(c_n, c_{n-1}, \mathbf{S}) &= \mathbb{1}_{\{c\}}(c_n) \mathbb{1}_{\{l_j\}}(s_{(n-k)j}) \end{aligned}$$

and for features of succeeding segments

$$\begin{aligned} e(c_n, c_{n-1}, \mathbf{S}) &= \mathbb{1}_{\{c\}}(c_n) \mathbb{1}_{\{l_j\}}(s_{(n+1)j}) \\ &\vdots \\ e(c_n, c_{n-1}, \mathbf{S}) &= \mathbb{1}_{\{c\}}(c_n) \mathbb{1}_{\{l_j\}}(s_{(n+k)j}). \end{aligned}$$

The resulting number of emission feature functions increases by a factor $(2k + 1)$.

Apart from the extensions of feature functions e_k , emission feature functions are adapted to use a combination of two features j and m . The idea is to consider two features dependently and thereby use and identify good feature value pairs for the considered classification. Therefore, the feature functions are modified in the way

$$e_q(c_n, c_{n-1}, \mathbf{S}) = \mathbb{1}_{\{c\}}(c_n) \mathbb{1}_{\{l_j\}}(s_{nj}) \mathbb{1}_{\{l_m\}}(s_{nm}) \quad (3.6)$$

for all feature combinations j, m with $j < m$. This results in a total number of $\sum_{j, m, j < m} CL_j L_m$ emission feature functions.

Dealing with real-valued features when constructing binary feature functions results in an overfitted model. Furthermore, it leads to a significant increase of the models complexity, which makes parameter estimation more challenging and requires more training data to achieve reliable estimates. One way to counteract these problems, is to perform feature discretization from Section 2.4. Additionally, we perform different feature subset selections as preprocessing step, see Section 2.5. In the addressed tasks of social media text detection and Web page cleaning a 2-class sequence labeling problem is solved, so that the number of classes C is the minor part in complexity

increase. We are interested in the influence of different feature functions on problems with higher class dimensions. Therefore, the 7-class problem, see Section 3.2, for a more fine-grained Web page corpus classification is additionally evaluated.

3.6 Web Page Corpora

Evaluations are performed on different corpora investigating different aspects of the proposed methods. However, one Web page corpus called *GWebTrain* is the main evaluation corpus, which serves particularly for training but at the same time for testing. The additionally used corpora are in general annotated more coarse-grained and hence are not useful for training but serve for further validation. The *GWebTrain* corpus is a social media text Web page corpus enriched with manual annotation according to the guidelines described in Section 3.3 and hence serves to train the classifiers. It consists of 200 manually assessed German Web pages all from different Web sites/domains. Web pages contain forums, blogs or different news sites, which allow for commenting of the published articles. Samples are selected preferring highly commented articles or pages with social media texts exclusively. The Web pages topic is related to renewable energies with the topics "Windpark" (wind farm) (100 pages) and "Pumpspeicherkraftwerk" (pumped storage power plant) (100 pages). This is due to the fact that these data later serve for a topic detection study.

Class	# TB	Mean TB	# AS	Mean AS	# Tokens	Mean Tokens
Total	41,796	222 ± 120	31,305	167 ± 98	312,282	1,661 ± 1,171
BOILERPLATE	24,694	131 ± 126	23,179	123 ± 121	79,521	423 ± 522
CONTENT	17,102	91 ± 109	8,126	43 ± 56	232,761	1,238 ± 1,587
COMMENT_AREA	12,909	69 ± 87	6,191	33 ± 43	148,868	792 ± 1,081
NON-COMMENT_AREA	4,193	22 ± 22	1,935	10 ± 13	83,893	446 ± 507

Table 3.4: Class distributions in the *GWebTrain* Corpus with 200 Web pages calculated based on textblocks (TB) and fused segments (AS).

Distributions of the classes CONTENT, BOILERPLATE, and COMMENT_AREA at token, text block (TB), and fused segment (AS) level for *GWebTrain* are depicted in Table 3.4. Beside total corpus frequencies, mean values are depicted. Note that the standard deviation is depicted by \pm . In the first row of the table class independent segmentation statistics are depicted, which give an idea about the granularity of the different segmentation results proposed by Kohlschütter. In addition to the annotated classes, we calculate statistics on the CONTENT areas, which are not assigned to the COMMENT_AREA and call that NON-COMMENT_AREA. Comparing the total and mean numbers of TB and AS segments for BOILERPLATE and CONTENT, shows that particularly the main content of a Web page is further merged to bigger segments. This is established by the fact, that the content consists of long articles and comments and the BOILERPLATE parts contains many little links or menu items. COMMENT_AREA frequencies are consistently higher compared to the remaining

content area `NON-COMMENT_AREA`. It reflects the large proportion of social media texts inherent in the data.

Exemplary segmentation results of a Web page into text blocks and article segments are depicted in Figure 3.1 and 3.2. Segmentation results are illustrated with respect to the fine-grained annotation.

Background colors depict the manual annotation, whereas frame colors depict the resulting segment and its class assignment according to the manual annotation. Areas, which are not labeled by the annotator are not highlighted with any background color (`BOILERPLATE`). The colored frames depict the segments, which are used as training data. The following color scheme is used: `ARTICLE` in light green, `COMMENT` in yellow, `TITLE` in blue, `USER` in red, `TIME` in light olive and `META` in dark green. Frame colors are slightly darker than background colors for better illustration. Note that if a segment overlaps two different classes it is assigned to the class, with the higher number of tokens. For example in Figure 3.2, the first two posted comments are fused together, with user name (`USER`), and posting date (`TIME`). The whole segment is assigned to the class `COMMENT` due to the higher number of tokens. Comparing the results to segmentation results achieved on the third posted comment, shows that the comment is not fused to one comment segment.

Overall, the different segmentation results show that text blocks of articles and comment areas are fused adequately in general. Related links, menu items and headlines in boilerplate areas are kept in smaller segments, fusion of text blocks is only applied to few areas. For the task of Web page cleaning and social media text detection both segmentations are adequate. However, the article segmentation is not so appropriate if a fine-grained classification into the 7 classes is required. In that case too many text blocks from differing classes are fused to one segment.

Two additional corpora are introduced in order to show the transferability of the proposed methods to other languages exemplary to English and additionally evaluate results of the proposed methods on a benchmark corpus. Therefore, first the English *EWebTrain* corpus is introduced. *EWebTrain* consists of 50 English Web pages and is annotated in the same way as the *GWebTrain* corpus. Even if the corpus is significantly smaller it serves as training corpus for the introduced methods. Additionally, the *CleanEval* corpus, an English benchmark corpus serving for the validation compared to state-of-the-art methods, is used. *CleanEval* is part of a shared task on the cleaning of arbitrary Web pages, with the goal to prepare Web data for use as a corpus. The manually annotated corpus includes two divisions for English, a development set, and an evaluation set. For our experiments, we only consider the evaluation set consisting of 674 Web pages.

3.7 Evaluation

Evaluations are structured into four parts. First, we analyse the usability of the proposed feature types for the considered classification problems. Feature's qualities

Home | Blog | Klimadaten | Literatur | MWR | Kräfte | Autoren | Impressum

Der Wind schickt keine Rechnung... aber die Feuerwehr

9. Juli 2012 | Von [Hans Frauer](#) | Kategorie: [Energie](#) [Erneuerbare](#) [Wasser](#)

Generator Hans Frauer, Windrad brennt

Beckum (sp) – Großeinsatz für die Beckumer Feuerwehr zwischen Keiltinghausen und Sünninghausen: Der Generator eines Windrades fing am späten Freitagsnachmittag im Bereich Am Fimmersberg Feuer. 20 Kameraden der Beckumer Feuerwehr waren dort teilweise mit acht Fahrzeugen im Einsatz.

Dabei waren die Einsatzkräfte weitgehend zum Zuschauen verdammt. Denn eingestiegen konnten sie am eigentlichen Brandherd nicht.

Ein bedauerlicher Einzelfall? Wohl eher nicht. Weiter Beispiele finden sich hier: Wobei nicht immer Feuer im Spiel sein muss. Manchmal fallen Windräder auch einfach nur so um.

Im Fokus war letztes zu lesen, dass die Offshore Windkraft viele neue und krisensichere Arbeitsplätze schafft. Vorausgesetzt man ist ein wenig, ähem, abenteuerlustig:

Schwere Unfälle in Windparks
Ausbau der Windenergie forderte schon drei Tote

Die Energiewende hat mitunter tödliche Folgen: Nach FOCUS-Informationen gab es auf Baustellen deutscher Offshore Windparks bereits 80 schwere Unfälle. Drei Männer kamen dabei ums Leben – und es dürfte noch mehr Tote geben.

Und demnächst hier mehr über die **Spannungen der Wasserkraft** (weil auch der Regen keine Rechnung schickt...) und wie die Photovoltaik für einen Boom in der Baubranche sorgen kann.

Print PDF

TEILEN:

E-Mail Drucken Facebook +1 0

13 KOMMENTARE

1. **Hans Frauer** 9. Jul 2012 18:56

Klar, für Laien entstehen auch keine Wartungskosten der Windräder: brauchen die keine 240 Volt Stromversorgung für die aufwendige Regelungs- und Steuerungstechnik, keine Getriebe-Ölheizung und es brauchen auch keine Ozeanster zum regelmäßigen Ölwechsel zu den Getrieben auf die Kuppeln transportiert werden.

Und wenn... Wind kostet doch nix und off shore legen Wartungsschiffe ohne Probleme bei Wetter an, oftmals nimmt man Helikopter, was soll's.

Wenn man schon nicht rechnen kann, nicht schneit das der vielfache Aufwand, von wetterabhängigen Zufahrtswegen für ein und das gleiche Ergebnis, nämlich eine konstante Stromversorgung, viel teurer wird, als das was wir schon ausreichend hatten, dann wird deutlich was bei ist. Ein gigantisches Scheitlungsprojekt verschluckt Spekulationsgeld – nicht schad drum, wenn's weg ist, aber es versaut die Strompreise.

Die Bioerogas, die wie Pflze aus dem ländlichen Boden schaffen, sind angeblich ja auch völlig sicher, bis man auf die TÜV-Berichte stößt, auf die Optimaleraten und die Ammoniak-Versäuerungen der Umwelt.

Weicher manns aus Hygiene vor schimmern Kernschmelzen ein Land infiziert hat, wo eine Bande von Blindgängern und Neppen bis in die Politik verbanden unser Umwelt und Wohlstand zerstört, das ist einmalig.

Da hat Töpfer in seinem „Delirium“ recht.

2. **Hans Frauer** 9. Jul 2012 22:04

Jupp. Vor 10 Jahren gab es ja mal wilde Pläne, die Ostsee um meine Insel zuzustellen mit den Vogelschreibern. Bis jetzt sehe ich zum Glück noch keine.

Die Pläne existieren noch, nur sind die deutschen Kosten recht exorbitant.

Interessant, dass auch explodierende Kosten mal was Gutes haben können

3. **Hans Frauer** 10. Jul 2012 07:32

Pasend zum Artikel die neueste Innovation in Sachen Turmbau für IWKs.

<http://tinyurl.com/cwz6z4t>

Vera

PS: Wenn man glaubt die Messlatte für Intelligenz liegt am Boden, findet sich noch jemand der sie einbringt

Abonnieren: Bei neuen Kommentaren per E-Mail benachrichtigen

Suche

RSS-FEED

Artikel
Kommentare
Abonnieren

UNTERSTÜTZEN SIE UNSERE ARBEIT

Spenden Sie uns

Spenden

Oder falls Sie bei Amazon einkaufen, benutzen Sie diesen Link.

amazon.de

>Hier klicken

MEISTGELESENE ARTIKEL

Hoch die fächerübergreifende Sozialität?
Plauderecke 4
Fracking in den USA - 'Gasland' und die Folgen
JETZT REICHT'S!!! Oder, was ist am 11.03.2011 wirklich passiert?
Peak-Oil und Anti-Fracking Propaganda in der ARD - 'Wir haben noch 7 Jahre'
Aufwind nur für die Strompreise? Deutschlands Energiewende und die Realität

THEMEN

Artikel (455)
Bios (331)
Cancer (4)
Climategate (43)
Daten (14)
Energieerzeugung (199)
Erneuerbare Energien* (109)
Biomasse (20)
Erdgas (30)
Energie (22)
Kernenergie (81)
Kohle (18)
Photovoltaik (4)
Umweltfreundliches Gas und Öl (11)
Wasserkraft (7)
Windkraft (19)
Ernährung (18)
Gesundheit (7)
Gesundheit (22)
Internet (9)
Klimawandel (288)
Medien (88)
Profiteure (26)
Wasserschiff (112)
Kommentare (12)
Kopenhagen (11)
Kurioses (34)
Meerespiegel (13)
V&E-Topic-Areas (16)
Ökologismus (81)
Ökologische Mythen (24)
Tierechte (1)
Tierschutz (2)
Plauderecke (41)
Politik (262)
Emissionshandel (19)
Energiewende (47)
Innovationspolitik (28)
Kohlenstoff (6)
Nachhaltigkeit (31)
Wirtschaftsförderung (32)
Sichere und Eia (17)

Figure 3.1: Text block segmentation result (depicted by colored frames) with underlying fine-grained annotation (depicted by background colors).

ScienceSkepticalBlog

Politik | Wissenschaft | Klima & Energie

Home | Blog | Klimadaten | Literatur | WWP | Kräfte | Autoren | Impressum

Der Wind schickt keine Rechnung...aber die Feuerwehr

9. Juli 2012 | Von [Robert F.](#) | Kategorie: [Brennbares Erdgas](#) | [Wetter](#)

Generator-Hinat Feuer, Windräd brennen



Beckum (sp) – Großanatz für die Beckumer Feuerwehr zwischen Kellinghausen und Sünninghausen: Der Generator eines Windrades fing am späten Freitagsnachmittag im Bereich Am Fährberg Feuer. 25 Kameraden der Beckumer Feuerwehr waren dort teilweise mit acht Fahrzeugen im Einsatz.

Dabei waren die Einsatzkräfte weitgehend zum Zuschauen verdammt. Denn eingeleitet konnten sie am eigentlichen Brandherd nicht.

Ein bedauerlicher Einzelfall? Wohl eher nicht. Wetter Beispiele finden sich hier. Wobei nicht immer Feuer im Spiel sein muss. Manchmal fallen Windräder auch einfach nur so um.

Im Focus war letztes zu lesen, dass die Offshore Windkraft viele neue und krisensichere Arbeitsplätze schafft. Vorausgesetzt man ist ein wenig, ähem, abenteuerlustig:

Schwerer Unfall in Windrädern
Ausbau der Windenergie forderte schon drei Tote

Die Energiewende hat mitunter tödliche Folgen: Nach FOCUS-Informationen gab es auf Baustellen deutscher Offshore Windparks bereits 80 schwere Unfälle. Drei Männer kamen dabei ums Leben – und es dürfte noch mehr Tote geben.

Und demnächst hier mehr über die **Spannungen der Wasserkraft** (weil auch der Regen keine Rechnung schickt...) und wie die Photovoltaik für einen Boom in der Baubranche sorgen kann.

Print | PDF

TEILEN:

E-Mail | Drucken | Facebook | **8+1** | 0

13 KOMMENTARE

- Herrn Meier** 9. Juli 2012 18:56

Klar, für Lizen entstehen auch keine Wartungskosten der Windräder, brauchen die keine 240 Volt Stromversorgung für die aufwendige Regelungs- und Steuerungstechnik, keine Getriebe-Ölheizung und es brauchen auch keine Ozeanster zum regelmäßigen Ölwechsel von den Getrieben auf die Kuppeln transportiert werden.

Und wenn, Wind kostet doch nix und off shore legen Wartungsschiffe ohne Probleme bei Wetter an, oftmals nicht mal Helikopter, was soll's.

Wenn man schon nicht rechnen kann, nicht schneit das der vielfache Aufwand, von wetterabhängigen Zufahrtszeugen für ein und das gleiche Ergebnis, nämlich eine konstante Stromversorgung, viel teurer wird, als das was wir schon ausreichend hatten, denn wird deutlich was bei ist. Ein gigantisches Scheitlungsprojekt verschluckt Spekulantengeld – nicht schad drum, wenn's weg ist, aber es versaut die Strompreise.

Die Bioerogas, die wie Plitze aus dem ländlichen Boden scheffeln, sind angeblich ja auch völlig sicher, bis man auf die TÜV-Berichte stößt, auf die Optimalzahlen und die Ammoniak-Versuchungen der Umwelt.

Wetter treibt uns in Hysterie vor schrägen Kernkraftwerken ein Land infiziert hat, wo eine Bande von Blindgängern und Neppen bis in die Politik verbanden unser Umwelt und Wohlstand zerstört, das ist einmalig. Da hat Töpler in seinem „Delirium“ recht.
- Robert** 9. Juli 2012 22:04

Jupp. Vor 10 Jahren gab es ja mal wilde Pläne, die Ostsee um meine Insel zuzustellen mit den Vogelschreibern. Bis jetzt sehe ich zum Glück noch keine.

Die Pläne existieren noch, nur sind die geplanten Kosten wohl explodiert. Interessant, dass auch explodierende Kosten mal was Gutes haben können.
- Meier** 10. Juli 2012 07:32

Passend zum Artikel die neueste Innovation in Sachen Turmbau für IWK.

<http://tinyurl.com/cw6z8t4>

Venus

PS: Wenn man glaubt die Messlatte für Intelligenz liegt am Boden, findet sich noch jemand der sie eingibt.

Abonnieren: Bei neuen Kommentaren per E-Mail benachrichtigen

Suche

RSS-FEED

Artikel
Kommentare
Abonnieren

UNTERSTÜTZEN SIE UNSERE ARBEIT

Spenden Sie uns

Spenden

Oder falls Sie bei Amazon einkaufen, benutzen Sie diesen Link

amazon.de

> Hier Klicken

MEISTGELESENE ARTIKEL

Hoch die fächerbegriffene Sozialität?

Plauderecke 4

Fracking in den USA - 'Gasland' und die Folgen

JETZT REICHT'S !!! Oder: Was ist am 11.03.2011 wirklich passiert?

Peak-Oil und Anti-Fracking Propaganda in der ARD - Wir haben noch 7 Jahre!

Aufwind nur für die Strompreise? Deutschlands Energiewende und die Realität

THEMEN

Artikel (455)
Blog (531)
Energieerzeugung (199)
Gasland (4)
Klimatopas (43)
Jahres 141
"Ermessensfreie Energien" (109)
Biomasse (20)
Fracking (20)
Fisch (22)
Kernenergie (61)
Kobalt (18)
Photovoltaik (4)
Unkonventionelles Gas und Öl (11)
Wasserkraft (7)
Windkraft (19)
Erklärung (18)
Gesundheit (7)
Gesundheit (22)
Interner (9)
Klimawandel (269)
Medien (68)
Profiteure (26)
Wissenschaft (112)
Kommentare (12)
Kopfschmerzen (11)
Kurdas (24)
Wissenschaft (13)
Ökologie (16)
Ökologismus (81)
Ökologische Mythen (24)
Sensitivität (4)
Tierschutz (2)
Plauderecke (4)
Politik (282)
Energieerzeugung (19)
Energiewende (47)
Innovationspolitik (28)
Kernenergie (6)
Nachhaltigkeit (31)
Wirtschaftsförderung (32)
Schnee und Eis (17)

Figure 3.2: Article segmentation result (depicted by colored frames) with underlying fine-grained annotation (depicted by background colors).

are assessed and compared by the information gain and information gain ratio criteria. Furthermore, we evaluate our methods, by applying two 10-fold cross validations to the *GWebTrain* corpus. Cross validation splits are performed by randomly selecting Web pages. Classification performance is measured by the classes precision, recall, mean F_1 -Score and total accuracy. The metrics are depicted in percentage. All values are macro-averages (see Section 2.7) over the two cross validations. Due to the strong variety of class prior probabilities we especially compare mean F_1 -Scores instead of total accuracies. Classes such as NON-COMMENT in the 2-class problem or OTHERS in the 7-class problem occur with high probabilities. Note that performance measures are always calculated based on text block (TB) segments. The reason for that is that results are more comparable and the performance measure is more independent from potential segmentation errors as mentioned in the previous section.

In Section 3.7.1 results achieved for social media text detection, i.e., solving the 2-class problem COMMENT vs. NON-COMMENT and the more fine-grained classification into seven classes are evaluated. Section 3.7.2 evaluates the Web page cleaning methods, solving the 2-class problem CONTENT vs. BOILERPLATE. Finally, the proposed Web page cleaning methods are applied to the English benchmark corpus *CleanEval* to show the portability of our methods to other languages. It is shown that Web page cleaning performances lead to similar results compared to state-of-the-art approaches achieved on the English benchmark corpus, where no social media texts are covered.

For our experiments applying a KNN classifier, a decision tree (C4.5) and a Support Vector Machine, we use the WEKA software, [31]. All three classifiers are used with their standard WEKA parameters. The KNN classifier is applied with $K = 3$ neighbors and the euclidean distance. As a C.4.5 decision tree we use a tree, pruned with a threshold of 0.25 with a minimum number of 2 of instances per leaf. The SVM is employed with a polynomial kernel function.

Feature Evaluation

In this section, we investigate the different features by calculating the per-feature information gain and information gain ratio, see equation (2.7) and equation (2.8). Specifically, we compare and discuss the three types of features. Note that both criteria serve to assess and compare the usability of features with respect to the classes, but are not sufficient for determining a good subset of features for classification. Feature qualities are assessed independently and hence might be strongly correlated. We perform and evaluate subset selections of features with a correlation-based metric later in Section 3.7.1. However, calculating the per-feature information gain (ratio) allows for giving a first impression about a feature’s usability. First, we consider the 2- and 7-class problem in the context of social media text classification. Figure 3.3 shows the top 100 features in decreasing order according to their information gain ratio for the 2-class and 7-class problem. Different colors represent the three types of features: red for token-based, blue for POS-based and green for HTML-based features. The information gain ratio is lower than 0.27 for the 2-class and 0.47 for the 7-class problem.

Comparing features with similar ranks for the different classification problems shows a nearly constant difference of 0.1. This is caused by the fact that specific features are higher valued due to their relevance for one of the more specialized classes different from COMMENT. For example, some features are strong indicators for segments of the TITLE or TIME class. The amount of POS-based features ranked in the top 100 is significantly higher for the 2-class. Typical high ranked features in both classification tasks are the frequency of finite verbs (ST1-ST15), the frequency of adverbs (PO1-PO5), the frequency of interjections (ST18), the frequency of answer particles (ST19), the number of pronouns (PO1-PO5) or the verb noun ratio (ST1-ST15). Particularly, HTML-based features are ranked under the top 10 features for the fine-grained classification. Such features are CSS class name counts (CS1-CS30) of class names which are username, time stamp or comment specific. Additionally, link related features (LI1-LI4) such as the link density or the number of links in the segment yield high information gain ratios. Similar to HTML-based features, token-based features related to the COMMENT and TIME class are higher ranked for the 7-class problem. For example, counts of the content word *uhr* (LC1-LC50) and counts of date/time related tokens (DT1) are strong indicators to differentiate between the TIME class and other classes. Features based on frequency counts of content and function words (LC1-LC102) such as *ich* (*I*) or *so* reach high information gain ratios considering the 7-class problem. Social media text related token-based features (SO1-SO6) such as the number of emoticons, the occurrence of complimentary clauses yield high information gain ratios in both classification problems. In a second step, we evaluate the usability of the proposed features for the considered Web cleaning task. The information gain ratio is evaluated in comparison to the information gain criteria. Figure 3.4 shows the top 200 features in decreasing order of their information gain (ratio) for the 2-class problem CONTENT vs. BOILERPLATE. Comparing the values of the two criteria at similar ranks shows that differences are slightly higher for the top 100 features. Furthermore, the amount of HTML-based features ranked under the top 60 for the 7-class problem is doubled compared to the 2-class problem. This can be explained by the fact that most of the HTML-based features are binary features and hence ranked lower according to the information gain criteria. In contrast, many of the POS-based features are POS tag counts, i.e., elements of \mathbb{N}_0 where $|\mathcal{I}_j|$ is in the order of \tilde{N} . This leads to a higher information gain for such features, which is adjusted by calculation of the information gain ratio. Comparing information gain ratio ranks with the ranks achieved on the 2- and 7-class social media text classification problem shows very similar distributions over the top 100 ranks for the different feature types. Specific features are ranked similarly. In particular, social media text related HTML- and token-based features yield high ranks.

We conclude that for our task, in which features are a combination of binary, integer-valued (\mathbb{N}_0) and real-valued (\mathbb{R}) features with different value ranges, the information gain ratio seems to be the more adequate metric for the assessment and comparison of a feature's usability. However, results achieved with the decision tree in the following section will show that the information gain criterion can be effectively used to build a decision tree yielding high accuracies based on such features.

