Feature Selection in Aggressive Comment Detection

Tina Schuh

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im September 2017

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, September 18, 2017

Tina Schuh

Contents

Preface

Ernest Hemingway once said,

> *The first draft of anything is shit.*

His words, with regards to the implemented aggressive language model, are a perfect example of a *false positive.* On a more personal level, this prominent quote stuck with me as my mantra throughout the process of writing and completing my thesis. Thanks to the valuable and consistently constructive feedback of my advisor, Stephan Dreiseitl, this work has moved beyond the dreadful first draft.

I am also deeply grateful to my family. First and foremost, to my husband, Martin Schuh, for his unwavering support, through thick and thin. To my daughter, Marlene Schuh, for her daily pep talk as I dropped her off at daycare: *"Off you go to work, Mom."* To my parents(-in-law), Eva and Ernst Schwaiger, Helga and Alois Schuh, for pitching in, whenever balancing work and family failed—as it often did.

Abstract

Offensive comments or hate speech pose an increasing problem in the online world. In order to address these uncivil comments in online discussions, it is necessary to identify them first. However, this identification process is a time consuming and arduous task if handled manually.

Therefore we develop an aggressive language model for text classification that can be used as a means to automatically detect hostile user comments. While most automated approaches rely on n-grams of some kind, this work aims to find linguistic patterns that are independent of the exact vocabulary of a given corpus: A variety of 68 features—based on vocabulary, grammar, punctuation, style, diversity, and basic statistical observations—are extracted from annotated user-comments; in order to gain an accurate and compact model, the features are then ranked by their singular statistical relevance or evaluated together using a learning algorithm to search for optimal subsets. We employ the T-test, the Wilcoxon-ranksum-test and Mutual Information as feature rankers and two learners, Recursive Feature Elimination and the Lasso, to find favorable subsets. The learners' subsets and the subsets derived from the rankings are evaluated using three different machine learning models: Logistic Regression, Linear SVM, and Gaussian SVM. Each model is trained on three-quarters of the data and evaluated against the held-out quarter; the hyperparameters of the SVMs are tuned using 5-fold cross-validation.

We independently apply the feature selection and model evaluation process to two unrelated datasets, both with a balanced distribution of aggressive and non-aggressive user comments. The results show that depending on the dataset and model type, between 3 and 19 features are enough to outperform the full set of 68 features. The overall best F_1-scores per dataset are 89.4% (with a precision of 93% and recall of 86.1%), using 40 features with a Gaussian SVM and 82.7% (with a precision of 85.1% and recall of 80.5%), using 17 features with a linear SVM.

By comparing the top models from both datasets, we derive a core set of features that are most useful in predicting whether a comment is aggressive or not; the most informative features are the number of blacklisted words, the number of long words and the ratio of negative subjective words.

Kurzfassung

Beleidigende, hasserfüllte Kommentare zerstören zunehmend das Gesprächsklima in den Diskussionräumen des Internets. Das manuelle Moderieren von Benutzerkommentaren ist mit viel Mühe und Zeitaufwand verbunden; in Zukunft könnten Moderatoren durch eine automatische Klassifizierung unterstützt werden.

Während die meisten Lösungsansätze, die sich des maschinellen Lernens bedienen, auf N-Gramme in verschiedenen Ausprägungen zurückgreifen, widmet sich diese Arbeit sprachlichen Mustern die unabhängig vom spezifischen Vokabular eines gegebenen Textkorpus sind: Auf der Suche nach einem möglichst korrekten und kompakten Modell für aggressive Sprache werden 68 vielfältige Merkmale *(Features)* – basierend auf Vokabular, Grammatik, Punktuation, Stil, Sprachdiversität und einfachen Statistiken – von annotierten User-Kommentaren extrahiert und nach statistischer Relevanz gereiht oder mittels Lernalgorithmen möglichst effektiv kombiniert. Auf Basis des T-Tests, des Wilcoxon-Rangsummentests und der gegenseitigen Information (engl. *Mutual Information*) werden Feature-Ranglisten erstellt; mittels zweier Lernalgorithmen, Recursive Feature Elimination und Lasso, werden verschiedene Feature-Teilmengen generiert. Sowohl diese Teilmengen, als auch Teilmengen, die anhand der Ranglisten hergeleitet sind, werden mithilfe folgender Algorithmen aus dem maschinellen Lernen evaluiert: Logistische Regression, lineare SVM und nicht-lineare SVM (mit radialer Basisfunktion als Kern-Funktion). Drei Viertel des Datensatzes werden verwendet um die einzelnen Modelle zu trainieren, das restlichen Viertel dient zur Evaluierung; die Ermittlung der Hyperparamter je SVM erfolgt durch fünffache, stratifizierte Kreuzvalidierung.

Anhand zweier unabhängiger Korpora von Benutzerkommentaren werden effektive Featurekombinationen ermittelt und die daraus abgeleiteten Modelle evaluiert. Die Resultate zeigen, dass Teilmengen von 3 bis 19 Features ausreichend sind, um ein besseres Ergebnis zu erzielen als mit allen 68 Features – in Abhängigkeit von Datensatz und Modelltyp. Die insgesamt besten Modelle je Datensatz erzielen einen F_1-Score von 89.4% (Precision: 93%, Recall: 86.1%), unter Verwendung von 40 Features mit nicht-linearer SVM und 82.7% (Precision: 85.1%, Recall: 80.5%), unter Verwendung von 17 Features und linearer SVM.

Auf Basis der Spitzenmodelle beider Datensätze werden schlussendlich die einflussreichsten Features zur Unterscheidung von sachlichen und aggressiven Kommentaren hergeleitet: Die Anzahl der Wörter, die auf der schwarzen Liste stehen, die Anzahl der langen Wörter und der Anteil negativer, subjektiver Wörter.

Chapter 1

Introduction

Do not read the comments.
—unknown

This quote, a piece of contemporary wisdom, addresses the human reflex of scrolling beyond an online article, into an all too often hostile abyss of aggression, disrespect, and harassment—the comment section. What makes it hard to resist the urge to peek anyway—besides morbid curiosity—is that, on the other hand, reader comments can be a valuable source of additional information, entertainment, and discussion; they enable readers to voice concerns or ask questions. Moreover, if you are a moderator in an online community team, a journalist, a blog author or just generally interested in other people's thoughts on a topic, ignoring the comments is not a viable option.

1.1 Motivation

According to a recent Pew Report on online harassment [16], 62% of American internet users consider online harassment a major problem. The survey shows that

> [...] *41% of Americans have been personally subjected to harassing behavior online, and an even larger share (66%) has witnessed these behaviors directed at others.*

Besides the emotional stress and damage, the conversations' aggressive tone silences possibly more reasonable and valuable voices. 27% of users who had witnessed harassment of others subsequently decided not to post something online. One participant of the survey stated:

> *Like an adult, I turned off the computer and walked away. No one is forcing me to be online.*

Withdrawal might be a positive resolution for an individual, but for society at large this is a considerable loss; it entails a distortion of public debate and empowers the hateful voices. This is indeed a matter of far-reaching concern: As the issues of our time are increasingly discussed and negotiated online, the prevailing sentiments influence political proceedings [2]. Anderson et al. [1] show that uncivil comments have a polarizing effect on the readers; in effect, readers become less open to different views. The researchers call this phenomenon the *Nasty Effect*, and Brodnig [2, pp. 76–80] illustrates how trolls[1] strategically use it as a means to sabotage and prevent any factual, productive discussion. Pointing out how these attacks on public debates have the potential to erode our democracy, she emphasizes the need for online spaces that do not tolerate abusive language.

Taking Action

While 60% of Americans feel that those who witness others being harassed should step in, 79% assign the responsibility of handling harassment to the online platform where the issue occurs [16]. In June 2017, Germany passed a law that forces social media platforms to remove illegal, defamatory or racist posts within 24 hours and handle content which has been flagged as offensive within seven days. Earlier this year, *Facebook* announced it would increase the number of employees worldwide, to 7 500, to deal with flagged postings [17].

The task of manually moderating these disruptive, and often extremely hateful comments is time-consuming and not applicable in real-time—especially given the sheer amount of content that is being posted perpetually. Besides, the job is often frustrating and demoralizing and may have harmful consequences on the moderators' health [8].

Behind the scenes, major tech firms are already working on a more efficient approach; in the future, automated classification might curb the problem: Early in 2017, *Google* introduced *Perspective*, a machine learning tool that automatically identifies abusive comments [18]; *The New York Times* already uses the software to assist moderators in the comment approval process [26]. *Twitter*, which has been criticized for years for consistently failing to deal with hate speech, has decided to take more action shortly after the US presidential election of 2016 [15] and has since added an automated classification system to block abusive tweets and user accounts to their tool belt [20]. *Instagram* has recently launched a language filter that automatically blocks offensive comments [27]: The filter is based on Facebooks *DeepText*, a tool for text classification tasks that utilizes deep neural networks.[2] While *Facebook* still relies on human moderators, Instagram's tool is likely a test case for Facebook's future moderation approach [25].

1.2 Research Question

This thesis investigates the properties of aggressive language in online comments and user discussions. Its aim is to identify the most characteristic patterns in aggressive language that set it apart from non-inflammatory, factual language. Using a data-driven approach, we first collect aggressive and non-aggressive user comments, and then, we extract a wide range of statistical and linguistic properties—so called *features*—from each comment. Examples of such features are the length of the comment in words,

[1]The top definition of the term *troll*, in the Urban Dictionary, is: "One who posts a deliberately provocative message to a newsgroup or message board with the intention of causing maximum disruption and argument."

[2]Artificial Neural Networks are inspired by biological brains; they use artificial neurons which mimic how a brain processes information.

sentences, and paragraphs; the number of imperative sentences; lexical and grammatical diversity; repetition of punctuation; the number of blacklisted words; and the use of subjective words. We use these features to build a model of aggressive language by letting a machine learning algorithm learn the patterns in the data. The trained model enables us to classify new data as either aggressive or not aggressive; however, the accuracy of the model depends on the kind of features the model uses. This leads to the key question of our research:

What are the most effective features in creating a model for automated detection of aggressive language in online user comments?

We employ feature selection techniques to gain insights into which features are best suited to separate the two comment categories. Likewise, feature selection guides us in finding favorable feature subsets, also showing that more features do not guarantee better results.

1.3 Structure

Chapter 2 summarizes the concepts of natural language processing and machine learning and provides a short definition of the technical terms needed to understand this thesis. Chapter 3 dives into the history of offensive behavior online, presents a selection of research that has been undertaken so far to automatically detect offensive language (section 3.2) and describes what makes this task so difficult (section 3.3). Chapter 4 starts out with an overview of the classification process and technical design and continues with a detailed description of the data used for training and testing (section 4.2), the full set of extracted features (section 4.3), how feature selection is performed (section 4.4), the utilized machine learning algorithms (section 4.5) and the metrics that were used to measure the models' performance (section 4.6). The implementation details are to be found in chapter 5, which includes, among others, the use-case-specific approach to word tokenization and text normalization, as well as the developed imperative detection algorithm (section 5.2). The feature selection results and classification scores are depicted in chapter 6. Finally, in chapter 7, we summarize the results of our work, highlighting the most discriminatory properties of aggressive user comments.

Chapter 2

Technical Background

2.1 Natural Language Processing

Natural Language Processing (NLP) is a multidisciplinary field that overlaps with Artificial Intelligence, Computer Science, and Computational Linguistics. In recent years it has been driven by advances in Machine Learning, shifting the paradigm from hand coded rules to automatically learning rules through the statistical analysis of corpora.

NLP is concerned with translating human language—written or spoken—into a representation that can be understood and handled by a computer. Though today's systems are still struggling with major challenges such as ambiguity or the conceptual dimension of human speech, the application of NLP is already ubiquitous, ranging from spam filtering, topic based email classification (e.g., *Gmail*) and predictive typing (e.g., *SwiftKey*) to sophisticated grammar-correction (e.g., *Grammarly*) and smart assistants (e.g., *Siri, Alexa*).

In general, NLP tasks deal with *syntax* (e.g., part-of-speech tagging, grammatical parsing), *semantics* (e.g., word sense disambiguation, named entity recognition), *discourse* (e.g., coreference resolution) and *speech* (e.g., speech recognition and segmentation).

2.2 Machine Learning

Machine Learning (ML) is a particular approach within the field of Artificial Intelligence. It is closely connected to data mining and statistics. Learning from data is the core idea of Machine Learning; making predictions or decisions based on *training data* is its main application.

In *supervised learning*, the learning of a prediction model is guided by input data with known outcomes. If the output variables are categories, such as *healthy* or *sick*, the task is called *classification*. Real valued output, such as *centimeters* or *euros*, is the result of a *regression* task.

Unsupervised learning has no pre-defined outcomes. Instead, the algorithm tries to cluster the input samples into differents groups. It is a means to detect underlying patterns and associations in the input data.

2.3 Terminology

The following glossary briefly summarizes concepts of natural language processing and machine learning that are relevant to this work.

Corpus
: A corpus is a large collection of texts that find use in linguistic analysis. A corpus can consist of books, news articles, speech transcription, product reviews, etc. *Annotated corpora* provide additional linguistic information besides the raw text.