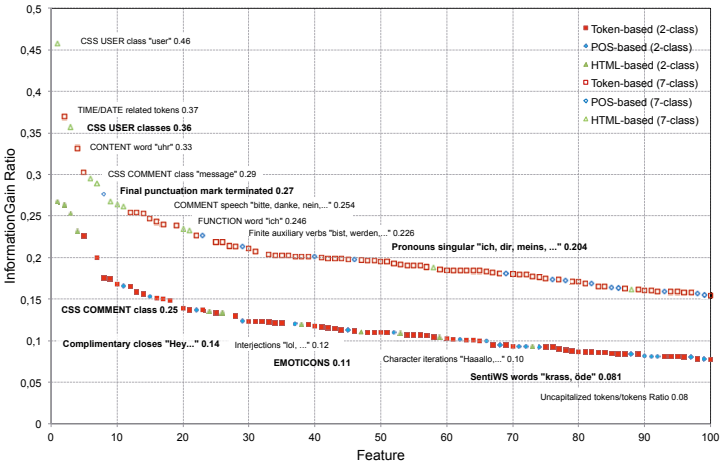


Figure 3.3: Per feature information gain ratio in decreasing order for the 2- and 7-class problem.

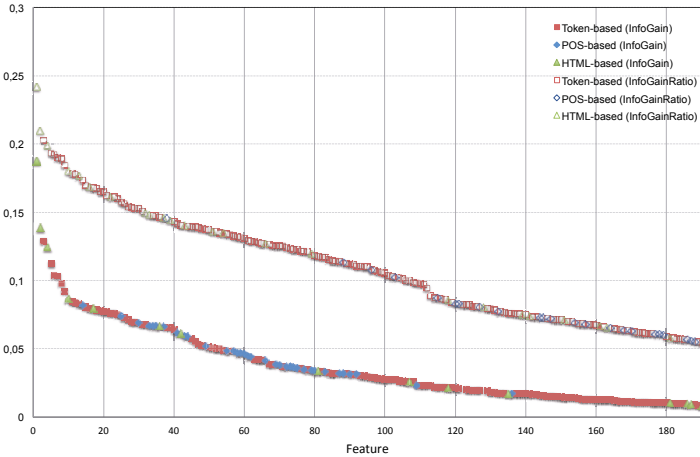


Figure 3.4: Per feature information gain ratio in decreasing order for the 2-class Web cleaning problem.

3.7.1 Social Media Text Classification

We start evaluating social media text classification. The cross validation results of social media text classification for the 2- and 7-class problem are discussed in the following and depicted in Table 3.5 and Table 3.6. We measure classification accuracy by COMMENT class precision P_{COM} , COMMENT class recall R_{COM} , mean F_1 -Score (\bar{F}_1) and total accuracy AC . The mean F_1 -Score particularly reflects performances achieved over all classes ARTICLE, TITLE, TIME, META, OTHER and COMMENT in the 7-class problem. The upper part of Table 3.5 depicts classification accuracies (2- and 7-class) achieved with the independent labeling approach for the different classifiers KNN, SVM and C4.5 trained and tested with all proposed features.

In order to analyse the influence of integrating features from predecessor and successor segments for classification in more detail, classification accuracies for $k = 3$ are depicted in addition. In the same way results for a CRF are depicted in the upper part of Table 3.6. Generally results achieved with a SVM and CRF significantly outperform the other methods. With respect to total accuracies and mean F_1 -Scores the SVM ($k=3$) reaches the highest values for the 2- and 7-class problem with 89.05% and 65.64%. Accuracies achieved on seven classes are significantly lower, however more classes have to be differentiated and hence the problem is more complex. Performances with a CRF for the 2-class problem are only slightly lower.

Integrating features from three preceding and succeeding text segments, yields mean F_1 -Scores and P_{COM} improvements up to 15 and 5 percentage points for the SVM. Consistently for all classifiers performances are improved for $k = 3$. Comparing F_1 -Scores for the SVM and CRF approach shows that values differ more for $k = 3$ compared to the $k = 0$ especially for the 7-class problem. Due to the fact that the CRF uses emission like feature functions for each combination of observed feature values combined with each class, especially for the 7-class problem model the complexity significantly increases and hence parameter estimates are less reliable performed on the same amount of training data. With the given amount of training data and the number of parameters to be estimated, on average for each parameter less than one training observation is available. Furthermore, it is worth mentioning that for $k = 0$ the KNN classifier and C4.5 decision tree significantly outperforms the SVM. A reason for that could be that SVMs are designed to solve 2-class problems. Solving classification problems considering more classes is done by combining several 2-class SVM and join the results. This could lead to a performance loss.

In order to investigate the relevance of different feature types in more detail, we exemplarily train a KNN classifier and CRF method for each feature type separately. Results are depicted in the middle part of Table 3.5 and Table 3.6. Using an approach based on token-based features outperforms the POS-based and HTML-based approach. This is consistent with our expectations. Beside high information gains ratios for such features, see Figure 3.3, the higher amount of such features contributes to better performances. However, for all feature types performances drop significantly, compared to the approach, when using all feature types in combination, see KNN ($k=0$) and CRF ($k=0$). The performance loss for the 7-class problem is worth mentioning.

The mean F_1 -Score drops by a maximum of 19 percentage points. This emphasizes the importance of combining different feature types, the more specific text classes are differentiated.

After analysing the features separately, feature combinations are evaluated in more detail. In order to investigate the combination of different features considering their redundancy, we apply a greedy correlation-based feature subset selection proposed by [30], see Section 2.5.2. For the 2-class problem 20 features combined of 35% token-based, 30% POS-based and 35% HTML-based features are selected by applying the algorithm. The amounts of different feature types are more balanced. Token-based features are for example the FUNCTION word *ich* (I), the CONTENT word *auch* (*also*) (LCn) or capitalization ratios (CAn). POS-based features are exemplarily the frequency of interjections ($ST18$), pronouns in singular form (POn) or adverb noun ratio (STn). Selected HTML-based features are for example the HTML-tag level or the occurrence of a link in the text segment. Analyzing the resulting feature subsets for the 7-class problem results in a set of 30 features combined of 47% token-based, 26% POS-based and 27% HTML-based features. The number of selected token-based features is significantly higher compared to the other types. Features such as the occurrence of date/time related tokens ($DT1$) or the frequency of digits (TOn) in the text segment are chosen. Additionally, features based on frequency counts of commas, colons and punctuation marks (PUn) are selected. Such features serve to differentiate between the additional classes considered in the 7-class problem. Overall, all selected features yield high ranks according to the information gain ratio as discussed in the previous section. Tagging accuracies achieved with the reduced set of features are depicted in the lower part of Table 3.5 (feature subset). P_{COM} rates are slightly lower, compared to the classifiers using all 245 features, however the number of features can significantly be reduced by 7/8, which reduces computational classification effort. Bringing together the previous results that (1) subset selections are a combination of all three feature types, (2) single per-feature information gain ratios are rather low and (3) classification accuracies are relatively high, shows that combining features from different types is one of the key components to achieve satisfying classification results.

Finally, we evaluate the influence of emission like feature functions with combined features as proposed in equation (3.6) for the CRF approach. Results are depicted in the lower part of Table 3.6. We apply a CRF based on a selected subset of features selected according to the correlation based measure described in Section 2.5.2. Applying this method, when the whole set of d features is used for classification would lead to a high complexity of the model and hence a high number of parameters would have to be estimated. For reliable estimates an adequately higher amount of training data is needed. Accuracies achieved by the CRF trained on the feature subset (CRF feature subset) serve as reference values in the following comparisons. Although precision and recall rates for the COMMENT class drop slightly for the 7-class problem, mean F_1 -Scores are improved by 3 percentage points for the 7-class problem. In general, combining features in that way in a CRF lead to higher mean tagging accuracies over all classes. This is also reflected in the 2-class problem differentiating between COMMENT and NON-COMMENT. Mean F_1 -Scores are improved by 1.5 percentage

points. Notable, is the increase from 63.38% R_{COM} to 67.22% for the 2-class problem. The interaction of single features in a combined way lead to a better detection of social media text segments. Total accuracies are only slightly affected. Hence, depending on the problem and application combined features are useful. It would be interesting to analyse the influence of feature functions based on a combination of all 245 features, if a sufficiently large training corpus is available.

Algorithm	d	P_{COM}		R_{COM}		F_1 -Score		AC	
		2-class	7-class	2-class	7-class	2-class	7-class	2-class	7-class
KNN (k=0)	245	70.39	69.53	62.50	63.78	80.49	59.02	91.87	82.20
KNN (k=3)	1715	72.67	70.79	76.89	77.86	85.33	64.49	93.57	86.24
SVM (k=0)	245	77.67	72.63	68.06	76.09	83.90	50.43	93.28	83.44
SVM (k=3)	1715	83.13	80.95	79.38	83.97	89.05	65.46	95.35	88.43
C4.5 (k=0)	245	72.09	71.18	69.31	68.36	82.89	59.67	92.62	83.14
C4.5 (k=3)	1715	72.33	68.79	66.81	66.63	82.23	59.01	92.49	82.65
KNN, token features	149	64.43	62.98	53.75	55.26	76.38	50.44	90.45	79.58
KNN, POS features	33	60.48	58.79	45.00	46.45	72.47	40.11	89.24	77.21
KNN, HTML features	63	57.23	56.46	47.35	48.75	72.22	42.68	89.02	74.95
KNN, feature subset	20 30	68.12	67.93	61.99	63.15	79.52	51.65	91.31	79.74
SVM, feature subset	20 30	72.96	69.16	57.18	69.25	79.06	41.97	91.62	80.77
C4.5, feature subset	20 30	72.90	68.32	63.15	66.21	80.99	56.24	92.11	81.49

Table 3.5: Cross validation results for social media classification (2-class/7-class) achieved with independent labeling approaches performed on text blocks based on a d -dimensional feature vector.

Algorithm	d	P_{COM}		R_{COM}		F_1 -Score		AC	
		2-class	7-class	2-class	7-class	2-class	7-class	2-class	7-class
CRF (k=0)	245	79.00	76.80	73.58	77.44	85.69	51.87	93.98	85.16
CRF (k=3)	1715	83.09	82.41	78.17	83.68	88.62	56.64	95.17	87.90
CRF, token features	149	74.01	73.75	66.63	75.53	82.79	48.21	92.97	83.51
CRF, POS features	33	66.87	68.44	51.11	62.34	75.84	40.05	90.56	81.65
CRF, HTML features	63	66.42	63.46	49.35	53.34	74.69	35.90	90.42	79.59
CRF, feature subset	20 30	74.91	77.23	63.38	73.15	81.38	47.21	92.50	83.94
CRF, combined features on subset	20 30	76.74	75.75	67.22	72.93	82.99	50.18	92.98	84.18

Table 3.6: Cross validation results for social media text classification (2-class/7-class) achieved with CRF approaches performed on text blocks based on a d -dimensional feature vector.

3.7.2 Web Page Cleaning

In this section we evaluate and compare the proposed Web cleaning methods to state-of-the-art methods. In addition to cross validation evaluations performed on the *GWebTrain* corpus, the Web cleaning methods are evaluated on an English benchmark corpus. Classification performance is measured by CONTENT/BOILERPLATE precision (P_{CONT} , P_{BOILER}), recall (R_{CONT} , R_{BOILER}) and mean F_1 -scores (\bar{F}_1 -Score) based on TB segments. The Web cleaning methods are particularly trained and developed for social media text platforms, i.e., Web pages containing social media texts. Hence, we additionally calculate the recall achieved on COMMENT_AREA (R_{COM_AREA}). Cross validation results are depicted in Table 3.7 for independent labeling approaches and in Table 3.8 for CRF methods. In addition to the different

classification methods, we concentrate on comparing results for different set of features.

Particularly good results are achieved by applying a CRF ($k=0$) or a SVM ($k=3$) with F_1 -Scores up to 92.52%. The KNN classifier ($k=3$) attracts attention by its high CONTENT precision, which is particularly interesting for Web corpus construction. More importantly, the CRF achieves a good balance for both precision and recall achieved on both classes. Furthermore, both CRF approaches ($k=0,3$) lead to higher recall values on the COMMENT_AREA compared to the independent labeling approaches up to 95.11%. This can be explained by the fact that the classes of the preceding and succeeding segments are good indicators if the current text segment is part of the commented area in a Web page. This dependency is particularly modeled in a CRF. However, we expect performance improvements to be higher when using a CRF modeling the dependencies between neighboring text segments (observations) and labels. We resume that for the 2-class problem CONTENT vs. BOILERPLATE under the given conditions like the training data amount and feature space dimension there is not much difference between the independent labeling approaches and the CRF. Performing the same cross validations on a bigger training corpus is expected to lead to improved results.

Additionally, we compare the results for specific classification methods based on the original feature vector ($k=0$) and extended feature vector ($k=3$). Different developments are observed. In contrast to the social media text classification results, applying the CRF approach with an extended feature space a consistent performance loss is observed with exception for the recall achieved on the COMMENT_AREA, which can slightly be improved. These results have to be analysed in more detail in future work. Nevertheless, one explanation could be the increased number of model parameters, which are not estimated reliably from the current amount of training Web pages. Especially, if the new parameters are highly correlated with existing ones and in addition contribute weak information with respect to the classes, leads to higher estimation noise. Consequently, performances decrease. Different optimizations could improve the performance: more training data, reduction of the parameters (features) to those with a high information gain or removing parameters (features) with high correlations. Applications to a bigger training corpus would help for clarification.

The middle part of Table 3.7 and Table 3.8 depicts classification performances achieved on single feature types. A detailed discussion with respect to social media text classification has been performed in the previous section. Overall results are similar for the 2-class problem CONTENT vs. BOILERPLATE and hence not further discussed in this section.

Equivalently to the evaluation performed in 3.7.2 we apply the subset selection proposed by [30]. For the Web cleaning task 23 features combined of 57% token-based, 4% POS-based and 39% HTML-based are chosen as feature subset. It is worth mentioning that CONTENT words (LCn) *antworten* (*reply*), *kommentar* (*comment*) *beiträge*, (*contribution*) are selected for the detection of commented areas in contrast to HTML-based features. This is in oppositional to the evaluation results in Section 3.7.1, where

Algorithm	d	P _{CONT}	R _{CONT}	P _{BOILER}	R _{BOILER}	F ₁ -Score	R _{COM-AREA}
KNN (k=0)	245	85.88	80.69	87.21	91.05	86.08	83.07
KNN (k=3)	1715	94.41	68.05	81.69	97.37	83.73	73.47
SVM (k=0)	245	88.37	81.41	87.91	92.80	87.44	83.32
SVM (k=3)	1715	92.24	88.95	92.63	95.06	92.12	94.66
C4.5 Tree (k=0)	245	85.00	82.75	88.42	90.23	86.49	86.19
C4.5 Tree (k=3)	1715	87.42	84.39	89.44	91.79	88.15	88.07
KNN, token based	149	85.51	74.77	84.07	91.38	83.62	77.52
KNN, POS based	33	83.89	66.15	79.67	91.47	79.45	67.04
KNN, HTML based	63	79.55	69.23	80.75	87.97	78.96	72.96
KNN, feature subset	23	82.68	77.45	85.08	88.99	83.39	79.91
SVM, feature subset	23	87.80	72.58	83.26	93.27	83.61	76.01
C4.5, feature subset	23	85.16	81.43	87.67	90.45	86.09	84.51

Table 3.7: Web cleaning cross validation results achieved with independent labeling approaches performed on article segments based on d -dimensional feature vectors.

rather HTML-based features such as the occurrence of specific CSS class names are selected. One explanation for this is that classification is performed on different segmentations, textblocks vs. article segments. Article segments often comprise whole areas of comments instead of single comments, hence CSS class names of the contained elements are not considered in the feature calculation. The result is that tokens such as *antworten (reply)* or *kommentar (comment)* become valuable features for the detection of commented areas.

All classification methods are trained using only the determined subset of features. Results are depicted in the lower parts of Table 3.7 and Table 3.8. With respect to the mean F_1 -Score over the considered classes, for all classification methods performance increases. This is consistent with the results achieved for social media text detection. However, performance losses are slightly lower between 0.5 and 4 percentage points for all methods. For similar reasons explained in the previous section, applying the decision tree again leads to the smallest decrease of about 0.5 percentage points. Considering the differences in the recall of the COMMENT_AREA shows performance losses in the same range or even higher compared to mean F_1 -Scores. This reveals that discarded features are more important for the correct classification of commented areas. However, overall a small accuracy decrease in combination with a significant reduction of feature space dimension of about 7/8, yields a good trade-off between classification accuracy and complexity.

Similar to social media text classification evaluations, we evaluate the influence of emission like feature functions with combined features according to equation (3.6) for the CRF approach based on feature subsets. Results are depicted in the lower part of Table 3.8. As reference values we choose performances achieved with the CRF trained on the feature subset (CRF feature subset) for the following comparisons. Accuracies achieved with subset selection serve as reference values in the following comparisons. Precision and recall rates for the CONTENT class are improved by 2 and 3 percentage points. This bears in relation to the improvement achieved for the COMMENT class considered in the social media text detection problem discussed in the previous section. Social media texts are part of the CONTENT class in the context of Web cleaning. Hence, improvements achieved by combined features are reflected by precision and

recall rates for the CONTENT class. Overall mean F_1 -Scores and total accuracies are only slightly increased in the context of Web cleaning.

Algorithm	d	P _{CONT}	R _{CONT}	P _{BOILER}	R _{BOILER}	\bar{F}_1 -Score	R _{COMAREA}
CRF [†] (k=0)	245	91.95	90.12	93.44	94.71	92.52	95.04
CRF [†] (k=3)	1715	91.57	89.98	93.43	94.52	92.33	95.11
CRF, token features	149	91.72	88.20	92.24	94.71	91.63	94.12
CRF, POS features	33	91.31	81.85	88.67	94.77	88.92	88.97
CRF, HTML features	63	91.33	76.89	86.03	95.07	86.65	86.13
CRF, feature subset	23	92.53	86.71	91.72	95.27	91.41	94.39
CRF, Combined feature subset	23	90.79	89.51	92.96	93.80	91.70	94.50

Table 3.8: Web cleaning cross validation results achieved with CRF performed on article segments based on a d -dimensional feature vector.

Finally, we compare the proposed Web cleaning methods to state-of-the-art methods. We depict *GWebTrain* cross validation results achieved with methods provided by Kohlschütter’s tool in Table 3.9. Comparing our results to those achieved by Kohlschütter shows that F_1 -Scores can be increased up to 25 percentage points compared to the best result achieved with a CRF (k=0) approach. Differences between our and Kohlschütter’s methods are statistically significant according to a corrected re-sampled paired t -test [48] applied to the two cross validation with a significance level of $p = 0.005$. The CONTENT recall for the LARGEST CONTENT Extractor is the highest compared to the other two methods. However, at the same time CONTENT precision rates drop significantly. This coincides with the principal of this approach aiming at classifying rather a large area as CONTENT. However, recall rates are still significantly lower, compared to our approaches. The ARTICLE Extractor follows the opposed principal following the idea to extract only the article from a Web page, i.e., rather a smaller content area. Hence, precision rates reach values in the range of our methods but recall rates drop significantly. All approaches suffer from bad recall rates on the COMMENT_AREA. Recall rates drop down by 50 percentage points or more compared to our CRF (k=3) approach. Due to the missing social media text specific features and adequate training data, such methods tend to classify social media text areas as BOILERPLATE.

Algorithm	P _{CONT}	R _{CONT}	P _{BOILER}	R _{BOILER}	\bar{F}_1 -Score	R _{COM}
LARGEST CONTENT Extractor	80.21	49.13	68.06	93.37	66.63	43.45
DEFAULT Extractor	91.38	31.36	64.33	98.72	58.69	16.89
ARTICLE Extractor	93.64	21.17	62.05	99.34	51.99	7.28

Table 3.9: Web cleaning cross validation results for three exemplary state-of-the-art methods.

Figure 3.5 illustrates the result for an example Web page with our approach compared to two approaches proposed by Kohlschütter, (1) our approach using a SVM with k=3 (red), (2) the DEFAULT extractor (black), (3) ARTICLE extractor (dotted black). Detected CONTENT parts of the Web page are depicted by the particular box. The two Kohlschütter’s extractors do not even detect any or only small parts of the comment area as CONTENT. Our classifier successfully detects the complete

COMMENT_AREA except for the headline. However, some keywords like *Blogsuche* (*Blogsearch*) or *Neuste Postings* (*recent posts*) are wrongly detected as CONTENT. This can be explained by the fact that these keywords are strongly related to commented areas and in other Web pages are part of the content. Moreover, these are very small areas.



Figure 3.5: Exemplary Web cleaning results for different approaches, red: our approach, dotted black: article extractor, black: default extractor.

Micro-average	CRF (k=3)	SVM (k=3)	LCExtractor	AExtractor	NCleaner
Mean F_1 -Score	0.84	0.90	0.69	0.81	0.90
Mean Precision	0.95	0.93	0.97	0.96	0.90
Mean Recall	0.75	0.86	0.54	0.70	0.90

Table 3.10: Evaluation of different methods applied to the *CleanEval* corpus.

Application to English Benchmark Corpus

It is important to us to show that the Web cleaning methods are portable to English Web pages and to test the domain-independence of the classifiers. In the previous section the English Web cleaning benchmark corpus *CleanEval* has already been introduced. For the benchmark, we train the same classifiers on our manually annotated English Web page corpus *EWebTrain*. We use the same set of features, but adapt the calculation of language-dependent features. In particular, calculations of token- and POS-based features are adapted by using English word lists and integrating a POS tagger for English.

We evaluate the 2-class problem CONTENT vs. BOILERPLATE on the *CleanEval* test set for two different classifiers that have been trained on the *EWebTrain* corpus and for the ARTICLE and LARGEST CONTENT extractor proposed by Kohlschütter. Exemplarily, we choose a CRF and a SVM classifier both with $k = 3$, i.e. considering features of the k preceding and succeeding segment features. Note that the *CleanEval* corpus only contains a few Web pages containing social media texts and no specific annotation of the commented areas. Therefore, no explicit evaluation for such Web page types is possible. Because the *CleanEval* corpus only provides content annotation on a text-level, we cannot directly use the evaluation setup applied to the *GWebTrain* corpus. In order to produce comparable results to existing approaches, we use the python script *cleaneval.py* proposed in [18] and calculate mean precision, recall, and F_1 -Score based on token level. Note that the accuracy measure is based on a weighted Levenshtein distance at token-level proposed by the *CleanEval* initiative and, hence, is not directly comparable to the results depicted in Table 3.9.

Results are depicted in Table 3.10 in the same form than published in [18]. For a direct comparison, the published results of *NCleaner*, [18], trained on the *CleanEval* training set are depicted in the right column of the table. Similarly to the cross-validation results, the CRF and SVM based on article segments outperform the results achieved with Kohlschütter’s extractors. However, in contrast to the previous results, the SVM based on article segments significantly outperforms the CRF based on text blocks and hence seems to be more robust against the data basis. Note that the *EWebTrain* corpus is significantly smaller than the German *GWebTrain* corpus. According to the higher complexity of the CRF model parameter estimates are less reliable. Experiments on the *GWebTrain* corpus have shown that enlarging the amount of training Web pages from 50 to 180 leads to a F_1 -Score increase of 1.5 and 4.2 percentage points for SVM and CRF. Improvements are much more higher for the CRF classifier. Hence, a significant improvement with the CRF classifier is expected, when an adequate amount of training data is available. A final comparison of our classifiers to *NCleaner* shows

that we can still compete with the results achieved by the *NCleaner* method, which has specifically been trained for *CleanEval*. Overall, this shows that the classifiers trained on the proposed Web page corpora can be applied to Web page corpora from other domains without significant performance loss. Furthermore, the methods are portable to English Web pages with little effort.

3.8 Conclusions

In this chapter, we have proposed methods for social media text classification and detection, and methods for Web page cleaning designed for Web pages hosting social media platforms, where sequences of Web text segments are classified based on a high-dimensional feature vector. Good classification performances for both tasks are particularly achieved by providing a representative training corpus consisting of Web pages from social media platforms and an adequate combination of token-, POS- and HTML-based features. This is substantiated by the results of a detailed feature analysis, where novel social media text related token- and HTML-based features yield high information gain ratios.

First, an approach for social media text (comment) detection in Web pages has been presented. In addition to the resulting 2-class problem a more fine-grained classification problem considering seven classes comprising meta informations is introduced. Social media texts which are of particular interest, are detected with highest precision rates of 83%, when applying a SVM to the 2-class problem with a feature vector extended by features of three preceding and succeeding text segments ($k=3$). For the more fine-grained classification, where several meta informations such as the user names and the posting times are considered as classes, mean F_1 -Scores up to 65.5% applying the SVM classifier are achieved. This is significantly lower compared to the 2-class problem with 89.1% mean F_1 -Scores, however this a biased comparison, since the differentiation between seven classes is more complex. Reasoned by the relatively small training corpus the SVM classifier outperforms the CRF ($k=3$) approach particularly for the 7-class problem. With the current training corpus size the SVM classifier should be chosen. However, with a bigger training corpus we expect, that the CRF modeling the dependencies between consecutive Web text segments would outperform the SVM classifier.