Dataset
: In general the term *dataset* refers to a collection of any kind of related data. In this thesis, a dataset either refers to a corpus, or data that is used to train or test a machine learning model. In machine learning, the dataset consists of a 2-dimensional matrix, where the rows are the samples (feature vectors) and the columns represent the features. In supervised learning, the term also encompasses the target array containing the outputs per sample.

Tokenization
: Tokenization is the task of chopping a character sequence into words, numbers and punctuation, so called *tokens.* The tokenized version of the statement *"Don't panic."* is `['Do', "n't", 'panic', '.']`. The task of separating an input text into its sentences is refered to as sentence tokenization.

Bag-of-Words
: In the bag-of-words model, a text document is viewed as a *bag* containing an unordered collection of words. In this bag, grammar, word order and semantic relationships do not matter. The only information of interest are the words themselves and their frequency of occurrence. These frequencies are the features used for training a classifier. The words in the bag are not necessarily unigrams; the same concept can be applied to n-grams of any type.

***n*-grams**
: In computational lingustics, n-grams usually refer to a sequence of n words (or tokens). Unigrams contain only one word; sequences of two tokens are called bigrams and sequences with three tokens are trigrams. The text *'not all those who wander are lost'* converted to trigrams is `['not all those', 'all those who', 'those who wander', 'who wander are', 'wander are lost']`

Character *n*-grams
: Character n-grams are n-grams at the character level. The slogan *'Yes, we can!'* as a 5-gram sequence is `['Yes,_', 'es,_w', 's,_we', ',_we_', '_we_c', 'we_ca', 'e_can', '_can!']` (here _ denotes a space).

Part-of-Speech Tags (POS-Tags)
: POS-tags are labels that refer to grammatical categories; each token of a sentence is assigned such a tag. A tagged version of the line *"In the end, it doesn't even matter."* looks like `In/IN the/DT end/NN ,/, it/PRP does/VBZ n't/RB even/RB matter/VB ./.`, where `IN` stands for preposition, `DT` for determiner, `NN` for singular noun, `PRP` for personal pronoun, `VB*` for various verb forms and `RB` for adverb.[1]

[1] For the full list of tags visit http://web.mit.edu/6.863/www/PennTreebankTags.html.

Stemming

Stemming is the task of normalizing a word to its common base form, the stem (or root), by removing the words affixes. For instance, the words *rains, raining, rained* will be trimmed to *rain*; the word *rainbow*, however, remains the same.

Stop Words

The term *stop words* refers to a list of highly frequent words that carry little meaning. Typically, in the bag-of-words approach these words are filtered out before the text data is processed. *A, and, at, in, it, that* and *the* are examples of English stop words.

Feature

A feature (also: predictor, attribute, variable) is the value measured (or derived) from a characteristic of the raw input. E.g., for the input comment *'Shit happens.'*, a feature representing the number of blacklisted words would have the value `1` because that comment contains exactly one blacklisted word. Features are either *discrete* (e.g., counts), *continuous* (e.g., ratios) or *categorical* (limited, fixed values; e.g., blood types).

Feature Vector

A feature vector (also: observation, sample) is a collection of numeric features that were extracted from one single sample. It appears in the form of a one dimensional matrix.

Feature Extraction

A machine learning model expects input data in the form of a matrix containing numeric values. When the original raw data cannot be directly fed into a model (e.g., a text or an image), characteristics of the raw data are translated into numeric form, this process is called feature extraction.

Feature Engineering

Feature engineering is the process of transforming features into more meaningful representations to improve the model. This task requires domain knowledge and manual effort.

Feature Selection

The aim of feature selection is to find an appropriate subset of features that approximates or surpasses the results of the full set of features. In automatic feature selection, the features are either evaluated individually (using feature rankings) or in combination with others (using a learning algorithm).

Target

The target (also: label, response, outcome) represents the class of a sample. It is encoded in numeric form (e.g., `0` represents OK, `1` represents *aggressive*).

Evaluation Metrics

To measure the performance of a classification model, predictions are made for a held out test set. The comparison of the predicted labels versus the actual labels can be expressed in four categories: *True Negatives* (e.g., the number of OK comments that were labeled as OK), *False Negatives* (e.g., the number of aggressive comments that were labeled as OK), *False Positives* (e.g., the number of OK comments that were labeled as aggressive) and *True Positives* (e.g., the number of aggressive comments that were labeled as aggressive).

Chapter 3

State of the Art

Offensive behavior in online discussions is not a new phenomenon. Initially referred to as *flaming*[1], it was observed as early as the 1970s in e-mail communication and its rise continued with the birth of Usenet in the early 1980s [19]. As the entry barriers to owning and using the necessary technology to communicate online are continuously lowered, the number of users—and consequently the amount of user-generated content, both legitimate and inflammatory,—is rapidly growing.

Similar to spam-filtering, a first automated approach to detect these disruptive interactions was a simple keyword spotting task using a blacklist. While blacklists are popular due to their simplicity and effectiveness, they often have a high false positive rate and fail to detect more subtle forms of offensive content [5]. Newer attempts rely on machine learning techniques for more accurate classification. The Naive Bayes classifier is regarded a classic approach, borrowed again from spam filtering: It typically employs bag-of-words features to calculate the likelihood of a comment being offensive.

After a short description of the scope of *offensive language* this chapter outlines past research on automated detection of offensive language, and closes with the challenges that arise when developing such a system.

3.1 Definitions

Aggressive language falls into the category of highly subjective language as it expresses emotions (e.g., hate, anger), judgments or opinions [10]. Definitions for the spectrum of offensive language range from

- hate speech, profanity, derogatory language [9],
- abusive speech targeting specific group characteristics (such as ethnic origin, gender or religion) [4, 5, 10, 12, 13],
- harassment, personal attacks, toxic behaviour [13],
- the main intent being to attack [10] to
- content that is hateful, threatening, pornographic, vulgar, incites violence [5, 12].

[1]flame: to send an angry, hostile, or abusive electronic message

3.2 Offensive Language Detection

Most research prototypes make use of a lexicon [5, 9–12] in which the entries are labeled for polarity and often have weights attached to grade the level of offensiveness. Another popular predictor are word-dependencies, with the simplest form being n-grams ($n > 1$); more elaborate methods employ phrase patterns with wildcards, identify words as intensifiers for other offensive words (e.g., '*fucking* stupid') or capture long-range dependencies by making use of POS-Tags and a dependency parser [4, 5, 9–12].

The earliest work on automated classification of offensive text is mailbox filter *Smokey* [11], which addressed private flame messages sent to a few selected political websites. The features are derived from rules that check for hostile as well as polite language. Detection rate of hostile messages is 64%. From the tested *okay* messages 98% were labeled correctly (ratio *okay* to *flames*: 4,5:1). The rule for epithets[2] proved to be one of the most useful. Spertus used a *C4.5 decision-tree generator (Quinlain)* to generate a classifier. Linear regression was tested too but turned out less successful.

Razavi et al. [10] implemented a flame detection system (accuracy 91%) including their own Insulting or Abusive Language Dictionary (IALD). Phrases could contain wild cards (e.g., *'Ball* `[Somebody]` *up'* where `[Somebody]` is the wild card) to catch different variations of a phrase. Their system utilizes a three-level classification on the preprocessed data: Complement Naive Bayes classifier, Multinomial Updatable Naive Bayes classifier and a rules based classifier ("Decision Table/Naive Bayes hybrid classifier") based on the IALD. The system works on the document level as well as on the sentence level.

Chen et al. [5] propose a user-level offensiveness system (as opposed to most solutions that operate on message- or sentence-level only) additionally to sentence offensiveness prediction. Their Lexical Syntactical Feature (LSF) approach uses a two-level offensive word lexicon, grammatical parsing to find dependencies between words, evaluates punctuation, all uppercase letters, and checks whether a sentence is in the imperative form. For better results, the data is preprocessed with automatic spelling and grammatical correction. Their system achieves a precision of 98.24% and recall of 94.34% in detecting offensive sentences. According to their test results, style features (punctuation, all uppercase letters) and cyberbullying-related content features (e.g., occurrence of words relating to religion, race, intelligence) yielded the most valuable improvements. Naive Bayes (NB) and Support Vector Machines (SVM) were used for classification, noting that NB was much faster than SVM, whereas SVM proved to be more accurate.

Burnap and Williams [4] wrote a classifier for hate speech in tweets, achieving an overall F_1-score of 0.95, using Bayesian Logistic Regression, a Random Forest Decision Tree and SVM in combination. They use a bag-of-words model with n-grams ranging from 1 to 5 tokens. They also use syntactic features that capture the dependencies between words, with the addition of the dependency type (e.g., `dobj(burn-6 korans-9)`, where `dobj` stands for Direct Object and the numbers relate to words' index within the sentence).

Warner and Hirschberg [12] are targeting anti-semitic speech by using a combination of n-grams, a mix of POS-Tags and n-grams (e.g., the chunk `DT (determiner) +`

[2]epithets: short insulting phrases (e.g., 'get a life')

`'jewish' + NN (noun)`, which was the strongest indicator for a positive, non-abusive message). They achieved a precision of 68%, recall of 60% and F_1-score of 0.64 using SVM.

Nobata et al. [9] point out that past research on abusive language detection spreads several fields and lacks a unified approach as well as a common testing set for comparison. One of their main contributions is a corpus annotated by *Yahoo* employees for evaluation purposes for the research community (though the data has yet to be released by *Yahoo Labs*). The team also conducted an annotation experiment: They let workers from *Amazon Mechanical Turk* annotate the user comments; Unfortunately, the results turned out to be less than ideal. Their classifier, which is built upon Vowpal Wabbit's regression model (standard settings, bitrate of 28), achieves recall of 79%, precision of 77% and an F_1-score of 0.78. They use a wide selection of different features (n-grams, linguistic features such as punctuation, the number of noise tokens, syntactic features to catch dependencies and distributional semantics). Character n-grams are the single most successful predictor for their reportedly noisy data sets.

In a research collaboration between *Wikimedia* and *Jigsaw*, a *Google*-affiliated technology incubator that tackles global challenges such as extremism and cyber-attacks, Wulczyn et al. [13] took up the idea of using character n-grams and were able to confirm and build upon the findings of Nobota et al. [9]. Their massive corpus of over 100k comments (and an additional 63M machine-labeled comments), stemming from "talk pages" of English Wikipedia and labeled by crowd-workers, is publicly available [28] and a major contribution to the research community. Each comment was labeled by at least ten *Crowdflower* annotators. The features of their model consist of bag-of-words representations of either word unigrams and bigrams or character n-grams in the range of 1 to 5. The tested models are Logistic Regression and Multi-Layer Perceptrons (MLP). Character n-grams consistently delivered the best results. Also, the team managed to boost performance by an unusual labeling-approach: Instead of using clear-cut labels determined by the judgement of the majority, the labels are empirical distributions (ED); this means that a comment that was considered an attack by 7 out of 10 annotators has the label [0.3, 0.7]. Their best evaluation score is an AUC of 96.59 using the MLP model, character n-grams, and ED labels. While the authors openly point out the need for further research, as well as current shortcomings of their system, the technology has already been launched under the name *Perspective API.*[3] Its partners include major news platforms such as *The New York Times* and *The Guardian.*

Table 3.1 offers a summary of the features used across the cited research literature.

3.3 Challenges

Challenges in offensive text classification span from obtaining a suitable corpus and the limitations of natural language processing tools to the informal nature of user-generated content.

[3]https://www.perspectiveapi.com/

Table 3.1: Overview of features mentioned in the researched literature.

obscene or blacklisted words [5, 9–12]	noise words [9]
polite speech [9, 11]	unknown words [9]
subjectivity [10]	modal verbs [9]
intensifiers [5]	one-letter words [9]
word n-grams [4, 5, 9, 12, 13]	capitalized letters [5, 9]
character n-grams [9, 13]	average word length [9]
POS-tag n-grams [12]	URLS [9, 11]
phrase patterns using wildcards [10, 11]	telephone numbers [11]
epithets [11]	short sentences [5]
word dependencies [4, 5, 9]	imperative [5, 11]
exclamation marks [5, 9, 11]	distributional semantics [9]
question marks [9]	2nd person pronouns [11]
periods [9]	laughter [11]
repeated punctuation [9]	offensive terms near 'you' [5, 11]
quotes [9, 11]	positive terms near 'you' [11]
total number of tokens [9]	lexical diversity [22]

3.3.1 Informal Text

Spelling

Informal text written by users often contains one or more of the following properties: Accidental misspellings or grammar-related errors (e.g., *your / you're; should have / should of*), misuse of punctuation, intentional misspellings used to evade blacklist filters (e.g., *noise words*, characterized by a mix of alphabetical and non-alphabetical characters (e.g., *'Sh1t', '$#*T'*) or *spaced words* (e.g., *'S H I T'*)). Since natural language processing tools are trained on formal and correct text corpora—which in turn is also the type of input these tools expect—they generally are not prepared for handling informal, noisy text.

Simple errors could be fixed by running a spell- and grammar correction as a preprocessing step [4, 5, 22, 12], or text normalization to correct lengthened words (e.g., *'niiiiice'*). On the other hand, for offensive language detection, some of these errors might prove to be valuable indicators.

Evolving Internet Slang

A blacklist will catch many offensive terms and yet it would need to be continuously updated due to the ever changing vocabulary of the internet community [9] which often gives birth to new word creations (e.g., *'covfefe'*—a cryptic but popular invention of a prolific twitter user; *'Hitlery'*—a disparaging reference to presidential candidate Hillary Clinton).

Sarcasm

Automated sarcasm-detection is an ongoing research field on its own. Even for humans, it is very hard to detect sarcasm in written text, especially if the context is not known.

3.3.2 Lexical Ambiguity

The offensiveness of a word may depend on the context (e.g., *'bitch'* is a common insult, though it is not an offensive term when it refers to a female dog) and there may be many subtle cases where natural language processing tools will fail. For instance, Spertus [11] observed that the statement *"Cool page... "*, which was supposed to be a compliment for a website, was misidentified as an imperative statement since *'cool'* can be interpreted as a verb too.

3.3.3 Data

Despite the upsurge in offensive language detection in the recent years, scarcity of data is still an issue. Likely due to the sensitive nature of the material only very few datasets relating to offensive language have been published so far; most researchers have relied on in-house annotators. Apart from the technical issues, the quality, amount, and ratio of the training data play a critical role. Also, the label of a comment depends on the personal interpretations of the annotators [14]:

> *Even following detailed discussion, human analysts would often still disagree on meaning, intent and purpose. Even where analysts discussed disagreements over how to classify specific texts (for example, whether it was 'nonderogatory' or 'casual use of slur') there remained continued disagreement. Two analysts working on the same data set were able to agree 69 per cent of the time when classifying the data. There were several reasons for this, including cultural biases.*

Wulczyn et al. [13] also point out that even though their crowd-annotators had reasonable levels agreement on the labels, their judgement might not align with that of the target community.

Chapter 4

Methodology

4.1 Concept and Architecture

The ingredients for building an aggressive language classifier that draws accurate predictions from an effective combination of features are:

- A corpus of text documents, each labeled in terms of aggressiveness (section 4.2),
- a wide range of features that can be extracted per text document (section 4.3),
- filters, wrappers and embedded methods to perform feature selection (section 4.4),
- machine learning models for training and testing (section 4.5), and
- a performance metric for model evaluation (section 4.6).

This section generally explores when and how these elements come into effect; the subsequent sections offer a more detailed description per item.

4.1.1 Feature Extraction

First, we translate each text document of a corpus into a feature vector that contains numeric values (figure 4.1(a)). We undertake the following steps to extract the features.

Text Preprocessing

For each comment of the corpus, different representations are created using natural language processing (table 4.1). Part of an initial set of attributes is extracted during the processing of the text, others are extracted from one of the text's representations.

Feature Engineering

We skip this step during the first iteration of the classifier building process. Once we have a better understanding of the underlying data (e.g., after analyzing feature distributions or instances where our classifier fails), we start to experiment with the initial set of features.

One approach in feature engineering is to combine features and thereby create additional ones. As an example, the comments in our corpora vary notably in the number of words and sentences. Dividing a feature (e.g., the count of imperative sentences) by

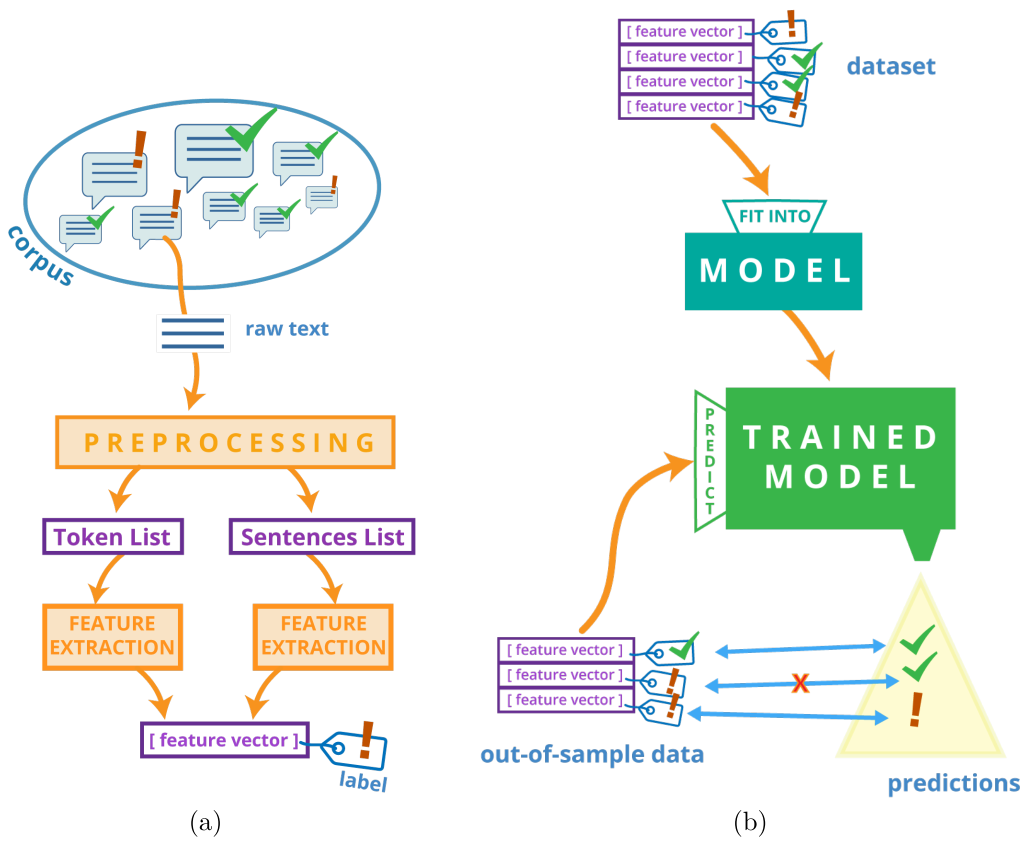


Figure 4.1: (a) Each raw comment text is preprocessed into different variations that serve as a basis for feature extraction. The result is a feature vector per text document. (b) The dataset consists of two parts: The *data*, an array of feature vectors (a 2D matrix), and the *target*, an array of the corresponding labels. Part of the dataset is used to train the model. The other part is called *out-of-sample data*. We use the trained model to predict labels for the out-of-sample data. The performance of the model is measured by comparing the predicted labels and the actual labels.

another attribute, such as the total count of sentences, creates a new, possibly better predictor. The opposite path—creating independent features from a single one—is a possibility too (e.g., separating individual periods and ellipses). Another approach is to redefine a characteristic itself (e.g., changing the definition of what counts as a *noise word*); this entails returning to the feature extraction phase.

4.1.2 Development Set and Validation Set

A methodologically crucial step to avoid bias and overfitting is the partitioning of the dataset into two parts: The *development set* (dev set) is used for feature selection, hyperparameter tuning and fitting the models; the *validation set* (val set) is exclusively used in the final step: to measure a models performance. Each dataset is shuffled prior to being split with a ratio of 3 : 1 (dev set : val set), yet maintaining the balance between the classes.

Table 4.1: An input text is transformed into different representations that serve as a basis feature extraction.

Representation Type	Output
raw text	`"I've seeeeeeeen th!ngs you people wouldn't believe. Attack $hips on fire......."`
stripped word tokens	`[['I', 've', 'seeeeeeeen', 'th!ngs', 'you', 'people', 'wouldn', 't', 'believe'], ['Attack', '$hips', 'on', 'fire']]`
normalized sentences	`[["I've seen things you people wouldn't believe."], ["Attack ships on fire..."]]`
normalized and POS-tagged sentences	`[[('I', 'PRP'), ("'ve", 'VBP'), ('seen', 'VBN'), ('things', 'NNS'), ('you', 'PRP'), ('people', 'NNS'), ('would', 'MD'), ("n't", 'RB'), ('believe', 'VB'), ('.', '.')], [('Attack', 'NN'), ('ships', 'NNS'), ('on', 'IN'), ('fire', 'NN'), ('...', ':')]]`

Data Scaling

Data scaling (also: feature scaling, data normalization) is a preprocessing step that standardizes the range of the independent features. Most machine learning models require a standardized input in order to function properly;[1] another benefit of scaling is an increase in speed, that is especially significant in Support Vector Machines.

In practice, we scale the development set to the $[0, 1]$ range (which reflects the fact that all features contain non-negative values). Most importantly, we must retain the scaling values: It is essential to apply the *same scaling transformation* to the validation set (and respectively to any other prospective out-of-sample data).

4.1.3 Generating Feature Subsets

Finally, feature selection comes into play:

1. We use common statistical tests (T-test, Wilcoxon-ranksum-test and Mutual Information) to get scores for each feature type individually.
2. For each test, we sort the full set of n features in descending order by score and thereby translate the results into a feature ranking list consisting of the features' column indices.
3. A fourth—artificial—ranking is generated by the combination of the three statistical rankings. The combined ranking is based on the sum of the statistical ranks

[1]Unstandardized features might lead to unexpected behavior, such as sensitivity of the model towards the order of the feature columns in the data matrix (an observation made with Scikit-learn's `LogisticRegression`).

per feature; the feature with the smallest rank-sum is deemed the best in the combined ranking.

4. From each ranking, we create $n - 1$ feature selections (in the following context a *feature selection* denotes a list of feature-column indices), simply by starting with the full list of feature indices—sorted by rank—and recursively slicing the current sub-list without the last element until the sub-list size is down to one.
5. We apply the same sub-list-slicing technique to the ranking derived from Recursive Feature Elimination.[2]
6. The Lasso, like the RFE, generates a list of favorable features without an internal ranking. However, it does not support a custom selection size. By systematically experimenting with different input parameters (the `alpha` value and a `threshold` to cut off less important features) we gain selections of different size.
7. Lastly, the actual subsets are created by applying each feature selection like a mask to the full dataset (in effect: the development set as well as the validation set); in other words, a subset is created by using only the feature columns of the current selection. This results in $5 \cdot (n - 1)$ feature subsets derived from the rankings, plus an individual number (dependent on the dataset) based on the Lasso selections.

4.1.4 Building and Testing Models

At this point, it is time to test each of the feature subsets using three different machine learning algorithms: Logistic Regression, linear SVM and Gaussian SVM.

In general, the process per algorithm and per subset is as follows (figure 4.1(b)): The model is trained, using the development *subset*. Next, predictions are made for the validation *subset*; what follows is a comparison of the predicted labels and the actual labels to derive the confusion matrix consisting of the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN).

In practice, the process is as simple as described for Logistic Regression (using default parameters). For the SVMs, the tuning parameter(s) per subset need to be determined beforehand: An F_1-score guided grid-search (using 5-fold cross-validation) is performed on each development subset; finally, each SVM model is trained using its individual tuning parameters per subset.

4.1.5 Model Evaluation

The F_1-score is our primary means to determine the most favorable models for separating the aggressive comments from the non-inflammatory ones. The aim is to arrive at the most predictive core subset. Therefore we incorporate additional heuristics to select the most compact subset among the top ranked: For each of the three machine learning algorithms we narrow its list down to the *top 10 models* and then pick the model with the *minimum number of features*. Out of these *three top choices*, we select the one with the highest F_1-score as the final model.

[2]The RFE does, in fact, return selections of predefined size without an explicit ranking; the ranking was created specifically, to enable a comparison.

4.2 Datasets

Two different corpora were used independently of each other in this research, the *Wiki Dataset* and the *Martin Dataset*.

4.2.1 Wikipedia Talk Corpus: Personal Attacks

The Wikipedia Talk Corpus [28] is a corpus of over 100k human-labeled comments from discussion pages of English Wikipedia. Each comment was labeled by at least 10 crowd-workers on whether it has aggressive tone.

For our research, the focus is set on clear-cut separation of labels from the human perspective. Therefore we apply a threshold of 25% from both ends of a comments aggression-mean[3]—analog to an inversion of a typical band-pass filter—to filter out ambiguous comments. From the remaining comments, the first 3 500 aggressive respectively non-aggressive samples form an inital dataset of roughly 7k samples.[4] To remove extreme outliers, the selection is further narrowed down on basis of the following criteria:

- A comment must contain at least 3 words and no more than 250 words,
- the maximum allowed number of sentences per comment is 25 and
- the maximum allowed length of a word is 100 characters.

Comments that violate any of these criteria are dropped. The final number of samples is 6 647 in total, which splits into 3 363 aggressive and 3 284 OK samples. We refer to this specific compilation as the *Wiki Dataset*. Comment samples are listed in table 4.2.

4.2.2 Martin Corpus

The Martin Corpus consists of just over 1k comments and is well balanced with 520 labeled as aggressive and 514 labeled as OK. The corpus was developed by a single volunteer during the early phase of this research project.[5] Aside from collecting and annotating the comments, the volunteer was not involved in the research. The corpus is named in his honor. Comment samples are listed in table 4.3.

Annotation Guidelines

- **OK (not aggressive):** Author uses neutral, factual, polite language. Opinions or statements are shared to inform or encourage a healthy, fair discussion.
- **Aggressive:** Content that is derogatory, hateful, threatening, pornographic, incites violence, makes use of stereotypes to express an ideology of hate.
- **Source:** Comment section of reputable news media (e.g., *The New York Times*) and extremists (e.g., *Infowars, Breitbart, Fox News*).
- **Language:** English (United States)

[3] An aggression value is 0 if the comment is considered neutral or friendly by a worker, respectively 1 if considered aggressive. The aggression-mean per comment is based on all available aggression values for the given comment.