Second, we have proposed an effective approach for Web cleaning applied to social media text platforms, i.e., Web pages containing social media texts. Significant improvements compared to state-of-the-art approaches are achieved by providing a representative training corpus and the usage of a combined feature set with token-, POS- and HTML-based features comprising features related to social media text characteristics. The proposed methods achieve mean F_1 -Scores up to 92.5% applied to our Web page corpus hosting social media text platforms. This significantly outperforms existing approaches by a minimum of 26 percentage points. Applying a Conditional Random Field yields the highest recall of 95.1% on the COMMENT_AREA, which is of special interest in our task. Furthermore, we analyse our Web cleaning methods on

the English benchmark corpus *CleanEval*. Results show that the extended feature sets are domain-independent and can be transferred to other languages with little effort. Satisfying results are achieved on the *CleanEval* evaluation corpus, even when trained on small training corpora.

4 POS Tagging for Social Media Texts

In this chapter, we consider the problem of social media text *Part-of-Speech*(POS) tagging, which is one of the most fundamental sequence labeling tasks in NLP. Besides sentiment analysis, NLP methods such as syntactical parsing, machine translation or text summarization require POS tag information for text to be processed. State-of-the-art POS taggers are basically developed for tagging standardized texts such as newspaper articles which are grammatically approved. Hence, parameter estimation is usually performed on newspaper text corpora as training data. POS tag information can be achieved by automatic taggers with accuracies up to 98% for such standardized texts.

Social media texts, however, differ from standardized text since they are characterized by a spoken language, a dialog and an informal writing style. Applying state-of-the-art taggers to such non-standardized text types leads to a significant performance loss. Developing automatic POS tagging for social media texts poses some special challenges to deal with. In particular these are the treatment of unknown (out-of-vocabulary) words and the different grammatical structure of social media texts in contrast to newspaper text. Furthermore, manually annotated in-domain corpora, i.e., social media texts are required for training and testing. Hence, the problem of POS tagging of social media texts can not be solved adequately by standard methods so that more sophisticated methods are needed.

This thesis proposes a Markov model tagger called WebTagger with parameter estimation enhancements for the POS annotation of social media texts. We particularly improve the parameter estimation for unknown tokens (words) in several ways. To enhance existing approaches, probability estimation methods are extended by mapping unknown tokens to tokens known from training or to some token classes represented by regular expressions. For still unknown tokens, we propose a semi-supervised verb and domain specific auxiliary lexicon instead of information from automatically tagged or unsupervised training data, beside different combination methods for tokens' prefix and suffix tag distributions. Furthermore, we consider the different grammatical structure of social media and newspaper texts leading to diverse distributions of POS tag sequences. In contrast to existing POS tagging approaches, we propose to combine a social media training corpus and a newspaper corpus by an efficient oversampling of the in-domain training data. We experimentally evaluate the proposed methods for a German social media text corpus and different social media text types. Results are compared to widely used state-of-the-art POS taggers. The proposed approach

outperforms state-of-the-art POS taggers significantly for German social media texts. We show that by applying our extended Markov model tagger to an existing social media text corpus we are able to obtain accuracies of up to 95%, which comes close to accuracies achieved on standardized newspaper texts. Even when applying WebTagger to social media text types, where the tagger is not trained on the particular type, leads to a significant performance increase. Finally, we show that including grammatically non-standardized social media texts in the training data does not negatively affect tagging accuracies of standardized texts by means of the proposed approach.

In addition to the WebTagger, a new social media text corpus for training and test purposes is introduced. As a tag set we use the 54 *Stuttgart Tübinger* (STTS) tag classes without any text genre specific extensions in order to make our method usable for NLP methods requiring POS information without any adaptations. Hence, an extended annotation guideline for social media text specific characteristics based on the STTS tags is proposed. Furthermore, text tokenization as fundamental initial step for POS tagging is addressed in this thesis. This is a non-trivial step in the context of social media texts. However, since it is not the focus of our work it is only touched upon briefly.

The outline of this chapter is as follows: Section 4.1 summarizes related work about automatic POS tagging in general with a particular focus on approaches dealing with non-standardized texts. In Section 4.2 we briefly discuss social media text tokenization challenges and propose our rule based tokenization algorithm for adequate tokenization. Section 4.3 suggests an extended annotation guideline for social media text specific characteristics and introduces the used STTS tag set. In Section 4.4 we introduce our basic tagger model and propose our adapted parameter estimation methods for lexical probabilities in Subsection 4.4.1 and 4.4.2, as well as parameter estimation methods for transition probabilities in Subsection 4.4.3. Furthermore, Subsection 4.4.4 introduces a joint-domain training approach applying oversampling to further improve tagging results. In Section 4.5 we introduce the new social media text training corpus, which is manually annotated according to the annotation rules outlined in Section 4.3. In addition to our main corpus additional newspaper and social media text corpora and their corpus statistics are proposed in this section. Section 4.6 presents experimental results considering different aspects and particularly discusses the usability for different social media text types. Section 4.7 covers the conclusion and discusses future work.

4.1 Related Work

A variety of different approaches has been proposed to solve the task of automatic POS tagging. Over the last years the majority of proposed methods are probabilistic ones. However, a number of rule-based methods have been proposed in the early stages of POS tagging research. In [35] the first rule-based approach is presented which is based on a computation grammar coder for English POS tagging, differentiating between 30 POS classes. Rules are based on spelling, capitalization and specific tests on the words'

prefixes and suffixes. One of the most recent rule-based methods is proposed in [9], yielding similar accuracies as modern probabilistic methods. Nowadays, statistical POS taggers utilize two different probabilistic models: A Maximum entropy model or a Markov model capturing lexical and contextual information.

Common maximum entropy based taggers are proposed in [67, 83, 84, 24]. The proposed methods use the same baseline maximum entropy model and adapt their approach by using different features in the model in order to enrich the information source for tagging. Characteristic feature functions are binary valued and contain lexical information, e.g., a word's capitalization, spelling, suffix/prefix, contained numerals or hyphens. Toutanova et al. [84, 83] proposed the Stanford tagger, which is the most popular maximum entropy model based tagger for German and English. It models the sequence of words as bi-directional dependency network considering lexical and tag context information. In addition to the lexical information based feature functions mentioned before, feature functions of the two preceding and succeeding POS tags are integrated into the model. Tagging accuracies of up to 97% are achieved on English newspaper texts.

Markov model taggers are proposed in [74, 75, 8]. TreeTagger [75] and TnT [8] use a 2-order Markov model and apply smoothing techniques for the estimation of lexical probabilities in order to achieve reliable estimates for rarely seen words. TreeTagger is the most commonly applied Markov model tagger for German language texts. It distinguishes itself from other models by using a binary decision tree to estimate tag transition probabilities reliably. Further extensions to the basic model from [74] considering letter capitalization are proposed in [75] for German texts. Applying TreeTagger to German standardized newspaper texts leads to accuracies of up to 97.5%. Both taggers are developed in a very general way so that they can be trained on corpora of different languages. The German version of the taggers uses the STTS [72] tag set, which is commonly used for NLP methods.

In addition to the approaches mentioned lately, some other machine learning techniques such as Support Vector Machines (SVMs) [26] or Neural Networks [73] are applied to the problem of automatic POS tagging. Gimenez et al. [26] train a SVM tagger based on the same features used in the Stanford tagger. Competitive results are achieved for English and Spanish texts. Based on the same lexical and transition features used by Schmid [74], the author additionally trains a Neural Network tagger. Achieved results are slightly worse compared to the Markov model approach.

Performance investigations of the discussed state-of-the-art newspaper text taggers show that automatic POS tagging of non-standardized social media texts results in significant accuracy drops. In [90, 25] different POS taggers are compared and evaluated for German texts. Schneider et al. [90] point out the performance loss of a rule-based tagger for unknown words compared to a statistical tagger. Five state-of-the-art taggers applied to Web texts are studied in [25]. Accuracy drops are shown to be significant for various Web text genres, a result which serves as the main motivation to develop a specialized POS tagger for Web texts such as social media texts in the context of this thesis.

In [47, 76] two approaches are developed which particularly deal with the annotation of texts with a high number of unknown words, however no focus is put on social media texts. They suggest to use prefix and suffix information as applied in many state-of-the-art taggers recently used. However, they do not specifically address the task of social media text tagging, where tokens are unknown not only due to small lexica but rather due to a high degree of non-standardization. Recent, publications [24, 60, 61, 68] address these problems and deal with non-standardized texts such as Twitter messages, short message service (SMS) or online conversational texts. Most of the proposed approaches use CRFs, with different feature functions related to social media text characteristics. A variety of publications particularly deals with POS tagging for Twitter messages which exhibit even more specific characteristics such as hashtags, @-mentions or the fact that only a very limited number of characters (140) is used compared to other social media texts.

In [24], one of the few approaches dealing with social media texts different from Twitter messages is proposed. It introduces feature modifications to the Stanford maximum entropy tagger to handle noisy English text. Results are evaluated based on an SMS dataset. The following approaches all deal with Twitter messages. In [27] a Twitter tagger based on a CRF with features adapted to Twitter characteristics is proposed. The authors propose some additional word clustering and further improvement to their method in [61] and evaluate their approach on different English Twitter data where they achieve a maximal accuracy of 92.8%. Rehbein [68] proposes a CRF based POS tagger for German Twitter messages based on the work proposed in [27]. Applying word clustering with features extracted from an automatically created dictionary leads to 89% tagging accuracy which is slightly lower than results achieved for English Twitter data. In [14], one of the latest Twitter taggers is proposed. A vote-constrained bootstrapping on a large corpus of unlabeled Twitter messages is performed to create tweet-genre training data. By means of methods for handling the genre characteristic errors and slang in addition to adjusting prior tag probabilities of unambiguous tokens, tagging performance is further improved. All mentioned Twitter approaches make use of the strong restrictions and specific characteristics of such texts in the way features are constructed, e.g., the occurrence of hashtags (#) or @-mentions. Furthermore, the tag sets used are extended by tags for these particular text phenomena. As a result, it is not straightforward to transfer the proposed methods to general social media texts. Hence, a more general method is needed in order to handle social media texts such as blog comments, chat messages or any kind of Web comments. In order to develop a POS tagger which is usable for subsequent NLP methods without adaptations, an annotation guideline for social media text characteristics based on an existing tag set needs to be defined.

Finally, we briefly address related work for tokenization. Adequate tokenization is a fundamental step for automatic POS tagging. Typically, state-of-the-art taggers provide a tokenizer which do not enable for adequate tokenization when applied to non-standardized texts. Tokenization of non-standardized text in general is addressed in [63, 38, 27]. Gimpel et al. [27] propose a Twitter tokenizer for the task of POS

tagging. In [63] a sentiment tokenizer is developed which enables for adequate tokenization of emoticons.

4.2 Tokenization

Tokenization is a fundamental initializing step for POS tagging, where the text is divided into coherent units (tokens). Each token is either a word or something else, e.g., a punctuation mark. For standardized grammatically approved text, tokenization can be performed independently from POS tagging, since the tokenization is unambiguous. Typically, a simple white space tokenizer considering the German punctuation rules provides adequate text decomposition. In contrast to that, tokenizing social media texts is a more challenging task. We aim at developing a tokenizer for social media texts, which allows for adequate POS tagging annotation, i.e., STTS annotation. Particularly, phenomena from *social media language* and *informal writing style* categories, such as emoticons, merged words, or multiple punctuations claim for a more sophisticated tokenizer. For example, an emoticon at the end of a sentence without any whitespace, e.g., *ready;-)*, a cardinal number merged with a particular currency, e.g., *3\$*, needs to be split to enable for assigning unique POS tags.

Two tokenization approaches can be differentiated: A statistical approach based on classification and a rule-based approach. We choose a rule-based approach. Rules are designed and adapted to the manually tokenized data. In total 32 criteria, realized by regular expressions, are used to detect coherent tokens such as emoticons, URLs, dates or filenames. Furthermore, we use additional lexical information in order to enhance the tokenization performance. Moreover, we utilize an abbreviation list, introduced in [75], containing approximately 6,700 typical German abbreviations, e.g., *Dr.* and *bzw.*, to detect coherent tokens. Additionally, a merged word list containing token pairs, which are frequently merged to one word in social media language is generated. A typical example is merging a verb with an pronoun, e.g., replacing *geht es (does it)* by *geht's (does't)* or *gehts (doest)*. The list is conducted based on the social media text corpus *WebTrain*, see Table 4.3, by selecting the 90 most frequently merged token pairs. In total all token pairs occur 11200 times, which illustrates the importance to consider such token pairs in the process of tokenization. Merging both words is grammatically incorrect. Hence, a set of rules is designed to split such tokens, which enables for the assignment of the corresponding POS tag to each of the two tokens. Since, the second part of the token is rather ambiguous in the word itself nor in its POS tag, e.g., the origin of the merged word *ob's (if't)* can be *ob es (if it)* or *ob das (if that)*, it can not be replaced by its original two words. Hence, for all merged words we split the added letter with apostrophization, following the tokenization of English texts, e.g., *obs* or *ob's* is replace by the two tokens *ob* and *'s*.

A description of the tokenization is given in Algorithms 2 and 3. Algorithm 2 splits the text string \mathbf{t} of a social media texts (SMT) at white spaces. Then, each substring st is processed in $\text{APPLYCRITERIA}(st)$ in order to split words which have been merged. All of the resulting words $\hat{\mathbf{w}}$ are concatenated to a vector of words \mathbf{w} and returned to

the tagger which is discussed in Section 4.4. The principles of $\text{APPLYCRITERIA}(st)$

Algorithm 2 $\text{SMTOKENIZATION}(\mathbf{t})$

Input: SMT text string \mathbf{t}

Output: Token vector \mathbf{w}

```

st  $\leftarrow$   $\text{SPLITTEXTATSPACES}(\mathbf{t})$ 
w  $\leftarrow$  ()
for each string  $st \in \mathbf{st}$  do
     $\hat{\mathbf{w}}$   $\leftarrow$   $\text{APPLYCRITERIA}(st)$ 
    w  $\leftarrow$  (w,  $\hat{\mathbf{w}}$ ) ▷ Concatenate tokens
end for
return w

```

Algorithm 3 $\text{APPLYCRITERIA}(st)$

Input: SMT text substring st

Output: Tokens $\hat{\mathbf{w}}$ according to criteria

```

if  $\text{EXACTCRITERIAFOUND}(st)$  then
    return st
else
    if  $\text{SPLITCRITERIAFOUND}(st)$  then
         $(st_l, st_r) \leftarrow \text{GETSPLITS}(st)$ 
        return
            ( $\text{APPLYCRITERIA}(st_l)$  ,  $\text{APPLYCRITERIA}(st_r)$  )
    else
        return st
    end if
end if

```

shall be illustrated by the exemplary input $st = \text{Hello};-))$. This example is merged from the word *Hello* and the emoticon $;-))$ which shall be decomposed. First, with the defined regular expressions it is tested if a known non-composed token, e.g., emoticon, date or punctuation iteration is existent. For our example this is not the case. Thus, it is checked if it is merged and, hence, may be split again. Obviously, this is the case and it is split into $st_l = \text{Hello}$ and $st_r = ;-))$. Both substrings are now processed recursively and the return values are concatenated. The substring *Hello* has neither an exact match nor is it identified as merged word such that this substring is returned unmodified. The substring $;-))$ is now recognized as emoticon, i.e., an exact criterion has been found. Consequently, this substring is returned unmodified as well. Overall $(\text{Hello};-))$ is returned for concatenation in Algorithm 2. The recursive nature of $\text{APPLYCRITERIA}()$ allows for splitting substrings with more than two tokens.

4.3 Extended Annotation Rules

Before automatic POS tagging can be applied, the particular POS tags/classes need to be defined, i.e., a particular tag set needs to be chosen. According to the part-of-speech granularity different tag sets are available. For German the STTS tag set, [7], is the most commonly used in the NLP community and hence is also used in our work. It was developed 1999 in Stuttgart and has evolved over the years to the standard tag set for the morphosyntactic annotation in German; it provides information about the respective part of speech and its syntactic function and differentiates between 54 POS classes. The POS classes can be further differentiated into open and closed classes, which becomes relevant in the later approach. Open classes are nouns, verbs, adjectives and adverbs which do not consist of a finite number of words and acquire new members constantly, while closed classes consisting of pronouns and conjunctions are finite and well-established word sets.

The STTS tag set was developed for the annotation of standardized texts. Until now, no extension for the annotation of the special characteristics of social media texts, e.g., emoticons, is present. Moreover, an extension of the existing tag set is problematic from a technical perspective, since existing NLP methods, e.g., syntactical parsing, require STTS POS tag information. Thus, the existing STTS tag set is used and social media text characteristics are tagged according to their syntactic function in our approach. For instance, emoticons are either at the end of a sentence or at intermediate positions. Therefore, they obtain the tag for sentence final \$. and sentence internal \$(character. Contrarily, special characters and enumerations are only annotated with the internal character tag. Separated particles of apostrophization, e.g., ([*hab*]'s - *have it*), are tagged for verbs, conjunctions, and interrogative pronouns as *irreflexive personal pronoun* (PPER), *substituting demonstrative pronoun* (PDS), or *article* (ART). Numbers replaced by the corresponding digit in a word are annotated as *attributive adjective* (ADJA) or *proper noun* (NE), depending on the context. The overall annotation rules for particular social media characteristics are given in Table 4.1. All tags from the first column can be assigned according to the given grammatical context. Exemplary tokens are given in the last column. Note, that the text is manually tok-

Tag	Description	Example
\$. , \$(Emoticons	:-) , (*_*)
NE	File names, URLs	test.jpg , www.test.de
ITJ , PTKANT	Interaction words, inflectives	lol , seufz , yep
\$(Special characters	#, @, *, i. ii., a) b)
\$(, \$.	Multiple punctuations	... , !?!
PPER , ART , PDS	Apostrophization	[geht]'s , [wer]'s , [ob]'s
ADJA , NE	Number replacement	10er , 500er

Table 4.1: Annotation rules for social media texts.

enized such that adequate POS annotation can be performed according to the given rules. A more detailed explanation of the extended annotation scheme can be found in [86, 85].

4.4 Markov-Model Tagger

As a basic tagger model we use a Markov model. Our proposed WebTagger has much in common with the TreeTagger, [75], however is adapted to social media text POS tagging. The taggers mainly differ in the way lexical (emission) probabilities are estimated. particularly for unknown words.

In general the aim of the tagger is to predict the associated POS tag sequence $t_1, \dots, t_n, \dots, t_N$ with $t_n \in \mathcal{T}$ (STTS) for a given sequence of tokens $w_1, \dots, w_n, \dots, w_N$ with $w_n \in \mathcal{W}$, where \mathcal{W} contains all possible tokens. In order to achieve that the Bayesian decision rule is applied, resulting in the optimization problem

$$\hat{t}_1^N = \arg \max_{t_1^N} P(t_1^N, w_1^N)$$

using the notation t_l^n for a sequence of POS tags

$$t_l^n = \begin{cases} (t_l, \dots, t_n) & 1 \leq l \leq n \leq N \\ (t_1, \dots, t_n) & l \leq 0 \end{cases}$$

with $l \in \mathbb{Z}$, $n \in \mathbb{N}$, and $l \leq n \leq N$ is solved. The sequence of tokens w_l^n is defined analogously. This optimization problem is simplified by the following approach. First, the probability chain rule for w_N and t_N to describe the joint probability by conditional probabilities is applied:

$$P(w_1^N, t_1^N) = P(w_N | w_1^{N-1}, t_1^N) P(t_N | w_1^{N-1}, t_1^{N-1}) P(w_1^{N-1}, t_1^{N-1}). \quad (4.1)$$

Furthermore we use the assumptions

$$\begin{aligned} P(w_n | w_1^{n-1}, t_1^n) &= P(w_n | t_n), \\ P(t_n | w_1^{n-1}, t_1^{n-1}) &= P(t_n | t_{n-k}^{n-1}) \end{aligned}$$

with $k \in \mathbb{N}$. Applying those assumptions, a simple law of conditional probability, and iterating the procedure described in (4.1) leads to the equation:

$$P(w_1^N, t_1^N) = \prod_{n=1}^N \frac{1}{P(t_n)} \underbrace{P(t_n | w_n)}_{\text{Lexical Prob.}} P(w_n) \underbrace{P(t_n | t_{n-k}^{n-1})}_{\text{Transition Prob.}}$$

The assumptions are also referred to as k -order Markov model for transition probabilities and zero-order Markov model for the emission probabilities. Moreover, the token

probabilities $P(w_n)$ do not change with the tag sequences, and hence, may be omitted. Overall, this allows to model transition and lexical probabilities independently and the optimization task is rephrased as

$$\hat{t}_1^N = \arg \max_{t_1^N} \prod_{n=1}^N \frac{P(t_n | w_n)}{P(t_n)} P(t_n | t_{n-1}^n).$$

Before the tagger can be used for predicting POS tag sequences, i.e., solving the optimization problem given in equation (4.4), probabilities have to be estimated from a manually annotated training corpus

$$\mathcal{TR} = \left\{ (\tilde{w}_n, \tilde{t}_n) \mid 1 \leq n \leq \tilde{N} \right\}, \quad (4.2)$$

where for each word \tilde{w}_n the correct tag \tilde{t}_n is known. In other words, \mathcal{TR} represents a training text where the tag sequence is known for each sequence of words. Generally, this training text can additionally be given in the form of a full form lexicon. The lexicon is given as

$$\mathcal{L} = \left\{ \tilde{w}_i \mid 1 \leq i \leq \tilde{N} \right\}$$

where $|\mathcal{L}| \ll \tilde{N}$ holds, due to the characteristics of constantly repeating words in written texts. *Full form* means that for each word $\tilde{w}_i \in \mathcal{L}$ in the lexicon the corresponding set of possible tags \mathcal{T}_i is known.

Probability estimation is performed based on the training corpus. This comprises the estimation of lexical probabilities as explained in Section 4.4.2 and the estimation of transition probabilities, see Section 4.4.3. We basically use supervised learning methods, but extend them by some semi-supervised techniques. Before the different estimation methods are explained token preprocessing is described which aims at *normalization* of the non-standardized social media training and test corpus.

4.4.1 Token Normalization

Applying token preprocessing basically aims at achieving more reliable probability estimates by a *normalization* of the text. Considering, e.g., the token *überrascht* (*surprised*), which might occur in non-standardized forms in social media texts, e.g., as *überraaaascht* or *ueberrascht*. For this example without a *normalization* for the non-standardized forms, lexical probabilities are estimated separately or if one of the forms is only present in the test data it would be treated as unknown. Normalizing all forms to one unique form would obviously lead to more reliable estimates and better tagging accuracies. This normalization would be ideal if performed manually and it would include proper spelling corrections. However, the training data should comprise the same characteristics as the test data. Since manual corrections of the test data before tagging is no adequate solution, an automatic preprocessing step is suggested in this work. We expect best tagging accuracies when token preprocessing is performed

on both training and test data and no further manual corrections of the training data are performed.

Before describing the preprocessing in more detail we give a short note on case sensitivity in the data. On the one hand side in social media texts capitalization is rarely complied, on the other side using upper case letters for the whole word, e.g., *TOLL (NICE)*, is a frequently used tool to emphasize someone’s sentiment. Both phenomena lead to a higher variety in a similar token sequence when considering case sensitivity. Therefore, a mapping to a unique version is reasonable. However, in contrast to the approach in [75], we believe that considering case sensitivity at the beginning of a token leads more reliable lexical probability estimates. For that reason, we convert all upper case letters to lower case ones except at the beginning of a sentence.

In general, the preprocessing aims at mapping non-standardized tokens to related known tokens if there exists an appropriate one. Related tokens can be obtained by some transformation steps described by $s(w_n)$. These steps contain cross-language transformations, as well as some transformations specific to the German language, and basically treat tokens of the category *Informal writing style*. Among others, character iteration corrections, e.g., Helloooooo \rightarrow Hello, or *Umlaut* correction, e.g., Huette \rightarrow Hütte (cottage), character corrections, e.g., Kuss \rightarrow Kuß (kiss) or compound words, which are simply mapped to their second part token, e.g., Heise-Daten \rightarrow Daten (Heise data). Such transformations may be interpreted as substituting tokens by their normalized version. Therefore, we refer to this kind of transformation as *normalization*. Furthermore, there are language independent *word classes* which are easily recognized and anticipated using regular expressions. Some examples include emoticons, e.g., :-), and :/, and URLs, e.g., <http://www.test.de>, xy.ch, multiple punctuation marks, e.g., and !!! and number replacements, e.g., 50er (fiftieths). The set of possible POS tags for each word class differs from one to three.

In summary, our preprocessing step substitutes unknown tokens by its transformation if it is within the training set returns the regular expression r if the word is described by it, or returns the marker for unknown tokens ϵ . This procedure is described by the mapping function $m : \mathcal{W} \rightarrow \mathcal{X} \cup \{\epsilon\}$ which is defined as

$$m(w_n) = \begin{cases} /r/ & w_n \in \mathcal{W}_r, \\ w_n & w_n \in \mathcal{L} \setminus \mathcal{R} \\ s(w_n) & s(w_n) \in \mathcal{L} \setminus \mathcal{R} \wedge w_n \notin \mathcal{L} \setminus \mathcal{R} \\ \epsilon & \text{otherwise.} \end{cases}$$

An overview of the corresponding word sets is given in Table 4.2. The word set \mathcal{X} contains all tokens given by the full form lexicon \mathcal{L} created from supervised training data, extended by the set of words \mathcal{R} created by all regular expressions $r \in R$ as follows.

$$\mathcal{W}_r = \{w \in \mathcal{W} \mid w \sim /r/\}$$

indicates all tokens covered by a regular expression $r \in R$, thus $\mathcal{R} = \bigcup_{r \in R} \mathcal{W}_r$.

Word Set	Description
\mathcal{R}	Tokens covered by regular expressions
\mathcal{L}	Full form lexicon created from training data
$\mathcal{X} = \mathcal{L} \cup \mathcal{R}$	Full form lexicon extended by regular expressions
\mathcal{W}	All possible tokens

Table 4.2: Word Sets.

4.4.2 Lexical Probability Estimation

By applying the described preprocessing, a satisfying amount of unknown tokens can be mapped to known tokens. However, the number of unknown tokens is still higher compared to standardized texts. Therefore, adapting estimation methods for lexical probabilities of unknown tokens (words), enables improvements for tagging social media texts. In this section, we consider several cases and propose a corresponding estimation method.

First, we assume lexical probabilities to be position independent. For brevity, we therefore write $P(t_n | w_n) = P(t | w)$ in the following. If the word $m(w)$ is known, i.e., it occurs in the training set \mathcal{TR} , the estimation is given by

$$\hat{P}_L(t | w) = \frac{|\{i \mid (\tilde{t}_i, \tilde{w}_i) = (t, w)\}|}{|\{i \mid \tilde{w}_i = w\}|},$$

where the index L indicates that the word w is in the lexicon \mathcal{L} . Note that we generally distinguish between the same word uncapitalized vs. capitalized and hence the $\hat{P}(t | w)$ are also different. However, if $w \notin \mathcal{L}$ holds for the original word, but by swapping the first letter from upper/lower to lower/upper $w_{\text{swapped}} \in \mathcal{L}$ holds, then the lexical probability distribution $\hat{P}_L(t | w_{\text{swapped}})$ is used.

In the following the estimates if the word $m(w)$ is not in the lexicon \mathcal{L} are described. First, we explain the estimation of the probabilities if the unknown word is represented by a regular expression. The probabilities are given as

$$\hat{P}_R(t | r) = \frac{|\{i \mid \tilde{t}_i = t \wedge \tilde{w}_i \in \mathcal{W}_r\}|}{|\{i \mid \tilde{w}_i \in \mathcal{W}_r\}|}. \quad (4.3)$$

Using these estimates for regular expressions allows assigning reliable tag distributions even to previously unseen tokens from training.

Now we deal with (still) unknown tokens. Previous work has shown that a word's prefix and suffix can successfully be used to determine the word's POS tag. Based on the set of training tokens \mathcal{L} we determine each prefix $p_l \in P$ and suffix $s_l \in S$ of maximal length β with $l = 1, \dots, \beta$ and store them in form of a letter tree. The set of considered prefixes/suffixes is further restricted by pruning this letter tree considering

the frequency and POS tag related entropy of a particular prefix/suffix. Therefore, we introduce random prefix variables $T_{p_l} \sim (\hat{P}_{p_l}(t))_{t \in \mathcal{T}}$ for $l = 1, \dots, \beta$ with

$$\hat{P}_{p_l}(t) = \frac{|\{i \mid \tilde{t}_i = t \wedge p(\tilde{w}_i) = p_l\}|}{|\{i \mid p(\tilde{w}_i) = p_l\}|}.$$

Considering the decision tree, beginning from each terminal node $l = \beta$ we recursively calculate

$$I(T_{p_l}, T_{p_{l-1}}) = |\{i \mid p(\tilde{w}_i) = p_l(w)\}| (H(T_{p_{l-1}}) - H(T_{p_l}))$$

with

$$H(T_{p_l}) = - \sum_{t \in \mathcal{T}} \hat{P}_{p_l}(t) \log \hat{P}_{p_l}(t).$$

A terminal node p_l is deleted if

$$I(T_{p_l}, T_{p_{l-1}}) < \gamma$$

for a given threshold γ . The tag frequencies of all deleted child nodes of a parent node are collected at a *default node* of the parent node. If the default node is the only remaining child node it is deleted, too. In this case the parent node becomes a new terminal node and is also checked whether it is deletable. The same pruning procedure is performed for suffixes. Note, that p_{l-1} is a substring from p_l .

We assess the lexical probabilities for a given word w by

$$\hat{P}_p(t \mid w) = \frac{|\{i \mid \tilde{t}_i = t \wedge p(\tilde{w}_i) = p(w)\}|}{|\{i \mid p(\tilde{w}_i) = p(w)\}|}$$

where $p(w)$ returns the maximum prefix of a given word w seen in the training data, reduced by the pruned set or limited by β . Lexical probabilities $\hat{P}_s(t \mid w)$ are defined equivalently. An open question is how to combine prefix and suffix tag distributions. In our approach, we propose four different combination methods and discuss and compare them in Section 4.6. First, we assume prefix and suffix tag distributions to be independent and hence use the joint probability distribution

$$\hat{P}_{ps}^g(t \mid w) = \frac{\hat{P}_p(t \mid w) \hat{P}_s(t \mid w)}{\sum_t \hat{P}_p(t \mid w) \hat{P}_s(t \mid w)}$$

later referred to as *geometric mean*. Combining prefix and suffix distributions in that way has been successfully applied to POS tagging performed on newspaper texts in [75]. This approach is adequate for POS tagging of standardized texts where unknown words occur only if they belong to the open class category and by, implication, the closed POS class can be excluded as possible predictions. In contrast, all POS classes need to be considered in social media texts where, e.g., a misspelled pronoun might lead to be unknown in the process of tagging. Hence, a more complex distinction needs to be made by the prefix and suffix information.

Furthermore, a more robust combination method for uncommon prefix and/or suffix due to informal writing style characteristics, e.g., word shortenings or typing errors, is needed. Therefore, in a second step we combine prefix and suffix tag distributions by taking the *arithmetic mean* for each tag probability:

$$\hat{P}_{ps}^a(t | w) = \frac{\hat{P}_p(t | w) + \hat{P}_s(t | w)}{\sum_t (\hat{P}_p(t | w) + \hat{P}_s(t | w))}. \quad (4.4)$$

In a third step, we define an approach aiming at choosing the most reliable tag distribution between $\hat{P}_p(t | w)$ and $\hat{P}_s(t | w)$. Therefore, the entropy of prefix and suffix tag distributions is used as a criterion. We introduce random variables $T_{p(w)} \sim (\hat{P}_p(t | w))_{t \in \mathcal{T}}$ and $T_{s(w)}$ analogously. The idea is to minimize the conditional entropy and hence choose the tag distributions which contain less uncertainty about the tag t to predict:

$$\hat{X} = \arg \min_{X \in \{T_{p(w)}, T_{s(w)}\}} H(X) \quad (4.5)$$

with

$$H(T_{p(w)}) = - \sum_{t \in \mathcal{T}} \hat{P}_p(t | w) \log \hat{P}_p(t | w).$$

However, the significance of the empirical prefix/suffix POS tag distribution strongly depends on the frequency of prefixes/suffixes. A prefix, which has been seen once, leads to no uncertainty about the tag and will fulfill the minimum criterion. Hence, we apply some simple tests on the frequencies before applying the minimum entropy approach (4.5). The first test checks whether the frequencies of both prefix and suffix exceed a predefined threshold α , i.e.,

$$\hat{P}_{p(w)} > \alpha \wedge \hat{P}_{s(w)} > \alpha$$

In that case, the distribution given by \hat{X} in (4.5) is used. As optional tests we check if exactly one of the thresholds is exceeded and use the corresponding probability distribution. If all these tests fail the distribution from (4.4) is used. We will evaluate this strategy later on, with and without the optional tests, referred as *Rule-based-2-case* and *Rule-based-4-case*.

Summarizing all cases with different estimation methods, the lexical probability is given as

$$P(t | w) = \begin{cases} \hat{P}_L(t | m(w)) & m(w) \in \mathcal{L}, \\ \hat{P}_R(t | m(w)) & m(w) \in \mathcal{R} \setminus \mathcal{L}, \\ \hat{P}_{PS}(t | w) & w \in (\mathcal{P} \cup \mathcal{S}) \setminus \mathcal{X}, \\ \hat{P}_S(t | w) & \text{otherwise,} \end{cases} \quad (4.6)$$

where \mathcal{P}/\mathcal{S} describes all tokens with known prefixes (suffixes). The last case in the description is by default given by the empirical tag distribution over the whole set of training tokens:

$$\hat{P}_S(t | w) = \frac{|\{i | \tilde{t}_i = t\}|}{\tilde{N}},$$

which is independently calculated of the token w .

In the previous part of this section, we have described different modifications of estimation techniques. In the following, we will introduce two semi-supervised learning techniques which aim at extending the training lexicon \mathcal{L} by some lexica which are not available in fully annotated text form, but provide information for semi-supervised learning methods.

Semi-supervised Verb Auxiliary Lexicon

Generally speaking, POS taggers for social media texts regularly deal with a frequent number of unknown verbs. This can be explained by the different dialog style of social media texts where different verb conjugations occur. Even a tagger trained on social media data only contains a small amount of such verbs due to the small corpus size. Hence, lexical probabilities can not be reliably estimated from prefix and suffix tag distributions for such verbs. However, preparing a fully-supervised social media training text with adequate corpus size is extremely time-consuming and requires expert knowledge from the annotator. We therefore propose an alternative approach which reduces annotation effort significantly.

The basic idea is to create a verb auxiliary lexicon with corresponding tag sets for each token. For approximately 14,000 verbs, a conjugation table including indicative and subjunctive forms for different tenses, as well as imperative forms, participle and infinitive is extracted from *www.verbformen.de*. In particular, the conjugation table contains two imperative forms, the general form of an imperative and first person singular, as well as a shortened form of both forms which frequently appears in social media texts. For an exemplary conjugation table, the corresponding POS tag is assigned manually to each verb form. Corresponding POS tags are automatically transferred to all other conjugation tables. Based on that conjugation tables all possible tokens, with their corresponding tag set denoted by \mathcal{T}_w , are combined in a verb auxiliary lexicon \mathcal{V}^+ containing 115,000 entries. If there is more than one possible tag, an adequate tag distribution needs to be assigned. Therefore, two approaches are utilized. First, all words \tilde{w}_i of the manually annotated training corpus with the same POS tag set \mathcal{T}_w are determined and the cumulated tag distribution of those words is used. Hence, the lexical probability is refined as

$$\hat{P}_{\mathcal{V}^+}(t | w) = \frac{|\{i | \tilde{t}_i = t \wedge \mathcal{T}_{\tilde{w}_i} = \mathcal{T}_w\}|}{|\{i | \mathcal{T}_{\tilde{w}_i} = \mathcal{T}_w\}|}.$$

where $\mathcal{T}_{\tilde{w}_i} = \{\tilde{t}_l | \tilde{w}_l = \tilde{w}_i\}$. We assume all $t \in \mathcal{T}_w$ to be equally distributed if no word with the same POS tag set \mathcal{T}_w exists.

The following example illustrates a particular case of a shortened verb. Consider the word *benutz* (*use*), which in a standardized text is just used as an imperative content word (*VVIMP*) and hence $P(VVIMP | \textit{benutz}) = 1$. Informally, however the word is also used as short form for the verb inflections of the first person singular *benutze* (*[I] use, VVFIN*). This form is also contained in the conjunction table contained in the verb auxiliary lexicon \mathcal{V}^+ . Hence, the cumulated tag distribution from the training corpus results in $P_{\mathcal{V}^+}(VVFIN | \textit{benutz}) = 0.62$ and $P_{\mathcal{V}^+}(VVIMP | \textit{benutz}) = 0.38$.

Semi-supervised Domain Specific Learning

In addition to a verb auxiliary lexicon we propose a lexicon developed for the social media text domain. The lexicon is annotated with a considerable reduction in annotation effort and is used for semi-supervised learning. The basic idea is as follows. The tagger is used for automatic tagging of a large social media text corpus.

$$\mathcal{SL} = \{(w_o, t_o) \mid 1 \leq o \leq O\}$$

The most frequent unknown tokens, $m(w_o) = \epsilon$, are determined. The resulting tokens are added to an auxiliary lexicon \mathcal{L}^+ comprising 2,000 entries. In order to create a domain-specific lexicon, which is topic-independent, topic related tokens are removed. We do that to ensure that the auxiliary lexicon helps to improve tagging accuracies for social media texts in general, independent of the topic they are dealing with. For all tokens w_o of the auxiliary lexicon the possible tags, i.e., the corresponding tag set, is manually assigned and denoted by \mathcal{T}_{w_o} . Tag distributions are assigned in the same way as for the verb auxiliary lexicon, if at least one word \tilde{w}_i of the manually annotated training corpus has the same POS tag set as the manually assigned set \mathcal{T}_{w_o} . Hence, the lexical probability is refined as the cumulated tag distribution of those words:

$$\hat{P}_{\mathcal{L}^+}(t | w_o) = \frac{|\{i \mid \tilde{t}_i = t \wedge \mathcal{T}_{\tilde{w}_i} = \mathcal{T}_{w_o}\}|}{|\{k \mid \mathcal{T}_{\tilde{w}_k} = \mathcal{T}_{w_o}\}|}.$$

However, in contrast to the method used to construct the verb auxiliary lexicon, if there is no word with the same set of possible tags in the training text, further manual annotation is performed. A reliable amount of such tokens is manually annotated in the large social media corpus SL . The resulting tag distribution is assigned to such unknown tokens. Integrating the semi-supervised learning techniques, formula (4.6) is adapted to

$$P(t | w) = \begin{cases} \hat{P}_L(t | m(w)) & m(w) \in \mathcal{L}, \\ \hat{P}_R(t | m(w)) & m(w) \in \mathcal{R} \setminus \mathcal{L}, \\ \hat{P}_{\mathcal{V}^+}(t | m(w)) & m(w) \in \mathcal{V}^+ \setminus \{\mathcal{L} \cup \mathcal{R}\}, \\ \hat{P}_{\mathcal{L}^+}(t | m(w)) & m(w) \in \mathcal{L}^+ \setminus \{\mathcal{L} \cup \mathcal{R} \cup \mathcal{V}^+\}, \\ \hat{P}_{PS}(t | w) & w \in (\mathcal{P} \cup \mathcal{S}) \setminus \mathcal{X}, \\ \hat{P}_S(t | w) & \text{elsewise,} \end{cases} \quad (4.7)$$

Note that prefix and suffix estimations are applied after all auxiliary lexicon lookups have been performed.

4.4.3 Transition Probability Estimation

Estimating transition probabilities in a k -order Markov model, particularly for high values of k and sparse training data is problematic. Since many frequencies are small, probability estimates are often not reliable. Particularly, in the context of social media texts the variability of k -tuple sequences of POS tags is higher compared to standardized texts, due to missing articles or personal pronouns, which makes it even more challenging.

To counteract this problem the proposed tagger estimates transition probabilities by treating the prediction of the next words POS tag based on a sequence of preceding POS tags as simple classification problem. The k preceding POS tags are considered to be the features, which serve to predict the next words POS tag, considered as class. Hence, a simple classification problem has to be solved, e.g., with a decision tree. The main advantage is that the preceding tags are treated position independent for prediction and estimates are more reliable for small training corpora. Additionally, a feature selection can be performed in order to reduce the complexity of the model, particularly for higher k -order models. In this work we spend special effort on solving this problem using a decision tree, as proposed in [75]. As in [75] we use the ID3 algorithm, [65], and adapt parameter settings to the characteristics of social media texts. In a classification tree each non-terminal node corresponds to the test of a feature, while the terminal nodes contains the class distribution. The ID3 algorithm choses recursively at each node the test on a feature, which leads to the maximal information gain, see equation (2.7), with respect to the considered classes. According to the tested feature it splits the training data and invokes itself recursively at the child nodes based on the subsets of training items.

We consider the set of k -grams $\{\tilde{t}_{n-k}^{n-1} \mid n = k + 1, \dots, \tilde{N}\}$ to be extracted from a training text. In the following we describe how the decision tree is built recursively based on the set of k -grams. First a binary discretization of the features, i.e., k -grams, as proposed in [75] is applied, in order to avoid overfitting. Instead of differentiating between each POS tag $t \in \mathcal{T}$ at a position $i = n - k, \dots, n - 1$ a simple binary feature is used in the form of the two cases that $\tilde{t}_i = t$ or $\tilde{t}_i \neq t$. Consequently, a binary decision tree is built up, where each node represents a test (new binary feature), if at a particular position $(n - k) \leq i < n$ in its history a particular POS tag t has been seen. In order to choose the test for a node l and a $t' \in \mathcal{T}$ the information gain criteria, see equation (2.7), is applied by

$$\begin{aligned} (\hat{i}, \hat{t}) = & \arg \max_{i: |\mathcal{T}_i^{(i)}| > 1, t \in \mathcal{T}_i^{(i)}} - \sum_{t'} \tilde{P}_l(t') \log \tilde{P}_l(t') \\ & + \tilde{P}_l(\tilde{t}_{k+1-i} = t) \sum_{t'} \tilde{P}_l(t' \mid \tilde{t}_{k+1-i} = t) \log \tilde{P}_l(t' \mid \tilde{t}_{k+1-i} = t) \\ & + \tilde{P}_l(\tilde{t}_{k+1-i} \neq t) \sum_{t'} \tilde{P}_l(t' \mid \tilde{t}_{k+1-i} \neq t) \log \tilde{P}_l(t' \mid \tilde{t}_{k+1-i} \neq t) \end{aligned}$$

where $\mathcal{T}_i^{(l)}$ is the set of possible tests at position i for a node l . Note that $-\sum_{t'} P_l(t') \log \tilde{P}_l(t')$ is a constant here and can be neglected for maximization. In the beginning at the root node $l = 1$ we initialize

$$\mathcal{T}_i^{(1)} = \mathcal{T}, \forall i = 1, \dots, k$$

since all tests are still possible. After determining (\hat{t}, \hat{i}) for a node l the corresponding set of POS tags is updated by

$$\mathcal{T}_i^{(l_0)} = \mathcal{T}_i^{(l)} \setminus \{\hat{t}\}$$

for the child node l_0 with the subset of k -grams where $\tilde{t}_{k+1-i} \neq \hat{t}$ holds and by

$$\mathcal{T}_i^{(l_1)} = \{\hat{t}\}$$

for the child node l_1 with the subset of k -grams where $\tilde{t}_{k+1-i} = \hat{t}$ holds. Conditional probabilities are given by the empirical distributions based on the training data, which pass the tree until they reach node l by

$$\tilde{P}_l(t' \mid t_{k+1-i} = t) = \frac{\sum_{n=k+1}^N \mathbb{I}_{\{t'\}}(\tilde{t}_n) \mathbb{I}_{\{t_i^k \in \mathcal{T}^{(l)} \mid t_{k+1-i} = t\}}(\tilde{t}_{n-k}^{n-1})}{\sum_{n=k+1}^N \mathbb{I}_{\{t'\}}(\tilde{t}_n) \mathbb{I}_{\{t_i^k \in \mathcal{T}^{(l)}\}}(\tilde{t}_{n-k}^{n-1})}$$

and

$$\tilde{P}_l(t' \mid t_{k+1-i} \neq t) = \frac{\sum_{n=k+1}^N \mathbb{I}_{\{t'\}}(\tilde{t}_n) \mathbb{I}_{\{t_i^k \in \mathcal{T}^{(l)} \mid t_{k+1-i} \neq t\}}(\tilde{t}_{n-k}^{n-1})}{\sum_{n=k+1}^N \mathbb{I}_{\{t'\}}(\tilde{t}_n) \mathbb{I}_{\{t_i^k \in \mathcal{T}^{(l)}\}}(\tilde{t}_{n-k}^{n-1})}$$

with

$$\mathcal{T}^{(l)} = \mathcal{T}_1^{(l)} \times \mathcal{T}_2^{(l)} \times \dots \times \mathcal{T}_k^{(l)}$$

Tag probabilities are calculated equivalently by

$$\tilde{P}_l(t_{k+1-i} = t) = \frac{\sum_{n=k+1}^N \mathbb{I}_{\{t_i^k \in \mathcal{T}^{(l)} \mid t_{k+1-i} = t\}}(\tilde{t}_{n-k}^{n-1})}{\sum_{n=k+1}^N \mathbb{I}_{\{t_i^k \in \mathcal{T}^{(l)}\}}(\tilde{t}_{n-k}^{n-1})}$$

and

$$\tilde{P}_l(t_{k+1-i} \neq t) = \frac{\sum_{n=k+1}^N \mathbb{I}_{\{t_i^k \in \mathcal{T}^{(l)} \mid t_{k+1-i} \neq t\}}(\tilde{t}_{n-k}^{n-1})}{\sum_{n=k+1}^N \mathbb{I}_{\{t_i^k \in \mathcal{T}^{(l)}\}}(\tilde{t}_{n-k}^{n-1})}.$$

The recursive expansion of the decision tree stops if the next test would generate at least one subset of k -grams whose size is below a threshold α . Since full disambiguation of the tag t_n based on the proceeding tags is in general not possible, the resulting terminal nodes contain probability distributions which reflect the tag distribution of the training data at that node.

Even with the given α this algorithm may still lead to an overfitted tree by splitting a training set too far. Therefore, the decision tree is pruned by the information gain criteria weighted with the frequency of the k -grams at the certain terminal node in a postprocessing. The weighted information gain at the terminal node has to exceed some threshold σ . Hence, the threshold σ can be adapted so that nodes, which do not contain enough training k -grams, are pruned to avoid overfitting.

In the previous part the construction of the decision tree, i.e., estimation of transition probabilities, has been explained. In order to use the decision tree for calculating $P(t'|t_k^t)$, the tree simply needs to be passed through with the given tag history $t_k^t \in \mathcal{T}^k$ and the probabilities $P(t'|t_k^t)$ for all $t' \in \mathcal{T}$ can be taken from the terminal node. Estimating the transition probabilities in that way still leads to many zero transition probabilities. To counteract the labeling bias problem, zero transition probabilities are replaced by an ϵ probability. This is particularly motivated by the fact that social media texts are non-standardized so that new k -tuples of POS tags might occur in the test data, e.g., due to missing articles or personal pronouns in a sentence.

Furthermore, we suggest to complement the social media text corpus by some out-domain training data to compensate the size of the relatively small corpus and reduce the number of zero transition probabilities, see the following Section.

4.4.4 Learning from Out-Domain Data

In this section, the term *domain* is associated with a text corpus characterized by a particular style. A social media text corpus is mentioned as in-domain corpus, whereas all texts with different characterization are out-domain texts. We define the combination of in- and out-domain training data as joint-domain training. Different experimental studies, e.g., [68, 52], have shown that out-domain training data can improve tagging accuracies. Particularly, the influence by adding newspaper training corpora for the task of tagging Twitter micro texts has been studied. A typical approach is to step-wise increase the amount of out-domain training and retrain the tagger on such data. Then the amount of out-domain training data achieving best results is determined. The literature reports consistently a significant performance increase by adding out-domain training data, particularly if only a small amount of in-domain training data is available. However, when the amount of out-domain training data exceeds the amount of in-domain training data significantly, accuracy drops again.

In contrast to existing approaches, we suggest an alternative method for combining in- and out-domain training data. The basic idea is a weighted joint-domain training. A manually annotated newspaper training corpus

$$\mathcal{TR}_{OD} = \{(\dot{w}_n, \dot{t}_n) \mid 1 \leq n \leq O\}$$

is added to our social media text corpus, see equation (4.2). In contrast to other approaches information from the whole available out-domain training corpus is used, no matter about corpus size. To cope with the different corpora sizes, we apply

oversampling to the in-domain social media text corpus. Therefore, we multiply the social media corpus \mathcal{TR} $\beta \in \mathbb{N}$ times, while combining it with the newspaper corpus \mathcal{TR}_{OD} . We use a set of combined training pairs

$$\mathcal{TR}_+ = \{(\tilde{w}_n, \tilde{t}_n) \mid 1 \leq n \leq \tilde{N} = O + \beta I\}$$

with

$$(\tilde{w}_n, \tilde{t}_n) = \begin{cases} (\hat{w}_n, \hat{t}_n) & 1 \leq n \leq O \\ (\tilde{w}_i, \tilde{t}_i) & n > O, i = 1 + (n - O - 1) \bmod (\tilde{N}). \end{cases}$$

The method of oversampling, see ,e.g., [62], has originally been proposed to handle the class imbalance problem in a training corpus. However, in this approach the oversampling technique is used to compensate the imbalance between in- and out-domain training data corpora. This plays a particular role in the task of POS tagging and has not yet been performed.

4.5 Text Corpora

In total three corpora are used in our experiments. Two corpora are used for training purposes, our social media corpus and an out-domain newspaper corpus. Furthermore, we introduce a third social media text corpus, which is used as additional test corpus.