[4] The groups are deliberately balanced and do not reflect the actual distribution of the corpus.

[5] The comments were collected between 2016-12-15 and 2017-04-13 from comment sections of US online News media like *The New York Times* and *Breitbart News*. The source articles revolve around the US presidential election of 2016 and its aftermath.

Label	**rev_id**	**Comment**
OK	103624	I removed the following:\n\nAll names of early Polish rulers are ficticious and therefore this index naming Oda von Haldensleben and her husband Dagome records for the first time rulers of the Polanen tribe. Therefore it is indicated as being the first document of the later developing land named Poland.\n\nThis is quite a comment. All names are fictitious? It deserves at least some backing.
OK	297866	\n\n\n\nUmmm. The article uses imperial measurements, not SI ones. Some translation is apparently in order. -) Anders Törlind
OK	93890	This page will need disambiguation.
OK	360594	\n\n::I disagree. I would leave it out of the introductory part, since its population (which changes every census) is not nearly as fundamental as the fact that it is in a particular state, its county seat is whatever it is, andstuff like that. Also, if Wikipedia survives till 2010, someone will have to be careful to change it in both places! BRG\n\n
aggressive	24464221	stop editing this, you dumbass. why do you have to be such a bitch? the ghosts of bill maas' past will haunt you forever!!! MWAHAHHAHAA
aggressive	24777589	== "Goofs" Goof ==\n\nSpeaking of goofs, who the fuck had the, uh, initiative, to do this? What kind of sick person watches the Simpsons, and checks how many buttons they've got on their coats, or how often they shave, etc.? And especially, on that last comment... What is your problem?! You were measuring the breast size of a cartoon character?! Why can't you just buy a magazine, or something (or possibly consider how close the shot was to her), you poor, lonely bastard?!
aggressive	24509777	\n\nUp yours, you authoritarian little Hitlers.
aggressive	10144987	\n\n:: First of all, who the fuck died and made you the god on what is and what is not plagiarism? It seems like a pretty clear cut case to me. If text from the article matches text found in another source word for word, then it seems fairly clear cut that it is plagiarized. I would also point out, again, for like the umpteenth fucking time, that this user has been using plagiarized material repeatedly and only takes it out after being beaten over the head with it. Secondly, don't you fucking dare try and threaten me with what might and might not be said in arbitration. Finally, in case you were not aware or just did not care, an AOL IP cannot be blocked, so that is not any threat for the anon.

Table 4.2: Wikipedia Talk Corpus example comments; the feature vectors derived from these samples are located in appendix A.4.

Label	Filename	Comment
OK	0___113.txt	Me too. I really feel he will get a lot done. He is a man of urgency and justice. I believe he wants to see us get back to being great again. He will have a hard road ahead of him though, but he can handle it.
OK	0___116.txt	OK. We can disagree. No problem.
OK	0___140.txt	I am deeply troubled by the Democrats laissez-faire attitude toward both Trump's election and the confirmation hearings. There was no push-back on a president-elect who lost by about three million votes and yet claimed a stunning victory. Today, there is no push-back on a man whose values are a throw-back to the pre-Civil Rights era South. Democrats behave as though there is no way to stop the GOP propaganda machine. Really? Clinton won by three million.
OK	0___406.txt	I can do that, I can get more involved locally, but I can't clean our air or water, I can't prevent climate change weather related catastrophic events, I can't prevent oil spills from pipelines or in our parks and oceans. This is not funny.
aggressive	1___067.txt	My plumber today told me that Donald Trump is full of sh!t, just like the sewer line he was cleaning out. I know he is right.
aggressive	1___191.txt	The failure of Obama will be amplified as Donald Trump's policies take root and actually improve the economy. The problem is that blacks got a taste of power and now think they should run everything. If blacks weren't so stupid, they would realize that is the surest path to failure as is evidenced everywhere in the world where blacks have power and control, whether it's a city, state or country.
aggressive	1___644.txt	lol,, i can see obama stealing everything ,, theyre classless tacky pigs ,, and i'd really like to see the cops just beat the hell out of these scumbag protestors
aggressive	1___694.txt	WE MEN GAVE WOMEN THE PRIVILAGE TO VOTE. IT IS OBVIOUS THEY ARE NOT READY FOR IT. NOW WE MEN NEED TO TAKE THAT PRIVILAGE BACK.

Table 4.3: Martin Dataset example comments; the feature vectors derived from these samples are located in appendix A.4.

4.3 Features

A portion of the engineered features are adopted from prior research, others are variations thereof or original ideas. Opposed to the majority of research projects, my approach does not include n-grams of any kind. Instead, the idea here is to look for more subtle predictors of aggressive language that ideally prove to be independent of a given corpus. Nonetheless, vocabulary still plays a significant role in the form of specific dictionaries and pre-defined word lists.

One of the strategies is to embrace the challenges of informal text (outlined in section 3.3.1) by detecting occurences like word spacing or leetspeak, transforming them into a more useful, proper form where applicable and incorporating their properties into the feature vector. Also, the amount of change the raw text has undergone during this transformation, expressed by the total edit distance, is an observed characteristic.

In a novel approach, even the dynamic changes in the language of internet communities can be leveraged. Internet users are as creative in making up new terms as they are diligent in defining these terms in the *Urban Dictionary.*[6] With regard to the features, this allows a differentiation between internet slang words and actual unknown words.

Curiosity rather than a challenge or expectation has fueled some grammar-related features: Are aggressive users more likely to write in the present, or in the past verb form? Do they compare more or use superlatives more frequently? Who uses interrogative pronouns more often?

The effect of punctuation-style has been researched before, though not extensively. This work introduces new groupings for punctuation—highlighters and connectors—, investigates whether ellipses or em dashes are useful predictors and experiments with variations of classic punctuation features.

To capture the richness of a comments vocabulary, we measure the lexical diversity[7] (as suggested in a blog-article on text moderation [22]). Based on that concept, the range of individual POS-tags has been formulated as a new measure of diversity.

Finally, we extract an assortment of low-level statistical characteristics. Among them are a ratio of whitespaces or non-alphanumerical characters, as well as simple word- and sentence-length statistics, most notably of which is a count of words that exceed a given length-threshold.

In total, there are 68 features—all of them purely non-negative, either discrete (e.g., counts) or continuous (e.g., ratios); none of them are of categorical nature. We group the features into six categories: Basic statistical, lexical, grammar-related, punctuation, noise and style, and diversity.

In the following listing, a number in a parenthesis refers to the column index of a feature. For instance, (1) is the index of the feature that represents the count of long words; the feature with index 62 is a variation of (1) and together they are summarized under the title *Long Words (1, 62)*.

[6]The Urban Dictionary (http://www.urbandictionary.com/) is an open online dictionary of slang words and phrases where users may submit their own definitions.

[7]Lexical diversity measures the spectrum of indivual words used in a text.

4.3.1 Basic Statistical Features

All basic statistical features are extracted before text normalization. A few are extracted directly from the raw text (features 6, 8 and 35), all other basic features are extracted after simple preprocessing, such as tokenization and stripping of punctuation at word boundaries.

Words (10, 12, 13, 14, 50)

For the total word count (10), we take into account only tokens that contain alphanumeric characters. Words connected by a hyphen (e.g., *'muddle-headed'*) are considered as one single word. Contractions, such as *"you're", "it's"*, and words separated by a slash or em dash are regarded as separate words. The total word count is also used in other features—as a divisor—to establish a reference between the respective feature and the length of the comment in words. The next set of features excludes solely numeric tokens: Punctuation and other non-alphabetical characters, that appear at a words boundaries, are removed, before finding the length of the longest word (12), the average word length (13) and the median word length (14). The shortest possible word length is 1, and even though one character words are not an uncommon phenomenon in the English language, an increased frequency may be regarded as an anomaly. The total number of one character words is divided by the number of sentences (50).

Long Words (1, 62)

Words that surpass a specified length-threshold are considered long words. We use a length-threshold of 7 characters, which was initially an intuitive, experimental choice that remains unchanged due to its obvious effectiveness. The first representation of this feature is the count of all long words (1), the second is the count of long words divided by the total word count (62).

Numbers (64)

This feature represents the amount of numeric values (plain numbers or monetary values, e.g., '1982', '$1.000') that appear in a comment.

Paragraphs (8, 54)

The paragraph count (8) simply represents the number of occurrences of the newline character.[8] A second feature is the ratio of the paragraph count and the number of sentences (54).

Sentences (9, 18, 19, 20, 21)

A sentence is a sequence of words that ends in a period, question mark or exclamation mark followed by a whitespace or end of line. We expand this common conception by defining also a linebreak as a sentence terminator, to account for not existing or

[8]In an initial preprocessing step divergent forms of newline characters ('`\r`', '`\r\n`') were standardized into '`\n`'.

extremely minimal punctuation style. The total number of sentences (9) is a feature on its own, as well as a divisor in other features to establish a reference between the respective feature and the comment length in sentences. Before the next set of features is extracted, each sentence is tokenized into a list of words, punctuation stripped. The number of words in a sentence list defines the length of that sentence. Extracted features are length of the longest sentence (18), average sentence length (19), median sentence length (20) and shortest sentence length (21).

Whitespaces (6)

Here we set the number of whitespaces (e.g., blanks, newline characters, tabs) in relation to the total length of the comment in characters:

$$\text{Whitespace-Ratio} = \frac{\text{count(whitespaces)}}{\text{length of comment in characters}}. \tag{4.1}$$

Non-Alphanumeric-Ratio (35)

To calculate this feature, the total count of non-alphanumerical characters (whitespaces excluded) must be divided by the length of the comment in characters (whitespaces excluded).

4.3.2 Lexical Features

All lexical features are extracted from normalized text, split into word tokens and ignore punctuation.

Regular Dictionary (43)

Each word in a comment that also exist in a regular US-English dictionary contributes to the regular dictionary word count.

Blacklist (36, 40, 56, 60)

Each occurence of a blacklisted word increments the total blacklist count (40). The total blacklist count is used after being divided by the number of sentences (56) and also by the length of the comment in words (60). The count of blacklisted words written in all capital letters (36) is a more specific blacklist feature.

Politeness (44, 67)

Each word in the comment that also appears in a list of polite words—like *please, thank, sorry*—contributes to the total count of polite words feature (44). An imperative is considered polite, if it contains the words *please* or *thank*, or if it uses the Empathic Do (e.g., *'Do sit down.'*). The number of polite imperatives is divided by the total number of imperatives (67).

Subjectivity (22, 23, 24)

For the overall Subjectivity-Ratio (22) the total number of all words of the comment that also appear in the subjectivity lexicon—regardless of polarity (*positive, negative, both, neutral*) and strength (*weak, strong*)—are divided by the total number of words (excluding numbers) of the comment.

Weakly subjective words are assigned the value `0.5`, strongly subjective words are assigned the value `1.0`. The Positive-Subjectivity-Ratio (23) is the summation of strength values of positive subjective words divided by the total number of subjective words. Similarly, the Negative-Subjectivity-Ratio (24) is the summation of values of subjective words with negative polarity divided by the total number of subjective words found in the comment.

Urban Dictionary (39, 55, 59)

Words that do not appear in a regular English dictionary, such as proper names or popular internet slang words (e.g., *'dafuq'*), are often to be found in the Urban Dictionary. The number of words that are only to be found in the Urban dictionary is used in total (39), divided by the number of sentences (55) and divided by the total word count (59).

Unkown Words (41, 57, 58)

Words that appear neither in a regular English dictionary nor in the Urban Dictionary are represented in the unknown words count (41). This number is also used after being divided by the total number of sentences (57) and divided by the total word count (58).

4.3.3 Grammar Features

All grammar features are extracted from normalized, POS-tagged text.

Imperative (45, 49, 65, 66, 67)

First, all sentences that are found to be in imperative form contribute to the total imperative count (45), then this number is also divided by the number of sentences (49). The number of indirect (65) or polite (67) imperative sentences is divided by the total imperative count. An imperative is considered indirect if it ends with a question mark (e.g., *"Everybody be quiet, will you?", "Would you help me, please?"*). An imperative ending in a period or a single question mark has strength `0`, an exclamation mark or two question marks or a combination of one exclamation mark and a question mark has strength `1`, any other sequence of exclamation marks or question marks is considered quite strong and has strength `3` respectively `7` if the punctuation sequence is longer than 3 characters. Feature #66 represents the combined strength of all imperative sentences divided by the number of imperative sentences.

Present, Past and Modal Verbs (31, 32, 33, 48)

The Modal-Verbs-Ratio (31), the Present-Verbs-Ratio (32) and the Past-Verbs-Ratio (33) differ only in the denominator. Each of the specific verb counts is divided by the total number of verbs. The total number of modal verbs is also divided by the number of sentences (48).

Comparatives and Superlatives (27, 28, 29, 30)

These features are very similar. The first two are about adjectives only, the last two concern adverbs and they are both grouped into comparative and superlative ratios. Comparative adjectives (27), respectively superlative adjectives (28), are divided by the total adjective count. Comparative adverbs (29), respectively superlative adverbs (30), are divided by the total adverb count.

WH-Pronouns (2, 52)

Why, Where, When, How, What, Who, Whoever, Which, Whatever are so-called WH-Pronouns. The total count of WH-Pronouns (2) is one feature. It is also divided by the number of sentences (52).