First, we introduce *WebCom* a new corpus that contains social media texts in form of comments collected from *Heise.de*, which is a popular German newsticker site treating different technological topics. The corpus contains comments posted 2008 to 2009. The comments for the manual POS annotation are selected from this underlying corpus. In order to obtain a corpus where many kinds of social media characteristics are represented, we select comments from different users. The selection of comments is carried out randomly over different users according to their posting frequencies.

Each token is annotated with manually validated STTS POS tags. Annotation rules, particularly for social media text characteristics, are given in Section 4.3. A detailed annotation guideline as well as Inter-Annotator Agreement (IAA) studies can be found in [86]. As an annotation tool *EXMARaLDA* is used, which stores information in form of a xml structure. The *EXMARaLDA* editor allows for arbitrary annotation in texts on different levels. We call the resulting corpus *WebTrain* because it provides supervised data for training purposes. To the best of our knowledge, *WebTrain* is currently the largest social media text corpus enriched with POS information. However, aiming at training a POS tagger, approximately 36,000 tokens is a relatively small number.

Therefore, we additionally introduce the *TIGER* treebank [7] text corpus. It is the largest manually POS annotated German corpus and contains about 900,000 tokens of German newspaper text, taken from the *Frankfurter Rundschau*. The corpus annotation provides manually validated POS tags, lemmas, morphosyntactic features, and parse trees. For our purposes, only the STTS POS tag information is used.

	<i>WebCom</i>	<i>WebTrain</i>	<i>WebTypes</i>	<i>TIGER</i>
Text length, \tilde{N} :	15,080,976	36,284	4,006	888,973
#Words $ \mathcal{L} $:	360,177	7,830	1,637	89,417
#Sentences:	1,124,649	2,817	361	45,707
#Comments:	153,740	429	-	-
#Users:	15,007	183	-	-

Table 4.3: Corpus Statistics.

To have a deeper look in the general applicability of the retrained taggers for social media texts, we create an additional corpus *WebTypes*. It is composed of roughly 4,000 tokens, where comments from different Web sites and a corpus extract from the *Dortmunder chat corpus BalaCK 1-b* [5] are annotated in the same way than *WebTrain*. Three different types of social media texts are represented, YouTube comments, blog comments, and chat messages.

In order to give a first impression of the different requirements in POS tagging social media texts compared to newspaper texts, we determine some corpus statistics. First we determine the average POS tag ambiguity (size of the POS tag set \mathcal{T}_w) of tokens contained in the resulting lexicon of each corpus. For the *WebTrain* corpus we achieve a value of 2. This is significantly higher as the ambiguity in German newspaper texts, e.g., 1 for the *TIGER* corpus and is an intensified clue for the difficulty of POS tagging of social media texts. Table 4.3 gives further statistical corpus information, i.e., the text length \tilde{N} , the size of the resulting lexicon $|\mathcal{L}|$ for all corpora, the number of sentences and partly the number of comments and users contained in the corpus.

Spelling Correction

In Section 4.4.1 we have described the token preprocessing in form of a normalization, which can be automatically applied to any social media text, training as well as test text. This normalization includes basically the correction of tokens assigned to the category *Informal Writing Style*, see Chapter 1.1, but correction of, e.g., spelling, typing errors, or phenomena from one of the other categories is not included. Particularly, the correction of spelling errors is a challenging task and is not the focus of this work.

However, for experimental results it would be interesting to see how much non-standardized tokens/words from different categories influence on one hand the training process and on the other hand the performance of tagging previously unseen test data. For future experiments, we perform a manual correction of the *WebTrain* corpus in a two-level approach. The corrected texts on each level are stored and can later be used for evaluations. On a first level, called *SpellCorrected*, we correct spelling errors, typing errors and capitalization errors (category *Informal Writing Style*). *Spoken language characteristics* and *Social media language* phenomena, like, shorten words and word transformations, e.g., *nix -nothing* are corrected on the second correction level, called *WordCorrected*. The two level correction allows for a detailed analysis of the different types of characteristics inherent in social media texts.

Synthetic Data Creation

In Chapter 1.1 we have introduced the different characteristic categories of social media texts. Generally, a majority of non-standardized tokens belongs to the class *Informal writing style*. Hence, our particular interest goes to that category, most notably containing spelling and typing errors. However, the ratio between the number of misspelled tokens and the number of all tokens in a text is still pretty low. This makes it a challenging task to perform a reasonable comparison of different methods developed for such characteristics.

In order to perform a more detailed analysis of the different parameter estimation methods proposed, some synthetic data are generated. First, we assemble a list of typical spelling and typing errors from different resources discovered from the Web. As Web resources we use the following Web sites, *Rotkel.de*, *Korrekturen.de* and *Wikipedia.de*, which provide several types of such errors in alphabetic order. For our purposes we consider only one word errors based on one single token and resulting in one single token, in order to let the number of tokens unchanged. After eliminating multiple token errors and double entries from the different sources, the resulting *manipulation list* contains 1,900 entries. Each entry consists of the correctly spelled word and one up to three entries how it is frequently misspelled, i.e. non-standardized forms. Furthermore, we use our *WebTrain* corpus to extract non-standardized words/tokens inherent in social media texts. The list is created by comparing our original text with the second level text correction, *WordCorrected*, described in the previous section and extract all tokens that differ. Note that this list not only contains typing and spelling errors but all kinds of word transformations, particularly errors in capitalization. Adding these entries to the *manipulation list* ignoring double entries results in a total list of 2,700 word entries.

Based on the *manipulation list* we synthetically incorporate non-standardized tokens into our corpus. Each token of the *WebTrain* text, which is found in the list, is replaced by the non-standardized/misspelled form. In case of multiple non-standardized forms, we perform a random choice. Applying these modifications leads to 32% manipulated tokens. The synthetic data are exclusively used for testing.

4.6 Evaluation

The following section presents a detailed evaluation of the proposed Markov model tagger in many different ways. We particularly evaluate the impact of the proposed estimation methods for lexical and transition probabilities, the influence of the semi-supervised learning techniques enabled by the developed auxiliary lexica and finally the performance with different combinations and amounts of in- and out-domain training data. For the purpose of testing, we evaluate results achieved on the *WebTrain* test sets and the synthetically manipulated *WebTrain* test sets, but additionally study the results achieved on the *WebTypes* and *TIGER* newspaper corpus. By doing so the transferability to different social media text types is studied. Furthermore, we show

that using non-standardized texts for training does not lead to performance degradation on standardized newspaper texts. In addition to opposing the different methods proposed, WebTagger is compared to two state-of-the-art POS taggers throughout the evaluation. Finally, we give a short comment on the transferability of the proposed methods to other languages, especially to English.

Most evaluations are performed in a cross validation way performed on our *WebTrain* corpus. In order to achieve reliable statistics, we repeat ten times a 10-fold cross validation. For each of these cross validations, the *WebTrain* corpus is divided into ten equally sized subsets which are created by randomly selected sentences. In most evaluations, the taggers are trained in a joint-domain way on a combination of nine subsets and the first 700,000 tokens of the *TIGER* data in each validation step. However, some evaluations are performed without additional *TIGER* data on the sparse *WebTrain* corpus. The remaining *WebTrain* subset and *TIGER* subset is used for testing. In general, mean results over all cross validations are given in different contexts. As already mentioned in Section 2.7, mean values are calculated by macro-averaging.

For the following evaluations, we fix a standard parameter setting of the proposed Markov model tagger. This means that if no explicit changes are mentioned in the particular subsection, the fixed setup is used. Depending on the evaluation, the influence of one of these parameters is analysed and not fixed anymore. This setup is fixed as follows: We use a 2-order Markov model ($k = 2$), where transition probabilities are estimated with a decision tree pruned with a threshold of $\sigma = 50$. Zero transition probabilities are set to $\epsilon = 0.13$. Note, that the influence of choosing different values for k is not analyzed in detail in this thesis. Experimental results have shown that by choosing σ and ϵ adequately, tagging accuracies are not influenced significantly when varying k . Lexical probabilities are estimated by means of the three proposed adaptations, i.e., token normalization, token mapping to word classes and the auxiliary lexica lookups. For all remaining unknown tokens, information of the word's prefix and suffix is used to estimate lexical probabilities. The prefix and suffix lexicon is combined by means of the *arithmetic mean* for estimating lexical probabilities of unknown words. Suffix and prefix lexica are created based on a maximum length of 5 with a pruning threshold of 2. Furthermore, all parameter evaluations are performed on joint-domain training data without oversampling the *WebTrain* corpus, i.e., $\beta = 1$. The influence of oversampling is investigated separately at the end of this chapter.

4.6.1 Impact of Enhanced Parameter Estimation

First, the impact of each parameter estimation adaptation for the given Markov model is analyzed. The main goal of the proposed lexical probability estimation methods is to improve tagging accuracies of unknown tokens. In order to get an idea, with which ratio the particular methods contribute to the performance increase a stepwise analysis is performed. We particularly consider the interplay of the three methods, token normalization, token mapping to word classes and the auxiliary lexicon lookup. A detailed analysis of lexical probability estimation by prefix and suffix information

for tokens which are still unknown is performed in an additional step. Furthermore, we analyze the interaction between the three proposed methods in dependence of the in-domain training corpus size in more detail in Section 4.6.2.

The three parameter estimation adaptations are analyzed in the same order then they are used in the algorithm. Therefore, we train the tagger on the described ten times 10-fold cross validation in each step and add the different methods incrementally. Results for 2-order Markov model taggers are depicted in Table 4.4 when trained on sparse *WebTrain* data and Table 4.5 when trained on joint-domain training. In addition to total accuracies (ACC), the ratio of unknown words (#Unknown) and accuracies (ACC Unkn. ϵ) achieved on tokens treated by the normalization (ACC $m(w)$), mapped to word classes (ACC \mathcal{W}_r) or found in the auxiliary lexica (ACC $\mathcal{L}^+, \mathcal{V}^+$) are depicted. Standard deviations are depicted for the single evaluation measures by \pm . For all cross validations tests are performed and results are depicted for *WebTrain*, manipulated *WebTrain*, *WebTypes* and *TIGER* texts.

For all stepwise adaptations unknown word ratios can significantly be reduced and at the same time a performance increase is achieved on all text types except for the *TIGER* test. As expected the normalization function is not applied to any token in the *TIGER* test, hence no changes in tagging accuracy are achieved by this adaptation. Due to the fact that the text is standardized normalization is not needed. Notable is that the remaining adaptations lead to a performance increase of 1.19 percentage points when applied to the *TIGER* newspaper data. This shows that the proposed methods are also useful for standardized texts, when only sparse training data is available.

The introduction of text normalization leads to a relatively small improvement of 0.26 percentage points, when trained and tested on the *WebTrain* data. However, text normalization is only applied to relatively few tokens, but tagging accuracies on such tokens reach up to 89.42%. Improvements are much more higher with 1.18 percentage points, when applied to the *WebTypes* texts. This is due to the higher noise level in the data, so that text normalization is applied to more tokens. It is also reflected in the successful reduction of unknown words from 27.10% to 23.46%. Normalization becomes more important in the case of less text standardization. Mapping to word classes by regular expressions is applied to a relatively small number of tokens consistently for all test types. Nevertheless lexical probability estimates are reliable for such tokens so that tagging accuracies for such tokens are always over 90%. Unknown word reductions lead from 0.64 to 1.56 percentage points and are very similar for all text types except for the *TIGER* test. Results confirm that the mapping to word classes is an adequate method for all text types considered in our evaluation. Additional usage of the two semi-supervised auxiliary lexica, further increases accuracy between 1.25 and 0.31 percentage points, when trained on the *WebTrain* corpus only. The lexica $\mathcal{L}^+, \mathcal{V}^+$ achieve about 83% accuracy, which is significantly higher compared to prefix/suffix methods. It proves that knowing the correct tag set for a given word can already improve results significantly. Notable is that the performance of the auxiliary lexica drops about 30 percentage points, when applied to manipulated *WebTrain* data. This can be explained by a high number of verbs, where no known word with the same POS tagset T_w exists and hence estimates are less reliable, due to the equal tag distribution.

Furthermore, it has to be considered that accuracies are averaged over 100 different trainings but the *WebTypes* and *TIGER* test set are fixed in contrast to *WebTrain* test sets, and hence not exactly comparable.

Comparing the results of Table 4.4 and 4.5 shows that the proposed adaptations are particularly suitable when only sparse training data is available. However, the combination of all adaptations trained joint-domainly still leads to an improvement up to 2.91 percentage points for the *WebTypes* test set. Differences in results achieved by the semi-supervised auxiliary lexica are particularly interesting. On one side the effect of using the lexica is significantly higher when trained on sparse training data. Knowing the correct tag set from such lexica results in more reliable lexical probability estimates instead of estimating lexical probability by means of the prefix and suffix lexica. On the other side auxiliary lexica achieve consistently higher accuracies for all test types for joint-domain training data, except for the manipulated *WebTrain* test texts. This confirms that even enriching the training by out-domain newspaper training data, does not help to learn dialog specific verb forms and there is still need for adaptation. But applying the proposed lexica to such verbs leads high tagging accuracies. Tagging accuracies for the manipulated *WebTrain* data particularly drop due to a high number of synthetically generated capitalization errors, which are unrealistic in real social media texts.

	WebTagger			WebTagger (1) + Normalization ($m(w)$) (2)		
	#Unknown	ACC Unkn. ϵ	ACC	#Unknown	ACC $m(w)$	ACC
<i>WebTrain</i>	14.44 \pm 0.65	65.61 \pm 2.22	91.28 \pm 0.55	14.01 \pm 0.64	89.21 \pm 2.39	91.54 \pm 0.53
<i>WebTr. Mani.</i>	27.07 \pm 0.84	57.28 \pm 1.91	82.76 \pm 0.75	26.89 \pm 0.84	72.96 \pm 0.95	82.92 \pm 0.76
<i>WebTypes</i>	27.10 \pm 0.13	55.81 \pm 0.76	83.09 \pm 0.29	23.46 \pm 0.13	84.22 \pm 0.09	84.27 \pm 0.28
<i>TIGER</i>	35.81 \pm 0.07	64.51 \pm 0.21	83.88 \pm 0.09	35.81 \pm 0.07	—	83.88 \pm 0.09

	WebTagger (2) + word classes (\mathcal{W}_r)			WebTagger (3) + auxiliary lexica (\mathcal{L}^+ and \mathcal{V}^+)		
	#Unknown	ACC \mathcal{W}_r	ACC	#Unknown	ACC $\mathcal{L}^+, \mathcal{V}^+$	ACC
<i>WebTrain</i>	12.49 \pm 0.65	97.22 \pm 2.03	92.25 \pm 0.51	8.84 \pm 0.60	82.73 \pm 3.39	92.84 \pm 0.48
<i>WebTr. Mani.</i>	25.33 \pm 0.89	94.72 \pm 2.64	83.64 \pm 0.79	20.85 \pm 0.84	51.54 \pm 3.45	83.95 \pm 0.80
<i>WebTypes</i>	22.27 \pm 0.14	91.45 \pm 0.39	84.89 \pm 0.28	17.05 \pm 0.12	75.37 \pm 1.14	86.15 \pm 0.27
<i>TIGER</i>	34.72 \pm 0.08	97.04 \pm 0.14	84.62 \pm 0.09	29.56 \pm 0.07	81.54 \pm 0.39	85.78 \pm 0.08

Table 4.4: Stepwise evaluation of different parameter adaptations for different text types trained on *WebTrain* data.

For unknown tokens, which are not treated by any of the later discussed adaptations, lexical probabilities are determined by a tokens prefix and suffix. Different combination of prefix and suffix probability distributions are discussed in the following. Cross validation results for the different methods applied to different test types are depicted in Table 4.6. Additionally we depict results achieved with suffix and prefix distribution solely. Furthermore, the overall POS tag distribution $\hat{P}(t)$ without using any knowledge about the token is performed and depicted. On average each *WebTrain* test set contains about 4.22% tokens, where prefix/suffix estimation is applied, see

Test sample	WebTagger			WebTagger (1) + Normalization ($t(w)$) (2)		
	#Unknown	ACC Unkn.	ACC	#Unknown	ACC $t(w)$	ACC
<i>WebTrain</i>	7.77 ± 0.51	71.14 ± 2.98	93.55 ± 0.51	7.51 ± 0.49	89.42 ± 1.98	93.72 ± 0.50
<i>WebTr. Mani.</i>	22.08 ± 0.69	56.23 ± 2.12	83.35 ± 0.96	22.00 ± 0.72	68.61 ± 1.19	83.41 ± 0.91
<i>WebTypes</i>	17.53 ± 0.05	58.81 ± 0.40	87.19 ± 0.10	12.28 ± 0.06	88.70 ± 0.05	88.73 ± 0.13
<i>TIGER</i>	5.96 ± 0.00	87.53 ± 0.06	97.07 ± 0.01	5.96 ± 0.00	—	97.07 ± 0.01

Test sample	WebTagger (2) + word classes (W_r) (3)			WebTagger (3) + auxiliary lexica (\mathcal{L}^+ and \mathcal{V}^+)		
	#Unknown	ACC W_r	ACC	#Unknown	ACC $\mathcal{L}^+, \mathcal{V}^+$	ACC
<i>WebTrain</i>	5.98 ± 0.42	94.62 ± 2.93	94.06 ± 0.48	4.22 ± 0.39	86.40 ± 4.33	94.29 ± 0.46
<i>WebTr. Mani.</i>	20.48 ± 0.70	93.05 ± 3.63	83.70 ± 0.91	17.26 ± 0.67	44.06 ± 4.33	83.99 ± 0.90
<i>WebTypes</i>	10.95 ± 0.06	99.66 ± 0.45	89.29 ± 0.13	8.44 ± 0.05	81.69 ± 0.72	90.10 ± 0.11
<i>TIGER</i>	5.22 ± 0.00	93.76 ± 0.04	97.13 ± 0.01	4.75 ± 0.00	91.06 ± 0.24	97.20 ± 0.01

Table 4.5: Stepwise evaluation of different parameter adaptations for different text types trained on joint-domain data.

the upper table. Corresponding numbers for the different test types and standard deviations (\pm) are given in the particular parts of the table. Note the *WebTypes* and *TIGER* test sets are fixed and only the training data vary, hence mean and standard deviation values can not directly be compared to the other results. We calculate mean class precision and recall rates and the total accuracies for the whole test text (Total) and for the tokens, where prefix/suffix estimation is applied (Pref/Suf), i.e., unknown tokens. Experimentally we determine $\alpha = 50$ to be the best threshold for the *Rule-based-2-case (R-b2c)* and *Rule-based-4-case (R-b4c)* method and results for that value are depicted. The arithmetic mean method results in the best tagging accuracies with 94.29% when tested on the *WebTrain* test set. It particularly outperforms tagging accuracies achieved on unknown tokens. Nevertheless, this is not confirmed when applying the tagger to the manipulated *WebTrain* and *WebTypes* test sets. Here, the tagger slightly outperforms the arithmetic mean by using the R-b2c and R-b4c method. The entropy rule based approach leads to a more robust approach against high noise level in the text. Furthermore, it is more robust in the application to texts different from the training texts. Comparing the results of the different methods evaluated on manipulated *WebTrain* and *WebTypes* texts show, that tagging accuracies differ more than on the *WebTrain* corpus itself. This confirms that for social media texts with a low standardization appropriate selection of these methods is important. Considering the mean class precision, the geometric mean method significantly outperforms the other methods with 65.73% accuracy achieved on prefix/suffix tokens. Results are confirmed, when comparing the mean class precisions for all other text types. The R-b4c approach reaches slightly better mean class recall results compared to the arithmetic mean.

Previous studies have shown, that adding prefix information for automatic tagging of newspaper texts only leads to little improvement of 0.05 percentage points, see [75]. In our approach running the tagger only with a suffix lexicon results in 0.15 and only with

a prefix lexicon in 0.35 percentage points performance loss compared to the arithmetic mean method applied to the *TIGER* newspaper test. Comparing the results to those achieved on different social media texts shows, that performance loss reaches up to 0.94 percentage points, when only using a prefix lexicon.

Depending on the later application, requiring POS tag information, different methods should be chosen. One might be interested in a high per class precision rather than total accuracy. Furthermore, the goal might be to choose the method, which achieves best results on the particular training text type or a method which is robust against a high noise level in the test texts. When, dealing with standardized newspaper texts one should stick to the geometric mean method, however would not suffer from a huge performance loss, when using any of the other proposed methods. For further evaluations in this thesis, the arithmetic mean method is used as standard method.

	Mean Precision		Mean Recall		Mean Accuracy	
	Pref/Suf	Total	Pref/Suf	Total	Pref/Suf	Total
WebTrain Test (4.22 ± 0.39%)						
Prefix	35.00	83.48	38.55	87.57	60.22	93.77
Suffix	36.35	82.27	34.02	87.03	69.18	94.16
Arithmetic	45.63	82.22	48.53	87.43	72.05	94.29
Geometric	65.73	84.14	39.58	87.19	71.72	94.27
Rule-base 2-case	46.25	82.14	49.86	87.35	71.78	94.27
Rule-base 4-case	41.37	82.22	47.27	87.60	69.38	94.17
$\hat{P}(t)$	16.98	82.98	17.61	86.23	45.99	93.17
WebTrain Manipulated Test (16.94 ± 0.68%)						
Prefix	38.91	70.17	41.79	74.96	53.94	83.27
Suffix	27.27	68.41	23.89	68.10	44.99	81.54
Arithmetic	41.62	72.84	43.42	75.14	58.97	84.21
Geometric	56.78	75.33	34.54	71.09	57.95	83.92
Rule-base 2-case	43.30	72.10	44.85	75.63	60.26	84.41
Rule-base 4-case	41.38	71.54	47.05	76.46	60.94	84.51
$\hat{P}(t)$	19.63	69.96	15.45	64.99	34.82	79.98
WebTypes Test (8.44 ± 0.05%)						
Prefix	30.44	76.70	30.56	83.07	47.45	89.26
Suffix	31.13	80.98	25.52	83.01	57.50	89.90
Arithmetic	33.72	79.58	33.12	83.51	57.86	90.10
Geometric	48.93	81.53	29.13	83.33	56.98	89.91
Rule-base 2-case	35.13	79.62	36.35	83.71	58.22	90.12
Rule-base 4-case	31.00	79.30	36.37	83.56	56.69	89.97
$\hat{P}(t)$	9.96	78.25	13.52	81.76	34.56	88.05
TIGER Test (4.75 ± 0.0%)						
Prefix	23.65	87.27	39.44	89.58	82.63	96.86
Suffix	25.65	87.93	42.60	89.67	86.75	97.06
Arithmetic	33.09	87.80	52.80	89.86	89.50	97.20
Geometric	55.80	89.05	49.73	89.69	89.58	97.21
Rule-base 2-case	32.05	87.78	51.11	89.83	89.50	97.20
Rule-base 4-case	26.85	88.06	47.90	89.75	88.43	97.15
$\hat{P}(t)$	14.87	88.61	19.34	89.29	66.36	96.07

Table 4.6: Influence of different prefix/suffix combination estimation methods trained on joint-domain training evaluated on different test types.

Finally, we investigate different maximum length of prefixes and suffixes for $\beta = 1, \dots, 10$. The interplay of the different β values for prefix and suffixes in a 2-order Markov model with respect to the total tagging accuracy is depicted in Figure 4.1. Results are mean accuracies over the ten cross validations performed on the

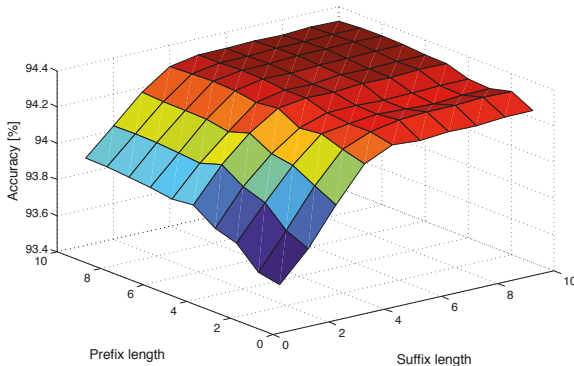


Figure 4.1: Accuracies with different prefix/suffix maximal length combined with the arithmetic mean.

WebTrain corpus. The 3-dimensional plot clearly illustrates the importance of suffix information. This, corresponds to the fact that the words suffix varies according to the particular POS class, e.g., the conjugation for verbs. Using a length of more than five does not lead to a significant performance increase rather for the suffix nor prefix lexicon. This is assumed, since most of the tokens are not much longer in their total length, hence the resulting prefix/suffix tree does not change much. At the same time, prefix and suffix information are overlapping with growing length and no additional information about the words POS class is gained by combining them. Note that different pruning thresholds for prefix and suffix lexica do not influence accuracies significantly. Experimentally, we determine $\gamma = 2$ for the prefix tree and the suffix tree.

4.6.2 Impact of In- and Out-domain Training Data

In this section we investigate the proposed tagger trained on numerous training data combinations composed of in- and out-domain data. On one hand we point out the importance of using in-domain training data for the purpose of tagging social media texts and investigate the results achieved by stepwise increasing the amount of such data. We additionally show that using non-standardized social media texts for training does not lead to a significant degradation when applying WebTagger to standardized newspaper texts.

On the other hand the way to combine the two corpora *WebTrain* (in-domain) and *TIGER* (out-domain) to reach maximum accuracy is analyzed. We particularly compare the proposed linear combination of joint-domain training by oversampling the

smaller in-domain *WebTrain* corpus to existing approaches, where the ratio between in- and out-domain training is adjusted by the out-domain corpus size.

Importance of in-domain training data

To further investigate the influence of using in-domain training data, i.e., social media texts for training, we train our model based on different training corpora. Note, that all trained models in this section are trained without replacing transition probabilities by a small epsilon. First, we train our model exclusively on newspaper *TIGER* texts. We stepwise increase the amount of training data from 100,000 to 700,000 tokens. In each step we randomly choose sentences comprising 100,000 tokens. This is performed 100 times and data is added to the data selected in the previous step. Hence, the model is trained on 100 different samples in each step. Second, in twenty steps 1,000 up to 20,000 tokens of the *WebTrain* corpus are combined with a sample set of 700,000 newspaper training tokens. Here, we choose the newspaper training sample (700,000 tokens) achieving mean tagging accuracy, when tested on *WebTrain* test set. Additional *WebTrain* tokens are chosen randomly, sentence wise. Again we select 100 sample sets, in the same way as for the newspaper training and train our model on such data for each iteration step. Testing is performed on the remaining data, a fixed test set of *WebTrain* with approximately 6,000 tokens. Mean results over 100 different trainings per point are depicted in the curve marked with \triangle in Figure 4.2. The plot contains different x-axis scalings for the left and right area next to the black vertical line to better illustrate the results. Significant slope increase can be observed in this point, which proves the success by using in-domain text specific training data for the task of POS tagging. Using 20,000 social media text tokens results in approximately 2.4 percentage points performance improvement on average. Hence, little effort of manual annotation leads to a significant performance improvement. Increase of 600,000 newspaper training tokens results in approximately 5.8 percentage points improvement solely. Furthermore, we show that including grammatically non-standardized texts as training data does not negatively effect the annotation of standardized text by means of the proposed approach. Random sentences are chosen from the newspaper *TIGER* corpus to create a test set of 90,000 newspaper tokens. We use WebTagger trained on the different training corpora to tag the newspaper data. The curve marked with \circ in Figure 4.2 illustrates the results. Results proof that adding 20,000 social media text tokens for training do not affect tagging accuracy for standardized texts essentially.

Comparison of tagging accuracies for social media texts and newspaper texts states that the tagging accuracy on standardized text can not be achieved when applying our approach to social media texts. However, the performance difference can be reduced from approximately 10 percentage points to 4 percentage points by increasing the amount of training data from 100,000 tokens to 720,000 tokens in total. Furthermore, matching the slope of both curves for the left area, states that increasing the amount of newspaper training data is more substantial for the application to social media texts compared to the application to newspaper texts. Tagging accuracy can be improved

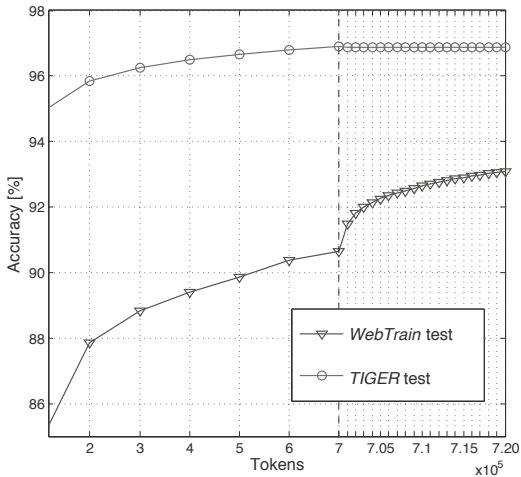


Figure 4.2: Influence of additional newspaper/social media training texts, tested on social media and newspaper texts.

by 2 percentage points for newspaper data and 5.8 percentage points tested on social media texts by adding the same amount of newspaper training data.

Training our model on 700,000 *TIGER* tokens leads to similar results, when tested on newspaper data compared to TreeTagger results reported in [25]. For 90,000 randomly selected testing sentences chosen from the *TIGER* corpus, WebTagger achieves 96.9% accuracy on average.

Following this evaluation, we study the influence of additional *WebTrain* training data to the different social media text types depicted in Figure 4.3. The accuracy improvements over the different training data amounts are depicted in the corresponding curves. For all social media types the stepwise addition of *WebTrain* training data leads to a consistent accuracy increase. For *WebTypes* related text types, which show more social media text characteristics, the slope of the curves is higher compared to the particular training data type *WebTrain* (test, 6,000 tokens). Increasing the amount of *WebTrain* training data leads to a significant performance increase, particularly for blog comments and YouTube comments. Results approve that general social media text characteristics can be learned from comments present in the *WebTrain* corpus. In summary, the results from Table 4.10 and Figure 4.3 show that the adapted parameter estimation methods combined with a sufficient amount of *WebTrain* training data leads to adequate tagging accuracies for social media texts in general. Results clearly demonstrate that the proposed tagger can successfully be applied to other texts belonging to the social media text genre.

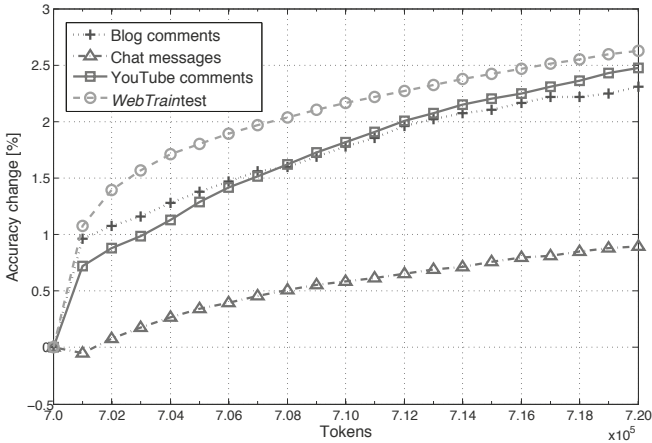


Figure 4.3: Influence of additional *WebTrain* training for different social media texts.

Note that we exclude Twitter messages from this scope, since this subset can not be addressed suitably with the presently developed method. Due to their special characteristic given by hard distractions to 140 characters, the proposed method needs to be further adapted.

Finally, the interaction between the amount of training data and the different adaptation methods for lexical probability estimation is illustrated in Figure 4.4. For testing the same 6,000 test tokens like in Figure 4.2 are used. We stepwise adapt the lexical parameter estimation method by our proposed methods, similarly to the procedure performed in Table 4.5. Significant impact of introducing text normalization and word classes is observed over the whole training data range. Using auxiliary lexica leads to a significant performance increase, particularly for a small amount of training data. Comparing the slopes of the curves marked with ∇ and \star illustrates that the sufficient training data amount is much higher to compensate the improvement achieved by normalization and word classes methods. In total, all estimation adaptations can be partially compensated by adding additional social media training texts at least for this test sample. This has to be studied in more detail for different test samples. However, manual annotation of complete texts for fully supervised training is a very time consuming step. Creating an auxiliary lexicon with our proposed methods shows a better trade-off between time for annotation and improvement in tagging accuracy.

Joint-Domain Training With Oversampling

In this section we investigate the influence of out-domain training data in more detail.

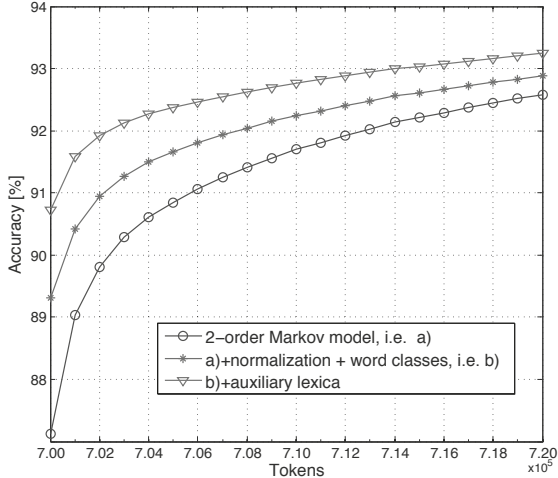


Figure 4.4: Stepwise parameter estimation adaptations for increasing social media training texts.

We particularly compare our proposed linear combination of joint-domain training to existing approaches, where the ratio between in- and out-domain training is adjusted by the out-domain corpus size. First we stepwise increase the amount of *TIGER* training data. Starting with a size equal to *WebTrain corpus* size, we randomly choose sentences in each step. This is performed 100 times and data is added to the data selected in the previous step. Each of these out-domain training samples is combined with each training of a 10-fold *WebTrain* cross validation (3,600 tokens each part). Mean accuracies of cross validation tagging over all 1000 training samples are depicted for different in-/out domain ratios in the curve marked with (\star) in Figure 4.5. Additionally the minimum and maximum accuracy of the 100 *TIGER* training samples is depicted in the curve marked with (Δ) and curve marked with (∇). We further investigate the influence of different join-domain training methods to different social media text types, where the tagger is not trained on that particular type. Therefore, we exemplarily evaluate tagging accuracies achieved on the YouTube data. Results are depicted in the same way in Figure 4.6.

In order to give some reference values, we train our tagger exclusively on the *TIGER/ WebTrain* corpus. Accuracies are depicted by the dotted line in both figures. Second we apply our linear combination approach with oversampling and combine the *TIGER* and *WebTrain* corpus in the same cross validation for different β values. Cross validation results and test results achieved are depicted in the curve marked with (\circ). First, we compare the accuracies achieved with our approach (\circ) to those achieved with the best *TIGER* training part (∇). The black curve (\circ) stays above the red

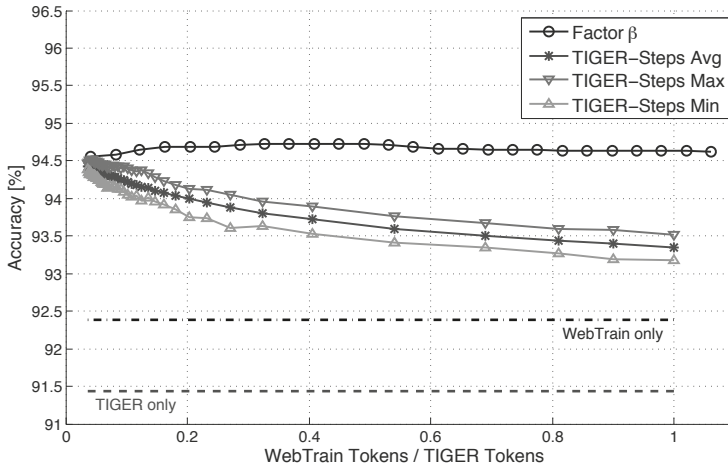


Figure 4.5: Influence of different joint-domain trainings evaluated on *WebTrain*.

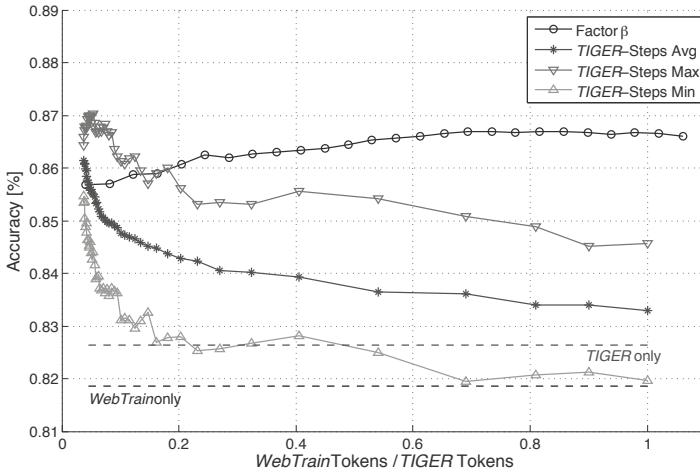


Figure 4.6: Influence of different joint-domain trainings evaluated on YouTube.

curve (∇) over all in-/out-domain ratios. The curve (∇) represents the optimum result for the given number of out-domain tokens. The plot indicates that exploiting this degree of freedom the performance of our approach is hardly reached. Determining the optimum training corpus results in a huge evaluation effort, which is very time consuming. If the *TIGER* training part is not determined properly and, e.g., chosen randomly, tagging accuracies can be significantly lower. In the worst case minimum accuracies depicted in the curve (Δ) are achieved. Applying our method with $\beta = 10$ results in a maximum cross validation accuracy of 94.74%. Determining the best β is considerably faster compared to identifying the best *TIGER* training part. Even if no effort is spent on determining the best β , accuracies are only slightly lower than optimum.

Moreover, we want to discuss how our approach performs on text types for which the tagger is not particularly trained. Therefore, we discuss the results achieved for YouTube data depicted in Figure 4.6. The maximum tagging accuracy of 87% is achieved with a standardized approach. This is slightly higher compared to the maximum of 86.7% achieved with our approach. However, comparing the mean, max and min curves shows that it is crucial, which out-domain data part is used as additional training. This requires a time-consuming study. Furthermore, the correct decision in practice is hard to achieve, since maximum values are located at different ratios compared to the cross validation results. For both evaluations it is obvious that our approach is robust in the sense that the performance slightly changes, if the ratio of tokens is changed. This is similar to the result depicted in Figure 4.5 and hence shows the robustness of the method, no matter what β value we choose.

Finally, we compare the results achieved for exclusively trained taggers on *TIGER/WebTrain* corpus. All combination methods significantly exceed accuracies achieved for single training over all in-/out domain ratios. This states that a joint-domain training approach is always reasonable. Note that for all following tagger evaluations training is performed on joint-domain training, where *WebTrain* data are oversampled with a factor $\beta = 10$.

Impact of Newly Seen Trigrams

In order to expand how different the grammatical structure in social media texts is compared to newspaper texts, STTS tag trigram frequencies are calculated for both corpora *TIGER* and *WebTrain*. The overall results are depicted in Table 4.7. The

	Trigrams	Trigram frequencies	Ratio
Total <i>WebTrain</i>	7,215	36,282	0.20
Total <i>TIGER</i>	16,563	888,982	0.02
Only in <i>WebTrain</i>	1,290	2,120	0.61

Table 4.7: Trigram comparison for *TIGER* and *WebTrain* corpora.

third column shows the ratio between different trigrams and their frequencies for the

different corpora. Results illustrate the higher variability in social media texts, which is ten times higher than in newspaper texts. Particularly, we compare statistics for tag trigrams that occur in *WebTrain* texts but are unknown from the *TIGER* corpus. The statistics are given in the last row. *WebTrain* texts contain 18% new trigrams, that never occur in the newspaper corpus *TIGER*. Those trigrams constitute 6% frequency of all *WebTrain* trigram counts. Particularly, for those trigrams the ratio/variability is increasing by a factor of three. Both results motivate the need of in-domain training data for reliable estimation of transition probabilities, e.g., for trigrams.

4.6.3 POS Tag Confusion and Category Specific Evaluation

In this section, some more detailed analysis on the types of tagging errors and their reduction by the proposed methods is performed. Apart from different POS tag confusion error types, we analyse tagging errors made in different social media text xcharacteristic categories. By considering these two points, it is shown in more detail which problems are addressed by the proposed methods, leading to enhanced accuracies.

First, the ten most frequently confused tag pairs, i.e., error types, for our approach are further investigated. Frequencies achieved with WebTagger and its standard setup plus oversampled joint-domain training are used for determining the ten tag pairs. Results achieved by this setup are compared to the Markov model tagger without any of the proposed adaptations (Table 4.5 WebTagger (1)). Tagging errors are represented by absolute frequencies and ratios of confused tag pairs for both tagger approaches. Results are depicted in Table 4.8. The left part of the table shows the confusion rates achieved with WebTagger (1) whereas the middle part of the table depicts confusion with the final adapted model. The right part of the table shows the error reduction in percent achieved for each type of confusion. Ratios/Relative errors are calculated between the sum of all confused tag pairs over all ten *WebTrain* cross validations and the total number of test tokens (362,800). The top two confusion pairs *noun* (NN) and *named entity* (NE) account for 10% of the errors when applying WebTagger. This effect is not unique to social media texts as it also occurs when tagging newspaper texts. Distinguishing proper nouns from named entities is done by named entity recognition which cannot be solved by general POS taggers. Nevertheless, with 34% and 24%, significant reduction of this error types is achieved. A detailed analysis shows that improvements are made over all adaptation steps integrated in WebTagger. Notable improvements are achieved for still unknown tokens where prefix and suffix lexica are applied. Lexical probabilities are more reliably estimated based on the oversampled training. Furthermore, improved tagging accuracies achieved on compound words such as *Heise-Seiten* fall into this error type reduction. Interchanging a *finite verb* (VVFIN) and a *non-finite verb* (VVINF) is caused by a non-local dependency particularly in German. This is also reported for state-of-the-art taggers and illustrated in [75]. Improvements on these confusion types are particularly made by more reliable estimates based on prefix and suffix estimation but at the same time improvements are achieved by applying the verb lexicon. Noticeable is the occurrence of tag confusion between *foreign language* (FM) and *named entity* (NE). Social media texts are often multilin-

gual and contain text parts written in different languages, e.g., a German comment contains English text parts (FM). The tokens of such text segments are annotated as *foreign language* (FM). Due to missing prefix/suffix information of such tokens, this leads to tagging errors. Tagging accuracies are slightly improved by more reliable suffix and prefix estimates. Frequent tag confusion between *noun* (NN) and *attributive adjective* (ADJA) results from missing noun capitalization which causes a valid adjective, from self created tokens or token transformations. For this error type, as well as for the confusion pairs *irreflexive personal pronoun* (PPER) with *reflexive personal pronoun* (PRF) and *substituting demonstrative pronoun* (PDS) with *article* (ART) significant tagging improvements are achieved by the proposed parameter estimation adaptations. Tokens which fall into one of these error categories are frequently known tokens where enhanced lexical probability estimates due to the oversampling of social media texts in the training lead to higher tagging accuracies.

Note that for all error types some improvements are achieved on known words. This can be explained by changed transition probabilities due to the fact that surrounding words are tagged correctly.

WebTagger (1)				WebTagger (3) + Oversampling				Error Reduc.
Correct	Predicted	Absolute	Relative	Correct	Predicted	Absolute	Relative	
NE	NN	1542	0.43	NE	NN	1025	0.28	33.53
NN	NE	1229	0.34	NN	NE	936	0.26	23.84
VVFIN	VVINF	941	0.26	VVFIN	VVINF	880	0.24	6.48
FM	NE	655	0.18	FM	NE	695	0.19	6.11
VVINF	VVFIN	465	0.13	VVINF	VVFIN	486	0.13	4.52
PPER	PRF	351	0.10	PPER	PRF	485	0.13	27.63
PDS	ART	686	0.19	PDS	ART	435	0.12	36.59
VVFIN	VVPP	490	0.14	VVFIN	VVPP	402	0.11	17.96
KON	ADV	616	0.17	KON	ADV	340	0.09	44.81
NN	ADJA	485	0.13	NN	ADJA	322	0.09	33.61

Table 4.8: Most frequently confused tag classes on *WebTrain* test before and after adaptations (total number of test tokens: 362,800).

Finally, we evaluate the result for all social media text types with respect to the four different characterization categories introduced in Section 1.1. The goal is to show which characterizations of particular categories can be handled successfully and which are still a problematic task. In order to show how strong the described adaptations improve accuracies in each category, we filter and classify all words which are not correctly tagged by using WebTagger with the fixed setup plus oversampled joint-domain training (WebTagger (3)+Oversampling) and the Markov model tagger without any adaptations (WebTagger (1)). For a detailed evaluation, the tagging errors are consistently classified into one of the four categories in a manual process by one person. Note that the resulting classification does not serve for training but rather for evaluation purposes. Figure 4.7 exemplarily depicts absolute errors for each category when tested on *WebTypes*. The shaded areas illustrate the absolute error reduction for each particular category evaluated on a sample 10-fold cross validation. Hence, errors are summed up over ten tests performed on 4,006 tokens each.

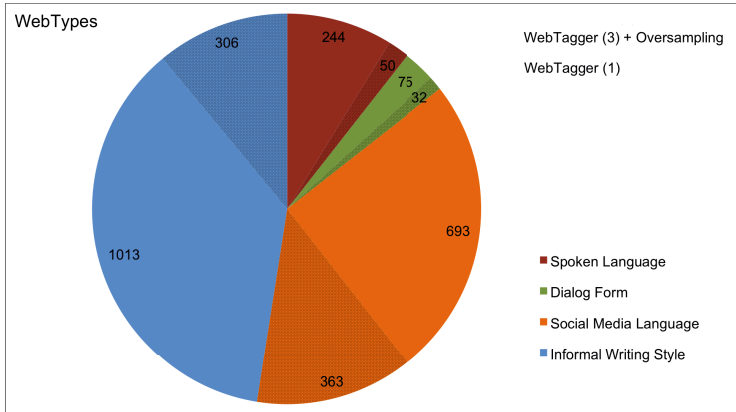


Figure 4.7: Error categorization evaluated on a sample cross validation performed on *WebTypes* (40,060 test tokens).

Applying WebTagger (3) trained with oversampling, the overall error rate for all four social media categories compared to WebTagger (1) without adaptations is reduced by 27%. The highest error reduction is achieved for the *social media language* category where errors can be reduced by more than a third. The combination of oversampling and adequate token handling by the auxiliary lexica leads to improved results. By introducing the verb auxiliary lexicon, tagging errors can be reduced by 30% in the category *dialog form*. In the *informal writing style* category, error reduction is achieved by applying token preprocessing where tokens are normalized and mapped to known tokens. The handling of spelling errors, which fall into this category, are still problematic. The lowest error reduction is achieved in the category *spoken language* with 17%. It is achieved by providing adequate oversampled in-domain training data. In total, adequate unknown word treatment plus oversampling the in-domain training data enables for the special handling of social media text characteristics, particularly for *social media language* and *dialog form* categories. However, a significant error reduction can be achieved over all categories.

4.6.4 Comparison to State-of-the-Art Taggers

Finally, we compare the adapted WebTagger to two state-of-the-art taggers, TreeTagger [75] and Stanford [83], see Table 4.9. Both state-of-the-art taggers, using their standard parameters, are trained and tested on the same ten cross validation samples based on a joint-domain training as described at the beginning of this chapter. Therefore, a fair comparison between the different tagger models can be performed. However, it still has to be kept in mind that state-of-the-art taggers benefit from the social media text corpus developed in this thesis.

	WebTagger	TreeTagger	Stanford
WebTrain Test (3,628± 33.07)			
Total	94.74 ± 0.43	93.84 ± 0.54	93.50 ± 0.56
Known	95.89 ± 0.38	95.87 ± 0.46	95.76 ± 0.44
Unknown	80.57 ± 2.43	68.70 ± 3.10	69.68 ± 2.76
Percentage unknowns	7.51 ± 0.49		
WebTypes Test (4,006)			
Total	90.81 ± 0.14	88.02 ± 0.13	86.27 ± 0.09
Known	93.97 ± 0.14	93.98 ± 0.11	93.44 ± 0.08
Unknown	68.23 ± 0.54	52.39 ± 0.64	49.32 ± 0.40
Percentage unknowns	12.28 ± 0.06		

Table 4.9: Tagger comparison for different text types trained on joint-domain training data evaluated for 10x10-fold cross validation.

Total tagging accuracies and accuracy rates achieved for known tokens and unknown tokens are determined. Mean accuracies and their standard deviation (\pm) as well as unknown token ratios are depicted in Table 4.9. In order to make results comparable, unknown token ratios are calculated differently compared to those in Table 4.5. Unknown tokens are tokens, which are not known from the training text corpus in their original form and therefore have the same ratios for all taggers. WebTagger significantly exceeds the mean tagging accuracy compared to all state-of-the-art taggers. During the ten test runs for the first cross validation we perform 30 single comparisons between WebTagger and the other two. WebTagger performs better in 28 of 30 cases. Differences between the taggers are statistically significant according to a corrected resampled paired t -test [48] applied to all cross validations with a significance level of $p = 0.001$. Particularly, the accuracy on unknown tokens can be improved by our approach. This is mainly due to the adaptation of how lexical probabilities of unknown tokens are estimated. Furthermore, combining prefix and suffix lexical probabilities by the arithmetic mean makes the method robust for unknown tokens. Overall, considering the noisy characteristics of social media texts, a considerable enhancement is achieved by WebTagger.