4.3.4 Punctuation Features

Most punctuation features are extracted from the raw, unprocessed comment. The only exceptions are ellipses and em dashes, which are normalized beforehand.

Punctuation Ratios (0, 4, 5)

These three features are almost identical, the only difference being the denominator. It is a ratio between a single punctuation type (exclamation mark / period / question mark) and the sum over the counts of all exclamation mark, period, question mark: Exclamation-Mark-Ratio (0), Period-Ratio[9] (4) and Question-Mark-Ratio (5).

Ellipses (3, 47)

An ellipsis (...) is used to indicate omission or, more informally, a pause, hesitation, etc. First, all ellipses are count and used in total (3), then the ellipsis count is divided by the number of sentences (47).

Highlighters (7)

Quotation marks, parenthesis and *asterisks* make up the ‘highlighters’ category. They visually separate content from the rest of the text, which may be interpreted as a form of ‘highlighting’. All occurrences of these highlighters (regardless of whether they are being used syntactically correct) contribute to the total count of highlighters.

Connectors (51)

Commas, semicolons, colons, and *dashes* fall into the ‘connectors’ category. They can be seen as connecting elements between otherwise separate parts of a single sentence. The count of all these connectors is divided by the number of sentences.

[9]In this case there is no distinction between a single period and ellipses: A sequence of three periods contributes a count of 3 to the total period count.

Em Dashes (25)

The number of all occurrences of the em dash[10] make up this feature. Before feature extraction, em dashes are being normalized, so they only appear as double hyphens—instead of triple hyphens or even more.

Repeated Punctuation (34, 53)

Any sequence of two or more exclamation marks, question marks or commas counts as repeated punctuation (e.g., *'!!'*, *'?!?'*, *',,,,,'*). Feature #53 is the total count of repeated punctuations (regardless of their individual length) divided by the number of sentences. Feature #34 is the average length of all repeated punctuations. Sequences of periods are a valid concept known as ellipsis and therefore excluded from this feature.

Regular Sentence Endings (46)

The common way to end a sentence is by use of a period, question mark or an exclamation mark. In this particular feature, there is no distinction between multiple singular periods and a sequence of periods, meaning that ellipsis are not recognized as such:

$$\text{Regular Sentence Endings} = \frac{\text{count}([.?!])}{\text{count}(\text{sentences})}. \tag{4.2}$$

4.3.5 Noise and Style Features

The majority of these features are extracted during the text normalization process.

Noise Words (38)

Often *asterisks* or similar special characters are used to replace vowels in curse words (e.g., *'f*ck'*, *'qu**r'*). Consequently, words that contain at least one non-alphabetical character (slashes and dashes excluded), are being registered as noise words and contribute towards the count of noise words (38).

Corrected Words (16, 42, 63)

Lengthened words and Leetspeak[11] will be corrected during the normalization process. The count of each corrected word is used in total (42) and divided by the total word count (63). The sum of all edit distances[12] denotes the amount of modification per comment (16).

[10] A long dash which—instead of an *actual* long dash—often appears as a pair (or triplet) of hyphens.

[11] Leetspeak is a form of writing where single letters of a word are replaced by similarly looking numerals or special characters. It is often used as a means to evade blacklist filters (e.g., *'d1ckh34d'*, meaning *'dickhead'*).

[12] The Levenshtein distance, commonly referred to as edit distance, is used to measure the difference between two strings. E.g., the edit distance from leetspeak word *'$w33t'* to its corrected version *'sweet'* amounts to 3 since three letters were exchanged.

Spaced Words (15)

Word spacing, either by separating the single letters of a word by a blank or a special character (e.g., *'H a w k e y e'*, or *'M*A*S*H'*) is an aesthetic effect of putting emphasis on a word. Spaced words are usually not automatically recognized during tokenization and therefore falsely regarded as single one character tokens. Therefore, we develop a method to detect and collapse spaced words, which also keeps track of the number of spaced words it encountered. The total count of all spaced word makes up this feature.

Word Lengthening (17)

Word lengthening (e.g., *'cuuute'*) is another form of emphasis or accentuation, mimicking prosodic characteristics which would otherwise be lost in written text. It is often used for words that have a strong indication of sentiment [3]. Lengthened words are being normalized during preprocessing. The extracted feature is the count of lengthened words.

All Caps (36, 61)

Writing words or even the whole comment in ALL CAPS is a stylistic choice that is often perceived as yelling. Feature #36 is the count of all words that are blacklisted and written in capital letters, feature #61 is the ratio of the count of all words written in capital letters and the total word count.

Mixed Case (37)

Words that display an informal mix of upper case and lower case letters (e.g., *'YiKeS'* or *'yIKES'*; but not *'Yikes'*) increment the count of mixed case words.

4.3.6 Diversity Features

The Type-to-Token-Ratio is extracted from the unnormalized text, while the POS-Type-to-Word-Ratio is based on the normalized, POS-tagged version of the text.

Type-to-Token-Ratio (TTR) (11)

The TTR is a simple measure of *lexical diversity.* Its value is low for comments where tokens are repeated frequently, and only reaches the maximum value, 1, if all tokens are unique. Before the calculation, punctuation is removed, all words are set to lowercase and stemmed (e.g., *'Stop!'* and *'stopped'* are considered the same, *'stop'*):

$$\mathrm{TTR} = \frac{\mathrm{count(unique\ tokens)}}{\mathrm{count(tokens)}}. \tag{4.3}$$

POS-Type-to-Word-Ratio (26)

This feature, inspired by the TTR, aims to capture the *grammatical diversity* by juxtaposing the number of unique POS-tags with the number of total words. Note, however, that POS-tags include punctuation too, while words strictly refer to words and numbers:

$$\text{POS-Type-to-Word-Ratio} = \frac{\text{count(unique POS-tags)}}{\text{count(words)}}. \tag{4.4}$$

4.4 Feature Selection

The goal of feature selection is to remove meaningless, noisy or redundant features in order to gain a model that generalizes well, needs less time and resources, and performs well on out-of-sample data. Another benefit is the knowledge and deeper understanding of the data that might be derived from the results [6]. Feature selection techniques are commonly split into three different categories: Filters, Wrappers and Embedded methods. The following approaches to finding the best feature subsets have been undertaken.

4.4.1 Univariate Feature Selection

Univariate feature selection is a fast method, independent of the learning algorithm, to gain insight about the features' *individual* statistical significance (figure 4.2). It is a *filter* method, that is often used as a preprocessing step to reduce dimensionality and handle overfitting. Filter methods assign a score to each feature and use the the derived ranking to keep or remove features from the dataset. With this approach, feature interactions are not taken into account, which may lead to a selection of redundant features. The following statistical tests were used to rank the features by relevance according to the obtained test scores.

T-Test

The T-test compares the means of feature values (e.g., the mean of the number of blacklisted words) per class (*OK* or *aggressive*). Its score—the ratio of the difference between the two classes and the difference within the two classes—indicates how different the classes are. A large score signals that the classes are different, which means it is likely a valuable feature to separate the classes by.

Wilcoxon-Ranksum-Test

Unlike the T-test, the Wilcoxon-ranksum-test does not assume a normal distribution of the data. It treats the feature values as ordinal data: The feature values from both classes together are assigned magnitude based ranks, and afterward reseparated into their corresponding classes. The test score is based on the sum of the ranks per class. A large score indicates that the ranks of the classes are separate enough—if one class contributes significantly more to the higher ranked values.

Mutual Information

The concept of Mutual Information is the reduction of uncertainty by measuring how much is known about a random variable when another variable is given. The Mutual Information score is computed by a combination of probabilities (e.g., the probability

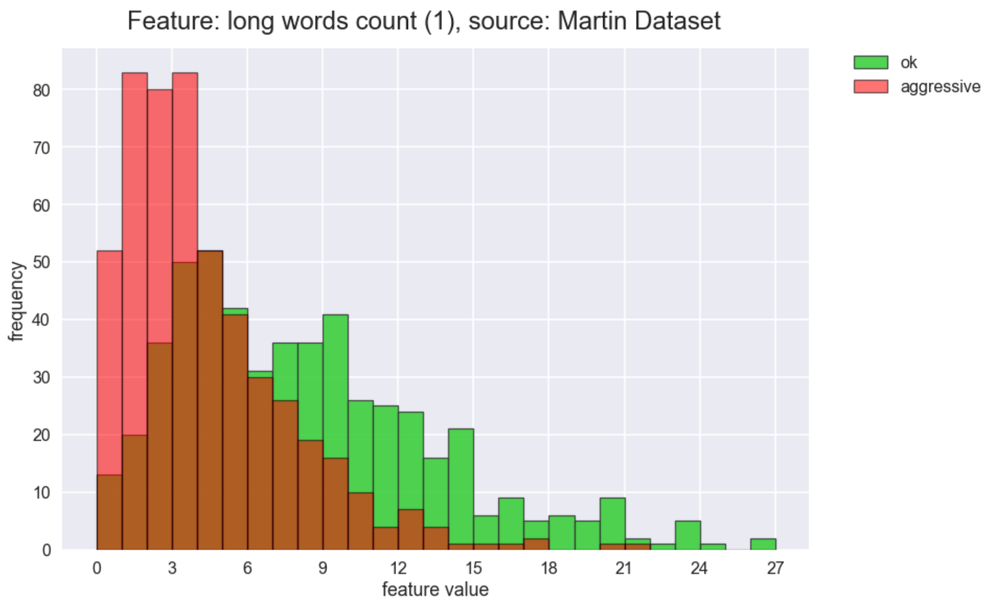


Figure 4.2: The plot shows the per-class distribution of a single feature. The red and the green histogram denote the aggressive and non-aggressive frequencies respectively; the brown area is where they overlap.

of there being X blacklisted words in a comment of class Y; the probability of there being X blacklisted words in any class; the probability of a comment being of class Y). Mutual Information is zero if feature values and classes do not provide information about one another, in other words, if they are independent.

4.4.2 Wrapper and Embedded Methods

Both, wrapper and embedded methods involve a learning algorithm and evaluate the usefulness of feature subsets—hence feature interactions are taken into account. Wrapper methods search the space of all feature subsets and use the learning algorithm only to score these subsets [6]. Embedded methods are less computationally expensive and less prone to overfitting than wrappers. The search for optimal feature subsets is guided by the learning process; feature selection and model tuning are carried out at the same time. The following methods were employed to find favorable feature subsets:

Recursive Feature Elimination (RFE)

Recursive Feature Elimination, a wrapper technique, is a greedy algorithm that starts with the full set of features. The least promising features—determined by current subset results of a classifier of choice *(here: Logistic Regression)*—are dropped step by step, until the remaining subset reaches a predefined size. Making use of the ability to predefine the final selection size, we also generate a feature ranking based on the RFE results.

Lasso

The Lasso is an embedded method that fits the model to the data by imposing a penalty to the coefficients. The penalty will shrink the least useful features to zero and therefore takes them out of the equation. The remaining, nonzero coefficients stem from the features that were deemed useful, they are the selected ones. The size of the Lasso selections cannot be predefined to an exact number, instead it is influenced via the `alpha` parameter, which regulates the strength of the penalty: The higher the `alpha`, the fewer features will be selected.

4.5 Models

We use three different learning algorithms, Logistic Regression and Support Vector Machines (linear and Gaussian), to train and test the datasets.

4.5.1 Logistic Regression

Logistic Regression is a common machine learning approach for binary classification problems. It is a generalized linear model that squashes the result into a range between zero and one by use of the standard logistic function, so it can be interpreted as a probability. The coefficients that serve as weights for the features are values that fit best to the training data and are determined by an optimization procedure, such as the maximum likelihood estimation.

The probability of an out-of-sample observation $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ being 1 is predicted by the exponentiation of the summation of the dot products of the feature vector $\boldsymbol{x}$ and the weight vector $\boldsymbol{\beta}$ (containing n coefficients) and the intercept $\boldsymbol{\beta_0}$ (a value that is computed alongside with the coefficients when the model is fitted) and divided by the same term plus 1 (to ensure the value is in the zero to one range):

$$P(y = 1|\boldsymbol{x}) = \frac{e^{\beta_0 + \beta^T \boldsymbol{x}}}{1 + e^{\beta_0 + \beta^T \boldsymbol{x}}} = \frac{e^{\beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n}}{1 + e^{\beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n}}. \tag{4.5}$$

Logistic regression can be prone to overfitting, especially when there are a lot of features. By introducing a regularization term—either Lasso (L1 regularization) or Ridge Regression (L2 regularization)—to penalize the coefficients overfitting can be avoided. While both regularization methods shrink large coefficients, the L1 penalized logistic regression returns a sparse model; the L2 penalty, on the other hand, does not set coefficients to zero, it only shrinks them in size.

In order to get an accurate evaluation for each selected feature subset—explicitly without potential further subset reduction—we choose the L2 constraint over the L1.

4.5.2 Support Vector Machines

Another widely used, robust approach for classification tasks are Support Vector Machines (SVM). In comparison to Logistic Regression, which is a probabilistic classifier, the idea behind Support Vector Machines is inspired by geometry (figure 4.3 (a)). The SVM constructs a hyperplane in (often high dimensional) space to separate the data points with the maximum possible margin. It focusses on the points that are most difficult to differentiate and chooses a hyperplane that maximizes the distance to the nearest data points on both sides.