Finally, we compare the performance of WebTagger to state-of-the-art taggers on different social media text types, where the tagger is not trained on the particular type. To illustrate the improvements, Table 4.10 shows tagging accuracies and standard deviations for WebTagger and the four selected state-of-the-art taggers. We compare the results for the combined *WebTypes* test data to results achieved for single types, blog comments, chat messages and YouTube comments, introduced in Section 4.5. Application of WebTagger leads to a consistent performance increase between approximately 2 and 7 percentage points for different social media text types. Best improvements can be observed for YouTube comments, which are highly characterized by a dialog form and social media text characteristics such as emoticons, word shortenings or letter iterations. Even though considerable improvements are achieved, the tagging accuracy of 88.16% is the lowest compared to all other types due to the low text stan-

	#Tokens	WebTagger	TreeTagger	Stanford
<i>WebTypes</i> test	4,006	90.81 ± 0.14	88.02 ± 0.13	86.09 ± 0.12
Chat messages	1,728	91.70 ± 0.17	89.63 ± 0.27	87.81 ± 0.16
YouTube comments	1,463	88.16 ± 0.28	84.41 ± 0.41	81.23 ± 0.16
Blog comments	815	93.58 ± 0.24	91.61 ± 0.25	91.35 ± 0.17

Table 4.10: Tagger evaluation for different text types trained on joint-domain data.

standardization. Overall, WebTagger outperforms the state-of-the-art taggers for all social media text types.

4.6.5 Portability to Other Languages

The basic model and parameter estimation enhancements of the proposed WebTagger are language independent. It is adapted to the social media text characteristics in general, e.g., emoticons or character repetitions. However, considering all minor effects that depend on language specific properties requires some additional effort. E.g., the principal word normalization can be adopted for other languages, but must be adapted to particular language effects or typical verb transformations need to be covered by a language dependent verb auxiliary lexicon. Moreover, language specific training would require an additional supervised social media text corpus in the particular language. For the corpus annotation the extended annotation rules can be used analogously. Evidently, POS tags need to be mapped to the language specific tag set.

Furthermore, we consider the generality of the proposed tokenizer. In total, 30 of the 32 criteria used to detect coherent tokens cover social media text characteristics and, hence can be transferred to other languages without any adaptation. The remaining 2 criteria as well as the merged word list are language specific adaptations to German.

4.7 Conclusions

A new POS tagger called *WebTagger*, designed for the annotation of social media texts, has been presented. WebTagger achieves an average accuracy of 94.7% evaluated in ten cross validation experiments on a German social media text corpus consisting of comments extracted from a news site. It outperforms state-of-the-art taggers considerably with about 1 percentage point accuracy improvement. Additionally, it yields a minimum improvement compared to state-of-the-art taggers of 2.8 percentage points for a social media text type corpus different from the training corpus type.

Our approach differs fundamentally from other statistical Markov model taggers in estimation of lexical probabilities for unknown tokens. A novel approach mapping unknown tokens to tokens either known from training or tokens which fall into a class represented by regular expressions has been presented. For remaining unknown tokens semi-supervised domain-specific and verb auxiliary lexica and an adequate combination

of tag distributions derived from prefix and suffix information are applied. By doing so, unknown word frequencies can be reduced up to a factor of two and accuracies on such tokens are improved. Altogether, lexical probability distributions are estimated more accurately for social media texts.

Furthermore, the interplay between in-domain, i.e., social media texts, and out-domain, i.e., newspaper texts, training data is investigated from different perspectives. First, the importance of manually annotated in-domain training data is investigated. Considerable improvements are achieved by using only a small amount of 20,000 tokens as additional data for supervised training. Using such training data allows for reliable transition probability estimation by learning the different grammatical structure of social media texts. At the same time, the enrichment by non-standardized training texts does not negatively affect tagging accuracies of standardized texts. Second, we have enriched the social media text corpus by a linear combination following an over-sampling technique with a newspaper training corpus. Tagging accuracies for different social media text types can be further improved by 0.5 up to 0.7 percentage points.

In our approach, we exemplarily use German social media texts. Even though WebTager's basic model and parameter estimation enhancements are language independent, we recommend a language specific training which requires an additional supervised social media text corpus.

5 Information Retrieval Applications

In this chapter, application examples of the proposed social media text related methods in the context of information retrieval are presented. We particularly concentrate our view on the advantages in different scenarios, when a Web crawler is utilized in order to set up a Web page corpus. Note, that we do not explicitly consider the usage of the proposed *WebTagger*, however, the usage of the tagger is implicitly given by calculating POS-based features for any of the considered Web page classification problems. Achieving better POS tagging accuracies on social media texts leads to more reliable POS-based features and implicitly improves Web page classification results.

Two scenarios are considered: (1) The Web page cleaning classifier, proposed in Section 3.7.2, is used in order to achieve a topic relevant Web page corpus. Therefore, a Web crawler starting from topic-relevant seed pages is applied and topic detection based on the cleaned Web pages is performed offline in a postprocessing step to refine the corpus. It is shown that Web page cleaning significantly improves topic classification results for Web pages. (2) In the same way social media text detection proposed in Section 3.7.1 is used in order to set up a social media text corpus, i.e., a corpus of Web pages containing social media texts. A Web crawler starting from different blogs, forums and news sites with posted comments is applied and in a postprocessing step Web pages containing comments are selected from a sample set of resulting Web pages. This results in a social media text corpus, with a high precision rate. For our experiments, we use the open-source crawler *Nutch*.

5.1 Topic Detection

A wide range of information retrieval tasks incorporate the detection of a given topic in a document. Nowadays, the World Wide Web serves as data source in many applications, hence topic detection in Web pages is indispensable. In general a Web page topic is defined by the topic of its main content, e.g., a posted article about pollution on a newspaper Web page. The goal of the following experiments is to show the improved topic detection in a Web page, when first applying the proposed Web cleaning to the Web pages as a preprocessing step. Therefore, we compare two classifiers, one classifying the topic based on the whole Web page content without any preprocessing and one which classifies the topic only based on the main content of the Web page detected by the proposed Web cleaning method. In particular, we perform a detailed per page analysis of such Web pages, which differ in their classification results.

As an exemplary topic, we choose *fracking*, which is a controversially discussed technology. At present, it is easy to find topic-relevant articles and at the same time many topic-related user discussions in the Web, e.g., on news sites or in blogs. For our experiments, we collect a Web page corpus of 100 pages which are all from different domains. The Web page contains, fracking-relevant articles or any kind of fracking-relevant social media texts, e.g., user comments, or a combination of both articles and comments. At the same time, off-topic Web pages are needed to train the classifier. Therefore, we combine the fracking corpus with the 350 off-topic Web pages from different domains, which are not related to the topic fracking.

Based on the combined corpus, two SVM classifiers (Section 2.6.3) are trained for topic detection. A simple bag-of-words model is used to represent a Web page. Bag-of-words models are a common approach in topic classification and they lead to satisfying results for our purposes. In this model, the text of the Web page is represented as the bag of its words, ignoring grammar, POS information or any word orders. However, in order to achieve better keyword matches, we apply word stemming as a preprocessing step and we separate compound words, e.g., *Fracking-Pumpe* (fracking-pump). For each Web page p a feature vector $\mathbf{x}^p \in \mathbb{N}_0^d$ is calculated, where the vector component x_i corresponds to the frequency of a token i in the Web page. The vector dimension d is predefined by the stemmed lexicon size, created from the training Web pages. The lexicon contains all stemmed tokens occurring in any of the training samples, that are in total about 10,000 entries.

5.1.1 Experimental Results

To solve the topic detection problem we use a SVM classifier with a polynomial kernel. Table 5.1 depicts 10-fold cross validation results achieved on the training samples for the two approaches, (1) on the whole Web page and (2) on the identified main content of the Web page. Classification accuracies vary only slightly, when applied to the

	Precision		Recall		Accuracy
	FRACKING	OFF-TOPIC	FRACKING	OFF-TOPIC	
Web page topic detection	96.2	95.6	82.4	99.1	95.7
Web page topic detection on content	100.0	96.2	84.6	100.0	96.8

Table 5.1: Cross validation results achieved for topic detection on the whole Web page and the identified main Web page content.

training corpus. This agrees with our expectations. Due to the fact, that all off-topic Web pages do not contain any topic relevant keywords, even not in the surrounding areas of the main content of Web pages. However, that is not always the case in real applications as the following experiments show. We point out that, in the context of topic-specific Web page corpus construction, the corpus quality can significantly be improved.

Thirteen Web pages of the fracking corpus serve as seed pages for the crawlers search. A crawl with a maximum width of 2,000 pages per layer and a maximum layer depth of

10 is performed. The idea is to access a fracking relevant Web page corpus and achieve an assessable corpus size, which can be analysed manually without too much manual effort. The resulting corpus contains 14,500 fetched Web pages from 285 different domains.

In a post processing step, the resulting corpus is filtered in two steps by applying the topic detection methods. In a first step, the whole Web page content is represented by the bag-of-words feature vector and classified by the corresponding topic classifier into FRACKING or OFF-TOPIC. Overall, 274 Web pages out of the 14,500 are classified as FRACKING. In a second step, these 274 Web pages are further filtered by applying topic detection only on the identified main content of the Web page. Overall, for 55 of such Web pages the classification result differs, where the Web pages are classified as OFF-TOPIC. These Web pages are analysed manually in more detail. Resulting Web pages can be split into two types of Web pages. First, 35% of the analyzed Web pages exhibit a number of topic related links in surrounding areas of the main content of the Web page. An exemplary Web page of this type is shown in Figure 5.1. The black frame marks the main content detected by our Web cleaning method. In the left area of the Web page frame, the topic related terms are marked in gray. It is easy to see, that all topic relevant keywords are outboard of the marked Web page content and hence the Web page is correctly identified as OFF-TOPIC. Second, for 65% of the resulting 55 Web pages no elements are identified as main content and hence are not classified as FRACKING relevant. Such pages are characterized by the fact that the pages are so called hub pages, where a collection of links to the topic, i.e., fracking, related Web pages and no article or any comment is contained. Our Web page cleaning tool is particularly trained for Web pages, where the main content is characterized by an article or listed comments. However, in order to create a corpus comprising topic related articles and social media texts, filtering out such Web pages of the corpus has a positive effect and leads to higher corpus quality. Overall results show that by applying the second filter corpus the FRACKING precision, can be increased by 20 percentage points.

5.2 Social Media Text Corpus Construction

In the previous section topic-relevant texts have been discussed. However, recently the interest in automatic evaluation of social media texts is growing. Particularly, in the context of marketing studies and technology acceptance such texts contain valuable information. However, social media text corpora need to be acquired regularly, in order to perform evaluations on up to date data. Hence, automatic acquisition from the World Wide Web is indispensable. The goal of this experiments is to show that with the proposed social media text classifier from Section 3.7.1 a Web page corpus with posted social media texts can be build. A Web search by means of a Web crawler is applied and Web pages containing social media texts are detected in a postprocessing step from a sample set of Web pages. Therefore, in a first step Web pages from the


NEBELMASCHINE

WWW.NEBELMASCHINE.DE


IM BLOG FÜR ALLE, DIE SICH GERN ALTERNATIV INFORMIEREN UND AUSTAUSCHER MÖCHTEN

JUNG - UNABHÄNGIG - SUBJEKTIV

FÜR DEN BLOG SPENDEN



WEITERSAGEN

Share | 

BLOG-ARCHIV

- ▶ 2014 (3)
- ▶ 2012 (59)
- ▼ 2011 (77)
 - ▼ Dezember (11)
 - Illegale Fans
 - Frehe Weihnachten
 - KenFM - Die Geldsendung
 - Die Yes-Men retten die Welt
 - Fracking - Gefährliche Gier
 - Wir sind wütend
 - Die Pangasius-Lüge
 - Kennen Sie das Iran-Quiz?
 - Iran schießt US - Spionagedrohne vom Himmel
 - Der ausgebeutete Planet - Fracking
 - ▶ November (8)
 - ▶ Oktober (5)
 - ▶ September (7)
 - ▶ August (4)
 - ▶ Juli (5)
 - ▶ Juni (1)
 - ▶ Mai (2)
 - ▶ April (4)
 - ▶ März (6)
 - ▶ Januar (2)
- ▶ 2010 (457)
- ▶ 2009 (12)

SPEZIALITÄTEN

Artikel:

Franchise Everybody

Liebe

Warum tun wir, was wir tun?

SAMSTAG, 10. DEZEMBER 2011

Die Pangasius-Lüge

Das große Geschäft mit dem Billigfisch


40.000 Tonnen Pangasius landeten im vergangenen Jahr allein auf deutschen Tellern. Der Exot aus Asien schont die überfischten Meere, heißt es. Doch Umweltschützer schlagen Alarm: Der Pangasius belastet Tier, Mensch und Umwelt. Was ist dran an den Vorwürfen rund um den beliebten, kostengünstigen Speisefisch? Die Autoren Michael Höft und Christian Jentsch gehen dieser Frage nach. Sie begleiten die Fischexpertin Catherine Zucco von der Umweltorganisation WWF, dem "World Wide Fund For Nature", bei ihrer Recherche in deutschen Supermärkten und im Produktionsland Vietnam.

90 Prozent der Pangasius-Filets stammen aus der Sozialistischen Republik Vietnam - aus dem Mekong Delta am südlichen Zipfel des Landes. In der Provinzhauptstadt Long Xuyen liegt das Zentrum der Pangasius-Industrie. Nur mit Hilfe eines Insiders gelingt es den NDR-Autoren, einen Blick hinter die Kulissen der Großindustrie zu werfen. Auf ihrer Recherchereise entdeckt das Team zahlreiche Missstände: Vom Rinneiten chemisch belasteter Abwässer aus der Fischzucht in den Mekong bis zum Masseninsatz von Antibiotika. Und immer wieder treffen sie auf Aquakulturen, in denen die Fische auf engstem Raum gehalten und gemästet werden.

Das Bild vom Fischerkutler auf der Gefrierpackung mit dem Hinweis auf schnellfließende Gewässer als Produktionsort entpuppt sich als reine Werbefantasie. Die Autoren können auch mit der Legende aufräumen, dass dieser Zuchtisch gepreist sei, die Überfischung der Meere zu stoppen. Das Gegenteil ist der Fall. Als ein Kutter seinen Fring in einer Fischfabrik ablädert, wird das Team Zeuge, wie unterschiedliche Fischsorten aus dem Süß-Christiansleben-Meer zu Pangasius-Futter verarbeitet werden. Vor der Schließung muss der Pangasius ein wahres Martyrium erdulden. 24 Stunden dauert der quälende Transport zur Fischfabrik. Doch nicht nur für Tier und Umwelt hat die Aquakultur in Vietnam Folgen. In den Fischfabriken werden die Pangasius-Filets häufig mit Phosphaten angereichert, damit das Fleisch Wasser speichert und schwerer wird: Ein umstrittenes Verfahren, das auf der Packung deklariert werden muss. Der Kunde erhält nicht nur eine Mogelpackung, was das Gewicht belangt, zu viele Phosphatreste können auch der Gesundheit schaden. Am Ende der Reise gibt es jedoch einen Hoffnungsschimmer: Die Autoren entdecken eine Pangasius-Biofarm, die von einem Deutschen betrieben wird. Die einzige Biofarm des Landes produziert natürlich etwas teurer. Catherine Zucco, die Fischexpertin des WWF, ist überzeugt, dass allein die Verbraucher die Zuchtbedingungen in Vietnam beeinflussen können. Nur wenn sie bereit sind, für einen "sauberen" Pangasius mehr zu bezahlen, werden die Züchter umdenken.


Hier die Doku auf www.ndr.de

Für Vietnam hat der Pangasius eine große wirtschaftliche Bedeutung. Dort haben sich laut Zahlendes WWF die Exporterlöse aus der Zucht binnen zehn Jahren (1997 bis 2007) auf 737 Mio. Dollar (über 550 Mio. Euro) versechsmalndreifacht. Von 2005 bis 2007 erhöhte sich die Ausfuhrmenge von 400.000 auf rund eine Million Tonnen. Auch in Thailand, in Indien und den USA wird der Fisch in großer Menge gezüchtet.

EINGESTELLT VON NEBELMASCHINE.AT.VU UR 15-38  Auf Google empfehlen

LABELS: ERNÄHRUNG, GESUNDHEIT

KOMMENTARE:

 **Anonym** 11. Dezember 2011 um 13:50
 Film entfernt auf Grund einer Regierungsanfrage? Welcher denn? Wahrscheinlich unserem Ministerium für Wahrheit und Verbraucherschutz. Lang lege der große Bruder und sein verdammtes Volk Antworten


 **Anonym** 15. Dezember 2011 um 05:35
 Das ist ein schlechtes Programm <http://www.sciencedirect.com/science/article/pii/S0308597X11001564> Antworten

Figure 5.1: Exemplary Web page with different topic classification results with and without Web cleaning.

sample set are segmented and each segment is classified as COMMENT or NON-COMMENT by applying the classifier.

Additionally, a Web page relevance criteria depending on the segments classification results is defined. Exemplarily, a brute force method is implemented, where Web pages

are considered to be RELEVANT, if at least one segment is classified as COMMENT. Hence, the condition for the relevance classification in the second step is

$$\sum_{i=1}^{N_p} \mathbb{1}_{\{COM\}}(\hat{c}_i^p) \geq 1, \text{ with } \mathbb{1}_{\mathcal{A}}(x) = \begin{cases} 1, & x \in \mathcal{A} \\ 0, & \text{otherwise} \end{cases} \quad (5.1)$$

representing the indicator function for any set \mathcal{A} . However, the criteria can be more complex depending on the requirements. For example a minimum threshold for the estimated probability for the COMMENT class can be set or the number of segments classified as COMMENT could be increased in order to achieve higher corpus quality. All NON-RELEVANT Web pages are filtered out of the Web page corpus.

The Web page corpus is acquired by starting a crawl process from 112 seed pages, which results in 72,000 Web pages from 1,400 different Web domains. The seed pages have been selected manually from 78 different domains, fulfilling one of the two criteria: The Web page is a blog, forum or news site, which contains at least one comment. The Web page is a so called hub page, which contains a high number of links to Web pages, which fulfill the first requirement. For the sample corpus *WPCrawl* 830 Web pages are selected randomly from the larger crawl. The crawl corpus results in a power-law distributed Web page domain ranking (maximum number of Web pages per domain: 18,915, average: 51.26, median: 1). Top-5 hosts are *www.androidpit.de*, *forum.spiegel.de*, *www.focus.de*, *www.macuser.de* and *www.sueddeutsche.de*. Hence, the same holds for the selected sample of 829 Web pages from 104 different domains (maximum number of Web pages per domain: 257, average: 12.75, median: 1). For the evaluation of our classifier the Web page corpus is manually annotated. In contrast to the *GWebTrain* corpus, see Section 3.6 the annotation is performed on Web page level rather than Web segment level. Web pages are labeled by two human annotators as RELEVANT, if it contains at least one social media text. All remaining Web pages are marked as NON-RELEVANT. In total 57% of the Web pages are RELEVANT, which leads to a useful sample set to validate the proposed classifiers.

5.2.1 Experimental Results

For our experiments, we apply the same four classifiers evaluated in Section 3.7.1. Results, achieved with the different classifiers using a model based on $k = 3$ proceeding and succeeding Web text segments, are depicted in Table 5.2 and Table 5.3. RELEVANT precision P_{REL} and total accuracy ACC_{PAGE} are given on the Web page level rather than a text segment. Hence, e.g., P_{REL} is the number of RELEVANT Web pages classified as RELEVANT divided by the total number of Web pages classified as RELEVANT pages. Considering the task of building a social media text corpus by selecting all RELEVANT classified pages, high P_{REL} are particularly important. Best P_{REL} results are achieved with the SVM classifier. In this case 686 Web pages are selected from the original *WPCrawl* corpus, where 77% would be RELEVANT pages. By applying our classifiers combining token-, POS- and HTML-based features the social media text corpus quality can significantly be improved. The amount of

KNN Classifier (k=3)			Decision Tree (k=3)			SVM Classifier (k=3)		
P_{REL}	R_{REL}	ACC_{PAGE}	P_{REL}	R_{REL}	ACC_{PAGE}	P_{REL}	R_{REL}	ACC_{PAGE}
75.78	94.82	79.53	66.67	99.44	70.89	76.82	97.41	81.56

Table 5.2: Social media text detection results validated on the *WPCrawl* corpus.

CRF Classifier (k=3)		
P_{REL}	R_{REL}	ACC_{PAGE}
73.52	93.90	77.00

Table 5.3: Social media text (RELEVANT) detection by applying a CRF validated on *WPCrawl* corpus.

RELEVANT Web pages containing social media texts, could be improved from 57% to 77%(76.82) using a KNN classifier.

6 Conclusions and Future Work

This thesis contributes to sequence labeling tasks in the field of Natural Language Processing by introducing novel concepts, models and algorithms for

- Social media text classification and detection in Web pages,
- Web page cleaning for pages containing social media texts, i.e., social media platforms,
- Part-of-speech tagging for social media texts.

First, we propose social media text classification methods, where sequences of Web text segments are classified in terms of a high-dimensional feature vector consisting of token-, POS- and HTML-based features. New features taking various social media text characteristics into account are proposed and investigated with respect to different classification methods. Particularly, a Conditional Random Field approach with specialized feature functions is implemented. Thereby, sequence labeling problems are solved based on a model with relaxed independent assumptions. Apart from solving the task of social media text classification and detection, we propose a method for Web page cleaning designed for Web pages hosting social media platforms. Good classification performances for both tasks are particularly achieved by providing a representative training corpus consisting of Web pages from social media platforms and an adequate combination of token-, POS- and HTML-based features. This is substantiated by the results of a detailed per feature analysis, where novel social media text related token- and HTML-based features yield high information gain ratios.

Social media texts are detected with precision rates up to 83%, when applying a SVM to the 2-class problem with an extended feature vector by features of three preceding and succeeding text segments ($k=3$). For the more fine-grained classification, where several meta informations such as the user names and the posting times are considered as classes, mean F_1 -Scores up to 65.5% applying the SVM classifier are achieved. This is significantly lower compared to the 2-class problem with 89.1% mean F_1 -Scores, however this a biased comparison, since the differentiation between seven classes is more complex. Reasoned by the relatively small training corpus the SVM classifier outperforms the CRF ($k=3$) approach particularly for the 7-class problem. With the current training corpus size the SVM classifier should be chosen. However, with a bigger training corpus we expect, that the CRF modeling the dependencies between consecutive Web text segments would outperform the SVM classifier.

Web page cleaning methods achieve mean F_1 -Scores up to 92.5% applied to our Web page corpus hosting social media text platforms. This significantly outperforms state-of-the-art approaches by a minimum of 26 percentage points. Applying a Conditional

Random Field model yields the highest recall of 95% on commented areas in Web pages, which is of special interest in our task. Evaluations performed on the English benchmark corpus *CleanEval* show that the trained classifiers are domain-independent and can be transferred to other languages with little effort.

Second, based on a probabilistic Markov model, we have proposed a new POS tagger called *WebTagger*, designed for the annotation of social media texts. The approach mainly differs from other Markov model taggers in the estimation of lexical probabilities for unknown tokens. Therefore, a novel approach mapping unknown tokens to tokens either known from training or tokens which fall into a class represented by regular expressions is presented. For tokens still unknown, we present semi-supervised auxiliary lexica and adequate estimation from combined prefix and suffix information. By doing so, unknown word frequencies are reduced up to a factor of two and tagging accuracies on such tokens are improved. Altogether, lexical probability distributions are estimated more accurately for social media texts. *WebTagger* achieves an average accuracy of 94.7% evaluated on a German social media text corpus consisting of comments from a News site and outperforms state-of-the-art taggers significantly. Additionally, it yields a minimum improvement compared to state-of-the-art taggers of 2.8 percentage points for a social media text type corpus different from the training corpus type. We show that combining sparse in-domain social media training data and a newspaper corpus by an oversampling technique improves POS tagging accuracies significantly. Tagging accuracies can be further improved from 0.5 up to 0.7 percentage points on social media text types.

Finally, it is shown that the proposed social media text detection and Web cleaning methods, as well as the presented POS tagger, can be efficiently used in the context of information retrieval for Web page corpus construction. By applying Web page cleaning and social media text detection to Web page corpora obtained from Web crawlers, the generated corpus can be further refined. Applying our social media text classifier in combination with a simple relevance criteria to Web pages accessed by a crawler, leads to a significant improvement in corpus quality. The amount of relevant Web pages containing social media texts can be increased from 57% to 77% using a SVM classifier.

Future Work

This thesis raises different aspects for future research.

As a first aspect we look at the application of the presented social media text classifier for corpus construction. Very basic applications have been presented in the previous chapter. Social media text detection has been applied as filter in a postprocessing step to a Web page corpus acquired by a Web crawler. However, we assume that Web page corpus results could be improved by integrating the classifier's results in the process of crawling. Considering the classifier's result in the search algorithm could provide better search directions and hence better Web page corpora.

The second aspect is the generality of the proposed POS tagger. *WebTagger*'s basic model and enhancements are language independent. For example, the approach could be transferred to English social media texts. However, due to the lack of English social media text corpora (except those provided by Twitter) the applicability to English texts has not been evaluated in this thesis. Evaluation of the proposed tagger model on English corpora therefore has high potential for future research. Even though several characteristics such as emoticons (:-)) or letter iterations (*Helllloo*) are observed in German and English social media texts alike, some characteristics are language specific. Therefore, some modifications would be necessary to achieve similar improvements in tagging accuracies.

In general, we believe that the models and methods developed in this thesis will provide suitable methodologies for the acquisition and automatic processing of social media texts in challenging future problems.

Acronyms

ACC	Accuracy
BFGS	Broyden-Fletcher-Goldfarb-Shanno
CFS	Correlation-based Feature Selection
CRF	Conditional Random Field
CSS	Cascading Style Sheets
DOM	Document Object Model
HMM	Hidden Markov Model
HTML	Hypertext Markup Language
KNN	<i>K</i> -Nearest Neighbor
L-BFGS	Limited Memory BFGS
MDL	Multi-Interval Discretization for Classification Learning
MEMM	Maximum Entropy Markov Model
MLE	Maximum Likelihood Estimation
NLP	Natural Language Processing
PCA	Principial Component Analysis
POS	Part-Of-Speech
STTS	Stuttgart Tübingen Tagset
SVM	Support Vector Machine
FN	False Negative
FP	False Positive
PR	Precision
RE	Recall
TN	True Negative
TP	True Positive
URL	Uniform Resource Locator
WWW	World Wide Web

List of Symbols

\subseteq, \subset	Subset and proper subset, respectively.
\emptyset	Empty set.
\mathbb{N}, \mathbb{N}_0	Set of positive natural numbers and natural numbers (including zero), respectively.
$\mathbb{R}, \mathbb{R}_{\geq 0}, \mathbb{R}_{> 0}$	Set of real numbers, non-negative real numbers, and positive real numbers, respectively.
$ \cdot $	Cardinality of a set.
c_n	Class label at position n .
c_1^N	Sequence of N class labels.
c'	Subsequence of class labels.
$P(c_1^N \mathbf{x}_1^N)$	Conditional probability of the label sequence c_1^N conditioned on the sequence of observations \mathbf{x}_1^N (Emission probability).
\mathbf{x}, \mathbf{X}	Bold face lower case letters and upper case letters denote column vectors and matrices, respectively.
$\mathbf{x}^T, \mathbf{X}^T$	Transpose of a vector \mathbf{x} and a matrix \mathbf{X} , respectively.
d	Feature vector dimension.
$\mathbf{c}_{\text{start}}$	Sequence of dummy labels c_{start} .
c_{start}	Dummy label added to the set of classes \mathcal{C} .
\hat{c}_n	Estimated class label at position n .
\hat{c}_1^N	Sequence of estimated class labels.
α_k	Step length in the Limited Memory BFGS algorithm.
∇	Gradient.
\mathbf{H}_k	Approximation of the Hesse Matrix in the k -th iteration.
\mathbf{H}_k^0	Initial Hesse matrix approximation.
∇^2	Hesse Matrix.
\mathcal{I}_j	Index set over feature realizations of feature j .
$P(c_1^N, \mathbf{x}_1^N)$	Joint probability of the label sequence c_1^N and the sequence of observations \mathbf{x}_1^N .
$\boldsymbol{\lambda}$	Model parameter vector.
$\ell(\boldsymbol{\lambda})$	Log-likelihood function.
$L(\boldsymbol{\lambda})$	Likelihood function.
c_l^p	Sequence of class labels from position l to p .
\mathcal{L}	Full form lexicon created from training data.

\tilde{x}_j^{\max}	Maximum of feature j over training samples.
$\tilde{\mu}_j$	Mean value of feature j over training samples.
\tilde{x}_j^{\min}	Minimum of feature j over training samples.
n	Position index in a sequence.
N, \tilde{N}	Length of a sequence/training sequence.
K	Number of neighbors considered in K -Nearest Neighbor.
$x_{nj}^{(1)}, x_{nj}^{(2)}$	Differently normalized feature values.
$P(c_1^N)$	Probability of the class sequence c_1^N .
$P(\mathbf{x}_1^N)$	Probability of the observation sequence \mathbf{x}_1^N .
C	Random variable with $C \sim (\tilde{P}(c))_{c \in \mathcal{C}}$ supported on the set of classes \mathcal{C} .
\mathcal{S}_j	Set of realizations of feature j .
\mathcal{R}	Tokens covered by regular expressions.
\mathcal{C}	Set of classes.
\mathcal{C}_s	Set of classes joint with dummy label c_{start} .
\mathbf{s}_n^p	Feature vector at position n of Web page p .
\mathbf{s}_{nk}^p	By k preceding and succeeding feature vectors extended feature vector at position n of Web page p .
\mathbf{S}^p	Sequence of feature vectors of Web page p .
$\tilde{\sigma}_j$	Mean value of feature j over training samples.
\tilde{c}_n	Label of training observation at position n .
\tilde{N}	Length of the training sequence.
t_n	POS tag at position n .
$P(c_n c_1^{n-1})$	Probability of the label c_n conditioned on the sequence of labels c_1^{n-1} (Transition probability).
$\tilde{P}(c_1^N)$	Empirical distribution of the label sequence c_1^N over the training samples.
$\tilde{P}(\mathbf{x}_1^N)$	Empirical distribution of the observation sequence \mathbf{x}_1^N over the training samples.
$(\tilde{\mathbf{x}}_n, \tilde{c}_n)$	Training data pair at position n .
\mathcal{TR}	Set of training data.
\mathcal{T}_w	POS tag set of the word w .
$\tilde{\mathbf{x}}_n$	Feature vector of a training observation at position n .
\mathcal{L}^+	Semi-supervised domain-specific auxiliary lexicon.
\mathcal{V}^+	Semi-supervised verb auxiliary lexicon.
w_n	Token at position n .
\mathcal{W}	All possible tokens.
\mathbf{x}_n	Feature vector of observation at position n .
X_j	Random variable with $X_j \sim (\tilde{P}(x_i))_{i \in \mathcal{I}_j}$ supported on feature j .
\mathbf{x}_1^N	Sequence of N feature vectors.

\mathcal{X}	Full form lexicon extended by regular expressions.
$\mathcal{O}(\cdot)$	Landau symbol, defining an upper bound for the computational complexity.

Bibliography

- [1] L. Armijo, "Minimization of functions having lipschitz continuous first partial derivatives." *Pacific J. Math.*, vol. 16, no. 1, pp. 1–3, 1966. [Online]. Available: <http://projecteuclid.org/euclid.pjm/1102995080>
- [2] P. Awasthi, D. Rao, and B. Ravindran, "Part of speech tagging and chunking with HMM and CRF," in *NLP Association of India (NLP AI) Machine Learning Contest*, 2006.
- [3] M. Baroni, F. Chantree, A. Kilgarriff, and S. Sharoff, "Cleaveval: a competition for cleaning web pages." in *LREC*, 2008.
- [4] J. S. Beis and D. G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," in *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, ser. CVPR '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 1000–. [Online]. Available: <http://dl.acm.org/citation.cfm?id=794189.794431>
- [5] M. Beißwenger, "Corpora zur computervermittelten (internetbasierten) Kommunikation," *Zeitschrift für germanistische Linguistik*, vol. 35, pp. 496–503, 2007.
- [6] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.
- [7] S. Brants, S. Dipper, P. Eisenberg, S. Hansen-Schirra, E. König, W. Lezius, C. Rohrer, G. Smith, and H. Uszkoreit, "TIGER: Linguistic interpretation of a german corpus," *Research on Language & Computation*, pp. 597–620, 2004.
- [8] T. Brants, "Tnt – a statistical part-of-speech tagger," in *Proceedings of the 6th Applied Natural Language Processing Conference*, 2000, pp. 224–231.
- [9] E. Brill, "A simple rule-based part of speech tagger," in *Proceedings of the Third Conference on Applied Natural Language Processing*, 1992, pp. 152–155.
- [10] C. G. Broyden, "The convergence of a class of double rank minimization algorithms: 2. the new algorithm," in *IMA Journal of Applied Mathematics*, vol. 6, 1970, pp. 76–231. [Online]. Available: <http://www.mendeley.com/research/the-convergence-of-a-class-of-doublerank-minimization-algorithms-parts-i-and-ii/>
- [11] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 121–167, Jun. 1998.
- [12] M. Constant and A. Sigogne, "Mwu-aware part-of-speech tagging with a crf model and lexical resources," in *Proceedings of the Workshop on Multiword Expressions: From Parsing and Generation to the Real World*, ser. MWE '11.

- Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 49–56. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2021121.2021134>
- [13] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE Trans. Inf. Theor.*, vol. 13, no. 1, pp. 21–27, Sep. 2006. [Online]. Available: <http://dx.doi.org/10.1109/TIT.1967.1053964>
- [14] L. Derczynski, A. Ritter, S. Clark, and K. Bontcheva, “Twitter part-of-speech tagging for all: Overcoming sparse and noisy data,” in *Proceedings of the International Conference on Recent Advances in Natural Language Processing*. Association for Computational Linguistics, 2013.
- [15] T. G. Dietterich, “Approximate statistical tests for comparing supervised classification learning algorithms,” *Neural Computation*, vol. 10, no. 7, pp. 1895–1923, Jan. 1998.
- [16] J. Dougherty, R. Kohavi, and M. Sahami, “Supervised and unsupervised discretization of continuous features,” in *Machine Learning: Proceedings of the Twelfth International Conference*. Morgan Kaufmann, 1995, pp. 194–202.
- [17] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000.
- [18] S. Evert, “A lightweight and efficient tool for cleaning web pages,” in *Proceedings of the Sixth International Conference on Language Resources and Evaluation*. Marrakech, Morocco: European Language Resources Association (ELRA), May 2008.
- [19] U. M. Fayyad and K. B. Irani, “Multi-interval discretization of continuous-valued attributes for classification learning,” in *International Joint Conference on Artificial Intelligence*, 1993, pp. 1022–1029.
- [20] S. Feldman, M. A. Marin, M. Ostendorf, and M. R. Gupta, “Part-of-speech histograms for genre classification of text,” in *ICASSP*. IEEE, 2009, pp. 4781–4784.
- [21] A. Finn, N. Kushmerick, and B. Smyth, “Genre classification and domain transfer for information filtering,” in *Advances in Information Retrieval*, ser. Lecture Notes in Computer Science, F. Crestani, M. Girolami, and C. Rijsbergen, Eds., vol. 2291. Springer Berlin Heidelberg, 2002, pp. 353–362.
- [22] R. Fletcher, “A new approach to variable metric algorithms,” *The Computer Journal*, vol. 13, no. 3, pp. 317–322, 1970.
- [23] G. Forman, “An extensive empirical study of feature selection metrics for text classification,” *J. Mach. Learn. Res.*, vol. 3, pp. 1289–1305, 2003.
- [24] P. Gadde, L. V. Subramaniam, and T. A. Faruque, “Adapting a WSJ trained part-of-speech tagger to noisy text: Preliminary results,” in *Proceedings of the 2011 Joint Workshop on Multilingual OCR and Analytics for Noisy Unstructured Text Data*, 2011, pp. 5:1–5:8.
- [25] E. Giesbrecht and S. Evert, “Is part-of-speech tagging a solved task? an evaluation of POS taggers for the german web as corpus,” in *Proceedings of the Fifth Web as Corpus Workshop*, 2009, pp. 27–35.

-
- [26] J. Giménez and L. Màrquez, “Svmtool: A general POS tagger generator based on support vector machines,” in *Proceedings of the 4th International Conference on Language Resources and Evaluation*, 2004, pp. 43–46.
- [27] K. Gimpel, N. Schneider, B. O’Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith, “Part-of-speech tagging for twitter: annotation, features, and experiments,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, 2011, pp. 42–47.
- [28] D. Goldfarb, “A family of variable-metric methods derived by variational means,” *Mathematics of Computation*, vol. 24, no. 109, pp. pp. 23–26, 1970.
- [29] M. A. Hall, “Correlation-based feature subset selection for machine learning,” Ph.D. dissertation, University of Waikato, Hamilton, New Zealand, 1998.
- [30] —, “Correlation-based feature subset selection for machine learning,” Ph.D. dissertation, University of Waikato, Hamilton, New Zealand, 1998.
- [31] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA data mining software: An update,” *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009.
- [32] J. M. Hammersley and P. E. Clifford, “Markov random fields on finite graphs and lattices,” Unpublished manuscript, 1971.
- [33] E. Harris, “Information gain versus gain ratio: A study of split method biases,” in *ISAIM*, 2002.
- [34] B. Kessler, G. Numberg, and H. Schütze, “Automatic detection of text genre,” in *Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, 1997, pp. 32–38.
- [35] S. Klein and R. F. Simmons, “A computational approach to grammatical coding of english words,” *J. ACM*, vol. 10, pp. 334–347, 1963.
- [36] R. Klinger and C. M. Friedrich, “Feature subset selection in conditional random fields for named entity recognition,” in *RANLP*, G. Angelova, K. Bontcheva, R. Mitkov, N. Nicolov, and N. Nikolov, Eds. RANLP 2009 Organising Committee / ACL, 2009, pp. 185–191.
- [37] C. Kohlschütter, P. Fankhauser, and W. Nejdl, “Boilerplate detection using shallow text features,” in *Proceedings of the Third ACM International Conference on Web Search and Data Mining*. New York, NY, USA: ACM, 2010, pp. 441–450.
- [38] G. Laboreiro, L. Sarmiento, J. Teixeira, and E. Oliveira, “Tokenizing microblogging messages using a text classification approach,” in *Proceedings of the Fourth Workshop on Analytics for Noisy Unstructured Text Data*, 2010, pp. 81–88.
- [39] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML ’01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 282–289.

- [40] Y.-B. Lee and S. H. Myaeng, "Text genre classification with genre-revealing and subject-revealing features," in *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 2002, pp. 145–150.
- [41] C. S. Lim, K. J. Lee, and G. C. Kim, "Multiple sets of features for automatic genre classification of web documents," *Inf. Process. Manage.*, vol. 41, no. 5, pp. 1263–1276, 2005.
- [42] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: The MIT Press, 1999.
- [43] M. Marek, P. Pecina, and M. Spousta, "Web page cleaning with conditional random fields," in *Building and Exploring Web Corpora (WAC3 - 2007), Proceedings of the 3rd Web as Corpus Workshop, Incorporating CleanEval*, 2007.
- [44] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *In AAAI-98 Workshop on Learning for Text Categorization*. AAAI Press, 1998, pp. 41–48.
- [45] S. Meyer zu Eissen and B. Stein, "Genre classification of web pages: User study and feasibility analysis," in *KI 2004: Advances in Artificial Intelligence*. Springer Berlin Heidelberg, 2004, pp. 256–269.
- [46] S. E. Michos, E. Stamatatos, N. Fakotakis, and G. Kokkinakis, "An empirical text categorizing computational model based on stylistic aspects," in *Proceedings of the 8th International Conference on Tools with Artificial Intelligence*, ser. ICTAI '96. Washington, DC, USA: IEEE Computer Society, 1996, pp. 71–.
- [47] A. Mikheev, "Automatic rule induction for unknown word guessing," *Computational Linguistics*, vol. 23, pp. 405–423, 1997.
- [48] C. Nadeau and Y. Bengio, "Inference for the generalization error," *Machine Learning*, vol. 52, no. 3, pp. 239–281, 2003.
- [49] S. A. Nene and S. K. Nayar, "A simple algorithm for nearest neighbor search in high dimensions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 9, pp. 989–1003, Sep. 1997. [Online]. Available: <http://dx.doi.org/10.1109/34.615448>
- [50] M. Neunerdt, E. Reimer, M. Reyer, and R. Mathar, "Enhanced web page cleaning for constructing social media text corpora," in *6th International Conference on Information Science and Applications (ICISA)*, Pattaya, Thailand, Feb. 2015, pp. 1–8.
- [51] M. Neunerdt, M. Reyer, and R. Mathar, "Part-of-speech tagging for social media texts," in *Proceedings of The International Conference of the German Society for Computational Linguistics and Language Technology*, 2013.
- [52] —, "A POS tagger for social media texts trained on web comments," *Polibits*, vol. 48, pp. 59–66, 2013.
- [53] —, "Automatic genre classification in web pages applied to web comments," in *12th Conference on Natural Language Processing*, Hildesheim, Germany, Oct. 2014, pp. 145–151.

- [54] —, “Efficient training data enrichment and unknown token handling for POS tagging of non-standardized texts,” in *12th Conference on Natural Language Processing*, Hildesheim, Germany, Oct. 2014, pp. 145–151.
- [55] M. Neunerdt, B. Trevisan, R. Mathar, and E.-M. Jakobs, “Detecting irregularities in blog comment language affecting POS tagging accuracy,” *International Journal of Computational Linguistics and Applications*, vol. 3, no. 1, pp. 71–88, Jun. 2012.
- [56] M. Neunerdt, B. Trevisan, M. Niermann, and R. Mathar, “Focused crawling for building web comment corpora,” in *The 10th IEEE Consumer Communications & Networking Conference CCNC 2013*, Las Vegas, Nevada USA, 2013, pp. 676–679.
- [57] M. Neunerdt, B. Trevisan, T. C. Teixeira, R. Mathar, and E.-M. Jakobs, “Ontology-based corpus generation for web comment analysis,” in *Proceedings of The ACM Conference on Hypertext and hypermedia*, Eindhoven, May 2011.
- [58] A. Y. Ng and M. I. Jordan, “On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes,” in *Advances in Neural Information Processing Systems 14*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds. MIT Press, 2002, pp. 841–848. [Online]. Available: <http://papers.nips.cc/paper/2020-on-discriminative-vs-generative-classifiers-a-comparison-of-logistic-regression-and-naive-bayes.pdf>
- [59] J. Nocedal, “Updating quasi-newton matrices with limited storage,” *Mathematics of Computation*, vol. 35, no. 151, pp. 773–782, 1980.
- [60] O. Owoputi, B. O’Connor, C. Dyer, K. Gimpel, and N. Schneider, “Part-of-speech tagging for twitter: Word clusters and other advances,” School of Computer Science, Carnegie Mellon University, Tech. Rep., 2012.
- [61] O. Owoputi, B. O’Connor, C. Dyer, K. Gimpel, N. Schneider, and N. A. Smith, “Improved part-of-speech tagging for online conversational text with word clusters,” in *Proceedings of NAACL-HLT*, 2013, pp. 380–390.
- [62] L. Pelayo and S. Dick, “Applying novel resampling strategies to software defect prediction,” in *Fuzzy Information Processing Society, 2007. NAFIPS ’07. Annual Meeting of the North American*, Jun. 2007, pp. 69–72.
- [63] C. Potts, “Sentiment symposium tutorial: Tokenizing,” 2011, stanford Linguistics, USA.
- [64] X. Qi and B. D. Davison, “Web page classification: Features and algorithms,” *ACM Comput. Surv.*, vol. 41, no. 2, pp. 12:1–12:31, 2009.
- [65] J. R. Quinlan, “Induction of decision trees,” *Machine Learning*, pp. 81–106, 1986.
- [66] —, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.
- [67] A. Ratnaparkhi, “A maximum entropy model for part-of-speech tagging,” in *Proceedings of the Empirical Methods in Natural Language Processing*, 1996, pp. 133–142.

- [68] I. Rehbein, “Fine-grained POS tagging of german tweets,” in *Language Processing and Knowledge in the Web*, ser. Lecture Notes in Computer Science, vol. 8105. Springer Berlin Heidelberg, 2013, pp. 162–175.
- [69] R. Remus, U. Quasthoff, and G. Heyer, “SentiWS – a publicly available german-language resource for sentiment analysis,” in *Proceedings of the 7th International Language Resources and Evaluation (8LREC’10)*, 2010, pp. 1168–1171.
- [70] E. Riloff, J. Wiebe, and W. Phillips, “Exploiting subjectivity classification to improve information extraction,” in *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3*, ser. AAAI’05. AAAI Press, 2005, pp. 1106–1111.
- [71] M. Santini, “A shallow approach to syntactic feature extraction for genre classification,” CLUK 7: The UK special-interest group for computational linguistics,” in *Proceedings of the 7th Annual Colloquium for the UK Special Interest Group for Computational Linguistics*, 2004.
- [72] A. Schiller, S. Teufel, C. Stöckert, and C. Thielen, “Guidelines für das Tagging deutscher Textcorpora mit STTS,” 1999, university of Stuttgart.
- [73] H. Schmid, “Part-of-speech tagging with neural networks,” in *Proceedings of the 15th Conference on Computational Linguistics*, 1994, pp. 172–176.
- [74] —, “Probabilistic part-of-speech tagging using decision trees,” in *Proceedings of International Conference on New Methods in Language Processing*, 1994, pp. 44–49.
- [75] —, “Improvements in part-of-speech tagging with an application to german,” in *Proceedings of the ACL SIGDAT-Workshop*, 1995, pp. 47–50.
- [76] H. Schütze, “Distributional part-of-speech tagging,” in *Proceedings of 7th Conference of the European Chapter of the Association for Computational Linguistics*, 1995, pp. 141–148.
- [77] F. Sebastiani, “Machine learning in automated text categorization,” *ACM Comput. Surv.*, pp. 1–47, 2002.
- [78] D. F. Shanno, “Conditioning of quasi-newton methods for function minimization,” *Mathematics of Computation*, vol. 24, no. 111, pp. pp. 647–656, 1970.
- [79] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks,” *Information Processing and Management*, vol. 45, no. 4, pp. 427–437, 2009.
- [80] M. Spousta, M. Marek, and P. Pecina, “Victor: the web-page cleaning tool.” in *Proceedings of the 4th Web as Corpus Workshop, LREC*, 2008.
- [81] E. Stamatatos, N. Fakotakis, and G. Kokkinakis, “Text genre detection using common word frequencies,” in *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, ser. COLING. Stroudsburg, PA, USA: Association for Computational Linguistics, 2000, pp. 808–814.

-
- [82] E. Stamatatos, G. Kokkinakis, and N. Fakotakis, “Automatic text categorization in terms of genre and author,” *Comput. Linguist.*, vol. 26, no. 4, pp. 471–495, 2000.
- [83] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, “Feature-rich part-of-speech tagging with a cyclic dependency network,” in *Proceedings of Human Language Technology Conference*, 2003, pp. 173–180.
- [84] K. Toutanova and C. D. Manning, “Enriching the knowledge sources used in a maximum entropy part-of-speech tagger,” in *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 2000, pp. 63–70.
- [85] B. Trevisan and E.-M. Jakobs, “Bewerten in Blogkommentaren : Mehrebenenannotation sprachlichen Bewertens,” Ph.D. dissertation, RWTH Aachen University, Aachen, 2014, aachen, Techn. Hochsch., Diss., 2013. [Online]. Available: <http://publications.rwth-aachen.de/record/444868>
- [86] B. Trevisan, M. Neunerdt, and E.-M. Jakobs, “A multi-level annotation model for fine-grained opinion detection in german blog comments,” in *11th Conference on Natural Language Processing (KONVENS)*, Vienna, Austria, Sep. 2012, pp. 179–188.
- [87] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.
- [88] —, *Statistical Learning Theory*. Wiley-Interscience, 1998.
- [89] A. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *IEEE Trans. Inf. Theor.*, vol. 13, no. 2, pp. 260–269, Sep. 2006. [Online]. Available: <http://dx.doi.org/10.1109/TIT.1967.1054010>
- [90] M. Volk and G. Schneider, “Comparing a statistical and a rule-based tagger for german,” in *Proceedings of the 4th Conference on Natural Language Processing*, 1998, pp. 125–137.
- [91] J. Wiebe, “Learning subjective adjectives from corpora,” in *xxxx*, 2000.
- [92] J. Wiebe, T. Wilson, R. Bruce, M. Bell, and M. Martin, “Learning subjective language,” *Comput. Linguist.*, vol. 30, no. 3, pp. 277–308, Sep. 2004.
- [93] P. Wolfe, “Convergence conditions for ascent methods,” *SIAM Review*, vol. 11, no. 2, pp. pp. 226–235, 1969. [Online]. Available: <http://www.jstor.org/stable/2028111>
- [94] Y. Yang and J. O. Pedersen, “A comparative study on feature selection in text categorization,” in *Proceedings of the Fourteenth International Conference on Machine Learning*, ser. ICML ’97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 412–420. [Online]. Available: <http://dl.acm.org/citation.cfm?id=645526.657137>

Curriculum Vitae

Melanie Neunerdt

Born on the 28th of May in 1982 in Essen, Germany

Education

1994 - 1998	Elementary school in Essen, Germany
1998 - 2001	Academic highschool in Essen, Germany
3rd of June 2001	General qualification for university entrance (Abitur)
2001 - 2004	Apprenticeship as Mathematical-technical assistant, RWTH Aachen University
2005 - 2006	Diploma in Mathematics, FH Aachen
2006 - 2009	Master in Software Systems Engineering, RWTH Aachen University
2010- 2015	PhD in Electrical Engineering, Chair for Theoretical Information Technology, RWTH Aachen University

Professional Experience

2004 - 2006	Mathematical-technical assistant, Chair of Electrical Engineering and Computer Systems, RWTH Aachen University
2006 - 2010	Research assistant, Chair for Theoretical Information Technology, RWTH Aachen University
2010 - 2015	Research assistant and PhD student, Chair for Theoretical Information Technology, RWTH Aachen University