Kernels

Two common kernels were tested: the *linear* and the *Gaussian* (also: radial basis function (RBF)) kernel. The Gaussian kernel is the recommended first choice [7]. It is best suited for data that is initially not linearly separable; here linear separability is achieved by transforming the data points into higher dimensional space which is effectively done by the kernel function (figure 4.4).

SVM Hyperparameters

To account for outliers, it is essential to regulate the complexity of the model (figure 4.3(b)). After choosing a kernel, this is the most crucial step.

The Gaussian SVM requires two hyperparameters[13] that control the trade-off between the complexity of the model and the correctness of the training samples' classification: The penalty parameter `C` of the error term—in short, the `C` parameter—and the `gamma`, which are sometimes less formally called *slack parameters.* The `C` parameter influences the simplicity of the decision surface: A low `C` results in a larger margin, leading to misclassified training samples (which is useful if these samples are outliers); a high `C`, on the other hand, means more complexity and the danger of overfitting. The `gamma` parameter defines the spread of influence of a single training sample: The higher the `gamma`, the less far its influence reaches [24]. For the linear kernel, there is only the `C` to be tuned; it is also computationally faster than the RBF.

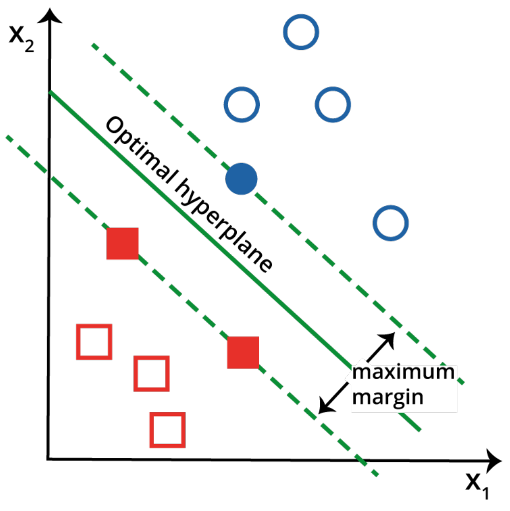
Hyperparameter tuning is best approached systematically with a *grid-search* and using *cross-validation.* The former is an exhaustive search for a good combination of `C` and `gamma` (or a search for the best `C` by itself, given a linear kernel); the latter is performed for each grid-search combination to measure its stability. The search was performed on a logarithmic grid ranging in `[0.01, 1.0, 0.1, 1.0, 10.0, 100.0, 1000.0]` for both values, `C` and `gamma`.

k-fold cross-validation divides the training set into k subsets (folds) of equal size. Each fold is once used as the validation set while the remaining folds were used beforehand to train the classifier.

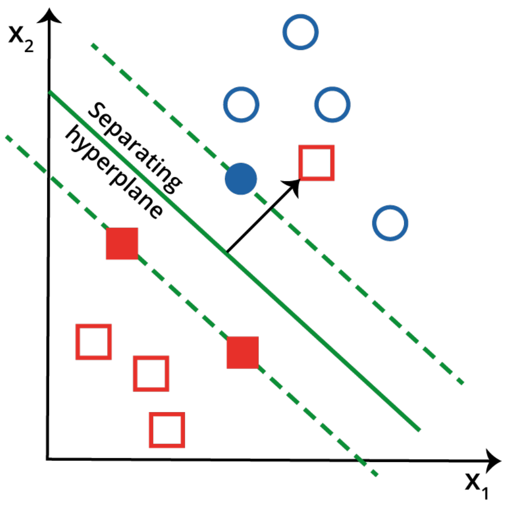
In practice, we split the shuffled training set into 5 folds, noting that the classes' balance closely resembles the balance of the whole dataset. Hyperparameter tuning per kernel is executed for each feature selection subset separately.

For the final evaluation of the feature selections, the models are trained on the development set and validated against the validation set, using the selected feature columns, as well as the specific parameters per kernel.

[13]Hyperparameters are parameters that need to be predefined befor the model training process. These parameters cannot be learned directly from the data.



(a)



(b)

Figure 4.3: (a) The geometric idea behind the SVM: Hyperplane separating the two classes, visualized as red squares and blue circles (image source: [23]). (b) Avoid overfitting: A model that admits training errors might generalize better.

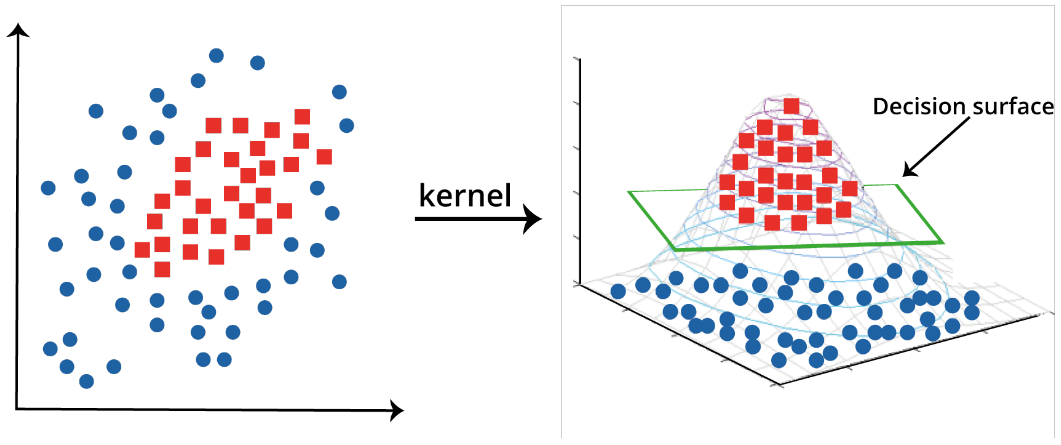


Figure 4.4: Visualization of the kernel-trick (image source: [21]).

4.6 Performance Metrics

We use the F_1-score to measure and compare each models performance. The F_1-score is the harmonic mean of precision and recall. Precision expresses what proportion of the comments that were predicted to be aggressive are in fact aggressive:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}. \tag{4.6}$$

Precision is high if most of the comments that were predicted to be aggressive are actually aggressive. It does not take into account the number of aggressive comments that were predicted to be OK (False Negatives). Recall expresses how many of the aggressive comments were correctly identified:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}. \tag{4.7}$$

Recall is high if most of the comments that are in fact aggressive were predicted as such. Recall by itself is however blind to the number of *False Positives*. Aiming to maximize both, precision and recall, the F_1-score comes into play:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{4.8}$$

Chapter 5

Implementation

5.1 Resources

5.1.1 Libraries

The system for automated classification of aggressive user comments is implemented in *Python* version 2.7.5. and utilizes the following libraries:

- *NLTK* version 3.2.2,
- *pandas* version 0.19.2
- *NumPy* version 1.13.0rc2,
- *SciPy* version 0.19.0,
- *scikit-learn* version 0.18.1,
- *Matplotlib* version 2.0.2,
- *editdistance* version 0.3.1,
- *PyEnchant* version 1.6.8,
- *urbandictionary* version 1.1 and
- *Stanford POS Tagger* version 3.7.0.

5.1.2 Lexical Resources

Blacklist

An inofficial compilation of commonly blacklisted words, the *"Full List of Bad Words and Top Swear Words Banned by Google"*[1], serves as a dictionary for all blacklist related features. The blacklist dictionary contains 550 single terms that have no additional information attached.

Polite Terms

The count of polite words feature draws from a small dictionary of 30 polite terms scraped from blog posts about polite communication with business clients.

[1]https://www.freewebheaders.com/full-list-of-bad-words-banned-by-google/

Subjectivity

The *MPQA Subjectivity Lexicon*[2] created by Wiebe et al. in 2005 contains information like polarity, strength, POS-tag and stemming-status per word. It is important to note that within this lexicon a word can appear multiple times with differing POS-tags, and linked to the tag a different strength and even polarity. The lexicon lists well over 8 000 entries, all in lowercase letters.

Regular Dictionary

The US-English dictionary that is part of *PyEnchant*, a spell-checking library for Python, is used as a means to detect whether a word is part of the regular US-English vocabulary. The spell-checker is case sensitive.

Urban Dictionary

The *Urban Dictionary API*[3] enables developers to check if a given term is part of the Urban Dictionary. The API call expects an UTF-8 encoded string; also an internet connection is required.

5.1.3 Corpora

Martin Corpus

Analogous to the *Movie Reviews* corpus that is part of the *Natural Language Toolkit* (NLTK), the Martin corpus consists of text files, where each file represents a single comment. The files are separated by label into corresponding folders. The raw text samples are accessible via the `CategorizedPlaintextCorpusReader` from the `nltk.corpus` package.

Wikipedia Talk Corpus

The Wikipedia Talk Corpus[4] is accessible in the form of a `DataFrame` via the *Python Data Analysis Library (pandas)*. Each comment has been rated by multiple crowd-workers. The label of a comment can be derived by thresholding the mean of the ratings.

5.2 Natural Language Processing

This section describes how the text is processed and transformed into different representations for feature extraction, using a custom word tokenizer, custom text normalization, POS-tagging, and chunking.

[2]http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/
[3]https://pypi.python.org/pypi/urbandictionary/
[4]https://figshare.com/articles/Wikipedia_Talk_Labels_Aggression/4267550

5.2.1 Word Segmentation

In order to extract features that relate to plain word statistics, such as the total word count, the length of the longest word, or the count of noise words, the raw text must be converted into a list of words without punctuation. The *ideal* output list for the (deliberately noisy) text *"What's... your f@vour!te colour?"* is as follows:

```
['What', 's', 'your', 'f@vour!te', 'colour']
```

A most basic, first approach to word segmentation is to split the raw text by whitespaces, though this would entail that the contraction *"What's"* (short for *"What is"*) is regarded as a single word. More sophisticated tokenizers, that are part of the standard repertoire of NLP tools, handle contractions correctly, though they are not suited well for very informal, user-generated texts; in general, tokenizers tend to split *noise words* into multiple tokens:

```
>>> nltk.word_tokenize("What's... your f@vour!te colour?")
['What', "'s", '...', 'your', 'f', '@', 'vour', '!', 'te', 'colour', '?']
```

For our particular use case, these standard solutions do not deliver satisfactory results. However, a more applicable segmentation of words, where removal of punctuation is already part of the solution, can be implemented in a few straightforward steps:

1. Any occurrence of an em dash is replaced by a blank.
2. Hyphens are removed (e.g., *'e-mail'* becomes *'email'*) in order to keep joined words as a single entity and prevent the term from being registered as a noise word.
3. Commas and slashes are replaced by blanks.
4. Apostrophes are replaced by a blank if the whole term follows the pattern of a commonly used contraction; a valid contraction[5] refers to a sequence of alphabetic characters, followed by an apostrophe, followed by either `'ll'`, `'d'`, `'ve'`, `'re'`, or `'t'`, trailed by an end of line, punctuation mark or whitespace. It is denoted by the following regular expression (compiled with the case insensitive option):

   ```
   r'([a-z]{1,}\'(ll|d|ve|s|re|t)($|[\.!? ]))'
   ```

5. Now, the processed string is split into tokens by whitespaces, with some punctuation still attached.
6. The remaining punctuation is discarded—without destruction of noise words—by stripping the leading (`lstrip()`) and trailing (`rstrip()`) punctuation from each item in the list. The example below shows the temporary word tokens of the raw string `"It's blue!... No; --YELLOOO@AAHH!!!"` and its final word tokens, after the removal of the outer punctuation:

   ```
   >>> temp = ['It', 's', 'blue!...', 'No;', 'YELLOOO@AAHH!!!']
   >>> punct = '\'"`[]{}()<>?.,;!'
   >>> [t.lstrip(punct).rstrip(punct) for t in temp]
   ['It', 's', 'blue', 'No', 'YELLOOO@AAHH']
   ```

[5]Note: This is not an exhaustive list of valid contractions. In fact, it is missing the *'m'* (e.g., *"I'm"*). It also does not cover common informal contractions, such as *"y'all"*.

5.2.2 Text Normalization

The correctness of grammatical features, such as the Past-Verbs-Ratio or the count of imperatives, depends strongly on the accuracy of the output of the employed part-of-speech tagger. Besides the choice of the tagger itself, it is the form of the input that greatly influences the quality of the tagger's output. Text normalization, in this context, means to 'correct' informal words and to improve the formal structure of the text. The main intention is to optimize the output of the tagger. As a byproduct, we can extract features that capture informal use of language, such as the counts of spaced words, lengthened words or 'fixed' (modified) words. Moreover, all lexical features are extracted from the normalized text for improved precision (e.g., the word *'$hiiiiiit'* is not part of a blacklist, but the normalized version—*'shit'*—is.)

The normalization process performs the following operations on the raw text:

1. Any spaced punctuation sequence is deflated, by removal of the blanks inbetween the punctuation marks (e.g., *'! ? ! !'* becomes *'!?!!'*)
2. Any sequence of two or more periods is converted into a standard ellipsis consisting of three periods with a trailing blank.
3. Em dashes are standardized into two dashes with a leading and a trailing blank.
4. Any sequence of more than one comma is replaced by a single comma.
5. Any non-textual elements (e.g., *'===>'*)—excluding punctuation—are removed.
6. Inflated words are deflated into a single word (e.g., *'D o n a l d M.i.c.k.e.y G*O*O*F*Y'* becomes *'Donald Mickey GOOFY'*)
7. At this point, the preprocessed text is temporarily split into tokens (using blanks). An inspection and possible modification per individual word follows:
   (a) If the word contains an at sign (@) and is not an e-mail address, the '@' is replaced by an 'a'.
   (b) If the word contains an exclamation mark that is not discernible as a punctuation mark, the '!' will be replaced by an 'i'.
   (c) If the word contains a dollar symbol ($), it is identified as either a monetary value with a missing blank between the '$' and the value, or the '$' is assumed to be an 's' in disguise. Thus, *'$3.5'* becomes *'$ 3.5'*, *'$wi$$'* becomes *'swiss'*.
   (d) If a word appears to be a mix of alphabetic and numeric characters, it is assumed that the numeric characters stand for visually similar alphabetic letters. Each numeral is replaced by a letter according to the *1337 alphabet*, e.g., *'9234t'* becomes *'great'*:

```
basic_leet_nums = {'0': 'o', '1': 'i', '2': 'r', '3': 'e', '4': 'a',
     '5': 's', '6': 'b', '7': 't', '8': 'B', '9': 'g'}
```

   (e) If a word contains three or more sequential repetitions of the same character, it is assumed that the underlying word has been artificially lengthened and its actual form can be retrieved by shortening of the lengthened sequence. By a check against an US-English dictionary, the most likely form of the underlying word is refined furthermore—though, given two or more lengthened sequences, the search for the correct spelling is not exhaustive. E.g., *'coooool'* becomes *'cool'*, *'niiiice'* becomes *'nice'*, *'hellllooo'* becomes *'hello'*.

The corrections[6] are applied word by word; each modification is applied on top of any previous modifications. Also, all capital letter noise words remain in all upper case style even after modification. Now, the text is rebuilt by joining the (possibly modified) tokens. Finally, it is ready to be segmented into sentences, tokenized with NLTK's `word_tokenize()` and tagged using the Stanford POS Tagger.

The following text normalization example demonstrates the transformation of an especially noisy text to showcase most of the corrections.

```
>>> normalize_text("YOu know what they say? ? ? some things in life are bad---they
    can reaaaallllly make you m@d----- Other things just make you $W34R (and
    cur$$$e). When you're chewing on l!fe's gristle...... don't g r u m b l e -->
    give aaaaaaaaaaaaaaaaaaaaaaaaa wh!stle!!")
"You know what they say??? some things in life are bad -- they can reaally make you
    mad -- Other things just make you SWEAR (and curse). When you're chewing on
    life's gristle... don't grumble -- give a whistle!!"
```

5.2.3 Imperative Detection

Typically, an imperative consists of a verb in base form at or close to the beginning of a sentence. An imperative can appear in the form of a statement (e.g., *"Dave, stop."*, *"Just take a stress pill and think things over."*) or disguised as a question (e.g., *"Stop, will you?"*, *"Stop it, please?"*). The POS-tag `<VB>` refers to a verb in base form, though its use is not limited to imperatives: For instance, the verb *'do'* in the sentence *"I'm afraid I can't do that."*, is in the base form too. Furthermore, in rare cases, an imperative might start with a modal verb (e.g., *"Will you stop, please?"*); modal verbs are always tagged as `<MD>`, regardless of form and tense of the modal verb.

Therefore, we define an imperative by its possible grammatical structures. We develop an algorithm (listing 5.1) that uses the position of the `<VB>`-tag (or the `<MD>-tag`) and shallow parsing—also known as *chunking*—to extract specific POS patterns, in order to determine if a sentence is an imperative or not. We define `VB-Phrase` (verb phrase) and `Q-Tag` (question-tag) patterns that are exclusive to imperatives (listing 5.2). Also, it is practical to split imperatives into two categories: commands (sentences with any type of final punctuation mark—including none at all—but with the exception of a question mark) and requests (sentences ending in a question mark). As an example, let's consider these imperative sentences and their POS-tagged[7] versions:

A sentence starting with the `VB-Phrase` pattern `<UH><,><VB>` is a definite imperative: *"Please, open the pod bay doors, HAL."*

```
[('Please', 'UH'), (',', ','), ('open', 'VB'), ('the', 'DT'), ('pod', 'NN'), ('bay',
     'NN'), ('doors', 'NNS'), (',', ','), ('HAL', 'NNP'), ('.', '.')]
```

A question that ends with the `Q-Tag` pattern `<,><MD><RB><PRP><.>` and starts with a `<VB>`-tag and indicates an imperative: *"Stop, will you?"*

```
[('Stop', 'VB'), (',', ','), ('will', 'MD'), ('you', 'PRP'), ('?', '.')]
```

[6]Automatic spellcorrection, on the other hand, is not applied as the tested spellcheckers failed some simple, random autocorrection tests and the original sentences would become distorted too much.

[7]`UH` refers to an *interjection,* `RB` is the tag for *adverbs* and `PRP` is short for *personal pronoun.*

```
from nltk.tree import Tree


def is_imperative(tagged_sent):
    """
    :param tagged_sent: Expects a normalized, POS-tagged sentence--a list of tuples.
    :return: True if the given sentence is an imperative, otherwise False.
    >>> is_imperative([('Stop', 'VB'), ('it', 'PRP'), ('!', '.')])
    True
    """

    # if the sentence is not a question...
    # (checks the last POS-tag tuple and whether the token it holds is not a '?')
    if tagged_sent[-1][0] != "?":
        # catches simple imperatives, e.g., "Open the pod bay doors, HAL!"
        # (checks the first POS-tag tuple and whether its tag is 'VB' or 'MD')
        if tagged_sent[0][1] == "VB" or tagged_sent[0][1] == "MD":
            return True

        # catches imperative sentences starting with words like 'please', 'just', 'you', etc.
        # E.g., "Dave, stop.", "Just take a stress pill and think things over."
        else:
            chunk = get_chunks(tagged_sent)
            # check if the first chunk of the sentence is a VB-Phrase
            if type(chunk[0]) is Tree and chunk[0].label() == "VB-Phrase":
                return True

    # Questions can be imperatives too, let's check if this one is
    else:
        # check if sentence contains the word 'please'
        pls = len([w for w in tagged_sent if w[0].lower() == "please"]) > 0
        # catches requests disguised as questions, e.g., "Open the doors, HAL, please?"
        if pls and (tagged_sent[0][1] == "VB" or tagged_sent[0][1] == "MD"):
            return True

        chunk = get_chunks(tagged_sent)
        # catches imperatives ending with a Question tag
        # and starting with a verb in base form, e.g., "Stop it, will you?"
        elif type(chunk[-1]) is Tree and chunk[-1].label() == "Q-Tag":
            if (chunk[0][1] == "VB" or
                (type(chunk[0]) is Tree and chunk[0].label() == "VB-Phrase")):
                return True

    return False
```

Listing 5.1: Imperative Detection program.

5.3 Machine Learning

The machine learning related tasks make use of `numpy`, `scipy` and the `sklearn` package. The results for each model are stored alongside the associated feature selections and tuning parameters (where applicable) using a `DataFrame` per learning algorithm.

```python
from nltk import RegexpParser


def get_chunks(tagged_sent):
    """
    :param tagged_sent: Expects a POS-tagged sentence--a list of tuples.
    :return: nltk.tree.Tree
    >>> get_chunks([('Please','UH'), ('stop', 'VB'), ('it', 'PRP')])
    Tree('S', [Tree('VB-Phrase', [('Please', 'UH'), ('stop', 'VB')]), ('it', 'PRP')])
    """
    chunkgram = r"""VB-Phrase: {<DT><,>*<VB>}
                    VB-Phrase: {<RB><VB>}
                    VB-Phrase: {<UH><,>*<VB>}
                    VB-Phrase: {<UH><,><VBP>}
                    VB-Phrase: {<PRP><VB>}
                    VB-Phrase: {<NN.?>+<,>*<VB>}
                    Q-Tag: {<,><MD><RB>*<PRP><.>*}"""
    chunkparser = RegexpParser(chunkgram)
    return chunkparser.parse(tagged_sent)
```

Listing 5.2: Method to chunk a sentence into verb phrases and question-tags.

5.3.1 Feature Selection

Univariate Ranking

The `sklearn.feature_selection` package is employed for the ANOVA implementation of the T-test (the F-test, function `f_classif()`) and for the Mutual Information ranking (`mutual_info_classif()`). The `scipy.stats` package provides the implementation of the Wilcoxon-ranksum-test (`ranksums()`).

Lasso

Feature Selection via the Lasso is executed with the help of the `LassoCV` class in the `sklearn.linear_model` package. It selects the best model using 5-fold cross-validation. Selections are searched for within a threshold range of $[0.000005-0.25]$, with a `stepsize` of 0.005 and 250 `alphas` per step.

Recursive Feature Elimination

Feature Selection via Recursive Feature Elimination is performed using class `RFE` which, in turn, uses `LogisticRegression` (L2 regularization, default settings) as an estimator to internally score the selections.

5.3.2 Models

Logistic Regression

All Logistic Regression models use scikit-learn's `LogisticRegression` class from the `sklearn.linear_model` package with standard parameters, with the explicit use of the L2 regularization and a fixed random state to achieve reproducibility.

SVM

The linear SVM models (class `LinearSVC`) and the Gaussian SVM models (class `SVC`), both from the `sklearn.svm` package, use standard settings, except for a fixed random state to achieve reproducibility and the hyperparameters, which vary per subset and are determined via a grid-search.

The grid-searches are performed using `GridSearchCV` with 5-fold cross-validation and the `f1_score` from the `sklearn.metrics` package as the scoring function. The range of the hyperparameter search for both kernels (`gamma` only for the Gaussian) is:

```
    C_range = [0.01, 1.0, 0.1, 1.0, 10.0, 100.0, 1000.0]
    gamma_range = [0.01, 1.0, 0.1, 1.0, 10.0, 100.0, 1000.0]
```

Chapter 6

Evaluation

This chapter presents the results for both datasets separately. The exploration of the feature selections per selection method is followed by a performance analysis of the generated models—across classifiers and selection methods.

6.1 Results Wiki Dataset

The Wikipedia Talk Corpus contains user comments collected from discussions found on the Talk Pages of English Wikipedia. The comments were posted within a timeframe of 14 years—between 2001 and 2015; the primary languages used are American English and British English. The average comment of the Wiki Corpus consists of 44 words (median 27, minimum 3, maximum 250 words) contained in 3.8 sentences (median 3, minimum 1, maximum 25 sentences). We hold out a set of 1661 samples for the final validation (table 6.1). The remainder of 4986 samples is what we generate our feature selections out of.

6.1.1 Feature Selections

Appendix A.3 shows the complete feature rankings for the univariate methods and Recursive Feature Elimination. The whole set of Lasso selections is listed in appendix A.4. Looking at figure 6.1, we can see that the univariate methods start out with good agreement on the first few features and then continue with increasing variance in ranks. A somewhat similar trend can be observed by juxtaposing the combined univariate methods and the RFE ranking (figure 6.2), though there the level of disagreement is much

Table 6.1: Wiki Dataset, 3:1 separation in development set and validation set.

	OK	**aggressive**	**total**
dev set	2463	2523	4986
val set	821	840	1661
total	3284	3363	6647

higher. Since the Lasso selections are not based on rankings, we use a Venn diagram (figure 6.3) to show where the univariate, the RFE and the Lasso selections intersect and which feature preferences are unique to a particular method. Here are the top ten features per selection method:

Filter Method: T-Test

1. Blacklisted words count divided by the total word count (60)
2. Blacklisted words count (40)
3. Blacklisted words count divided by the number of sentences (56)
4. Long words count (1)
5. Long words count divided by the total word count (62)
6. Ratio of negative subjective words and the total count of subjective words (24)
7. Period ratio (4)
8. Highlighters (*'()") count (7)
9. Count of ALL-CAPS words divided by the total word count (61)
10. Connectors (,;:–) ratio (51)

Filter Method: Wilcoxon Ranksum Test

1. Blacklisted words count divided by the total word count (60)
2. Blacklisted words count divided by the number of sentences (56)
3. Blacklisted words count (40)
4. Long words count (1)
5. Maximum wordlength (12)
6. Connectors (,;:–) ratio (51)
7. Long words count divided by the total word count (62)
8. Average wordlength (13)
9. Maximum sentence length in words (18)
10. Average sentence length in words (19)

Filter Method: Mutual Information

1. Blacklisted words count divided by the number of sentences (56)
2. Blacklisted words count divided by the total word count (60)
3. Blacklisted words count (40)
4. Long words count divided by the total word count (62)
5. Ratio of negative subjective words and the total count of subjective words (24)
6. Connectors (,;:–) ratio (51)
7. Count of ALL-CAPS words divided by the total word count (61)
8. Long words count (1)
9. Average wordlength (13)
10. Highlighters (*'()") count (7)

Wrapper Method: Recursive Feature Elimination

1. Blacklisted words count divided by the total word count (60)
2. Blacklisted words count (40)
3. Ratio of negative subjective words and the total count of subjective words (24)
4. Blacklisted words count divided by the number of sentences (56)
5. Highlighters (**'()"*) count (7)
6. Connectors (*,;:–*) ratio (51)
7. Polite words count (44)
8. Count of imperative sentences (45)
9. Long words count divided by the total word count (62)
10. Count of ALL-CAPS words divided by the total word count (61)

Embedded Method: Lasso

The following Lasso selection is the smallest of its kind; it contains 20 items sorted by index:

- Long words count (1)
- WH-Pronouns count (2)
- Highlighters (**'()"*) count (7)
- Paragraph count (8)
- Sentence count (9)
- Edit distance of 'fixed' words (16)
- Median sentence length (20)
- Count of subjective words divided by the total number of words (22)
- Ratio of negative subjective words and the total count of subjective words (24)
- Count of blacklisted words written in ALL-CAPS (36)
- Noise count (38)
- Blacklisted words count (40)
- Polite words count (44)
- Ellipsis count divided by the number of sentences (47)
- Connectors (*,;:–*) ratio (51)
- Blacklisted words count divided by the number of sentences (56)
- Blacklisted words count divided by the total word count (60)
- Count of ALL-CAPS words divided by the total word count (61)
- Long words count divided by the total word count (62)
- Count of numeric values divided by the total word count (64)

Rejected Features

Figure 6.3 shows the most relevant features determined by the intersection of prefered features across selection methods. Computing the intersections from the reverse end results in the least informative features across the six selections methods: The count of

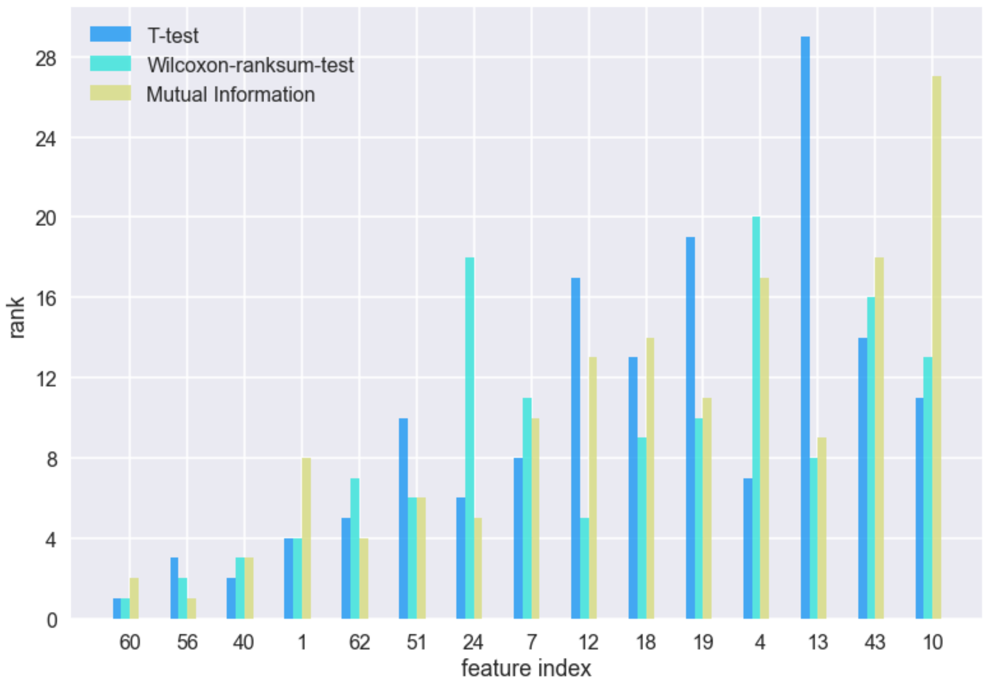


Figure 6.1: Wiki Dataset. The plot shows the level of (dis)agreement within the univarate feature ranking for the combined top 15 features. The x-axis refers the features in the order of their combined ranks. The bars per feature relate to the ranking (y-axis) of the feature according to the corresponding ranking method.

spaced words (15), the length of the shortest sentence (21), the count of em dashes (25), the ratio of superlative adverbs (30), the count of 'fixed' words (42), the per-word-ratio of 'fixed' words (63) and the ratio of indirect imperatives (65).

6.1.2 Subset Models Results

Training and testing of the models, based on each of the selections, produces 359 results per classifier—or 1077 results in total[1]—which are pictured in the following line graphs (please note that the y-axis is truncated). The three plots reveal that the full selection of features performs comparatively well. Even so, only a third of the full set is required to achieve an improved classification result. The baselines range from 85.2% to 88.4%, and the overall best result accomplishes an F_1-score of 89.4% using 40 features (table 6.2). The RFE method consistently produces the smallest, most useful subsets that are the first to surpass the baseline. The ranksum method also sets itself apart from the rest... by its initial stagnation. At the 18 features mark, the ranksum score reliably rockets up, powered by the Negative-Subjectivity-Ratio (24), which happens to be a steadfast booster across selection types and classifiers. Other features that typically lead to an increase are the per-word-ratio of all capitalized words (61), blacklisted words features (40, 60), the connectors ratio (51), and—among the linear classifiers—the

[1] $(5 \cdot (68 - 1)$ ranking selections $+ 23$ Lasso selections $+ 1$ full set$) \cdot 3$ classifiers $= 1077$ models

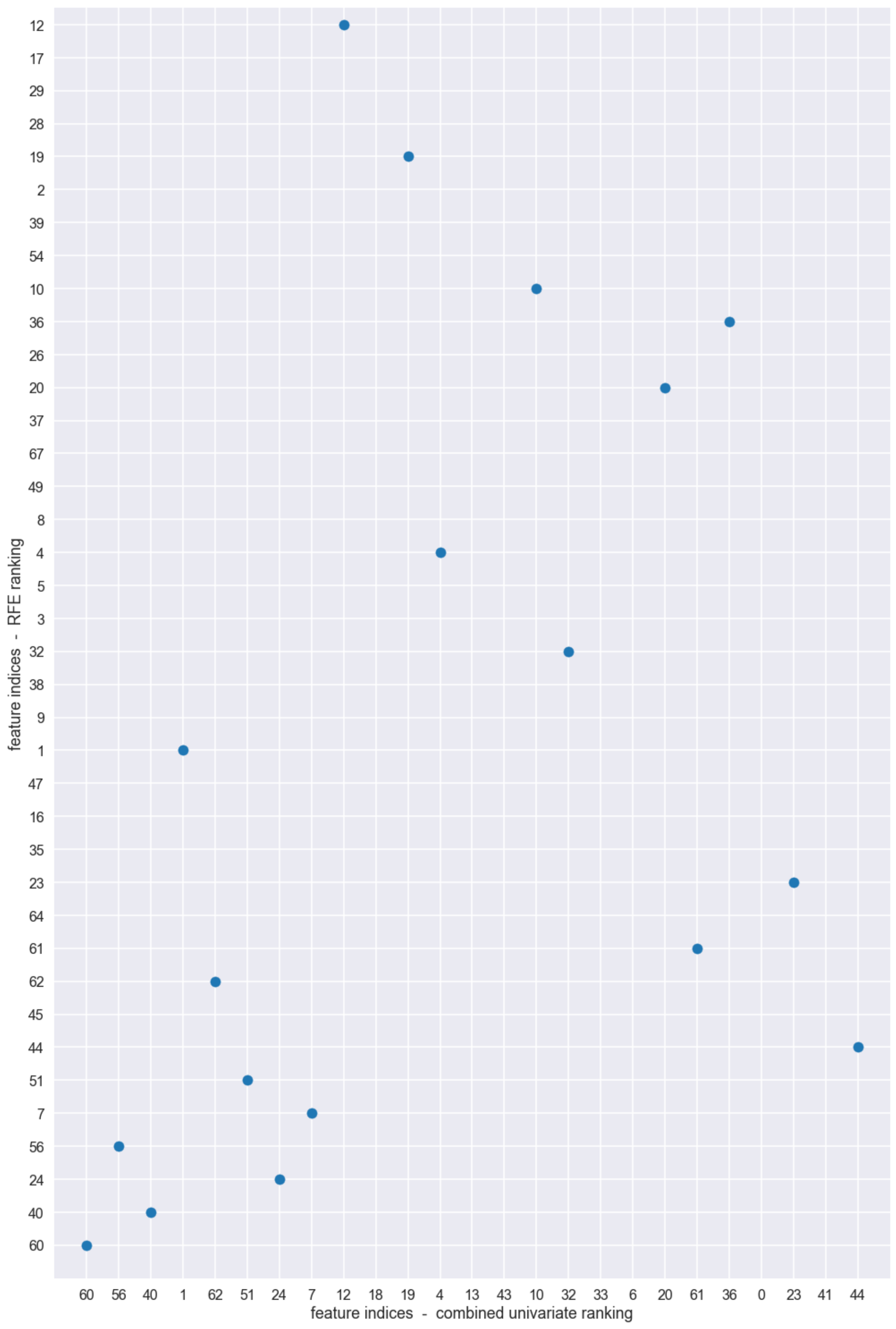


Figure 6.2: Wiki Dataset. A comparison of the RFE and combined univariate ranking shows a low level of general agreement on the importance of the single features. The scatterplot encompasses the ranked top 25 combined univariate features on the x-axis and the RFE ranking (top 63 features) on the y-axis. A diagonal scatter would indicate a perfect agreement. A correlation coefficient of -0.08 confirms the impression that there is no linear relationship between the RFE and univariate selections.

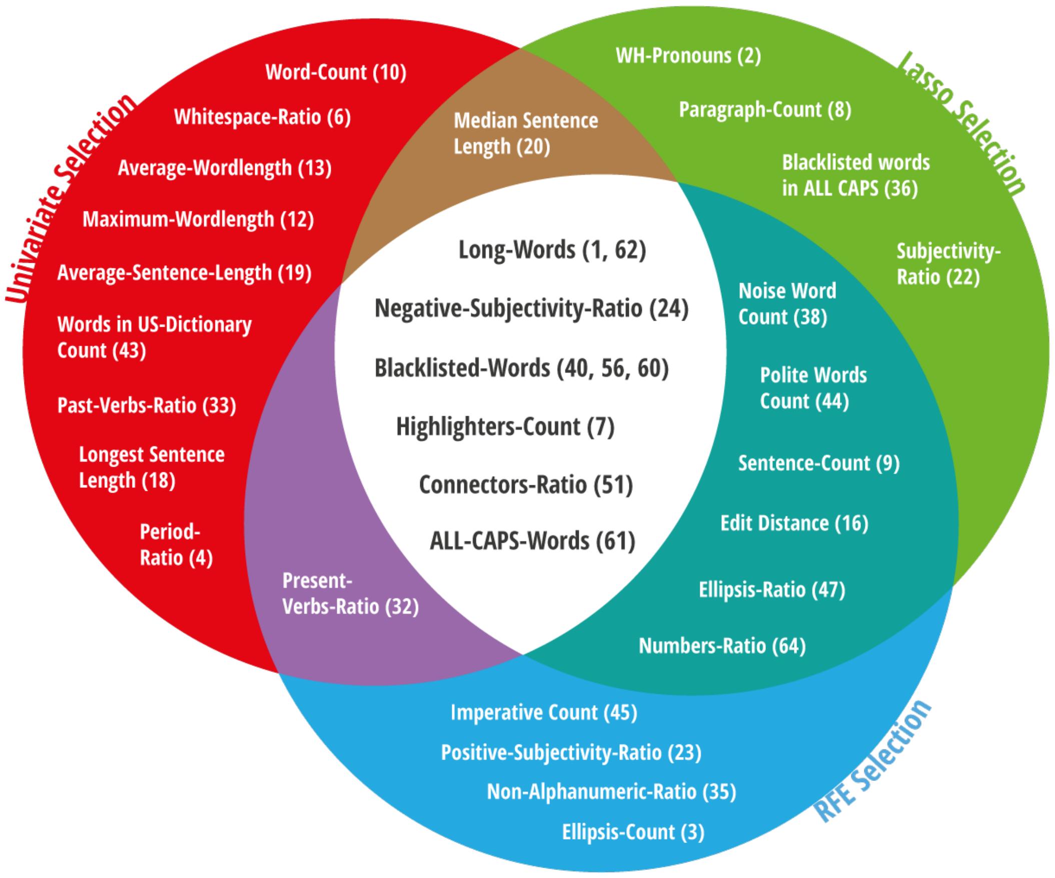


Figure 6.3: Wiki Dataset. Venn diagram of the top 20 features per selection type. The univariate selection, based on the combined ranking, and the Lasso selection agree on 10 features, which is also the same amount the univariate selection and the RFE selection have in common. The RFE and the Lasso share 15 out of 20 features. The 9 features all selections have in common relate to specific punctuation-types—the highlighters- (7) and the connectors-category (51)—and to word frequencies and ratios: longs words (1, 62), negative subjective words (24), blacklisted words (40, 56, 60) and words written in ALL-CAPS (61).

Exclamation-Mark-Ratio (0). Interestingly, the Present-Verbs-Ratio (32) frequently has a positive impact in the SVM models, whereas it is rather associated with a drop in the LR models. In any case, we must keep in mind that the features in the model are not isolated: the jumps or drops in performance are not necessarily based on the predictive power of a single feature, but rather on feature interactions. In the words of machine learning researcher and co-inventor of the SVM kernel-trick, Isabelle Guyon [6]:

> [...] *a variable that is completely useless by itself can provide a significant performance improvement when taken with others.*

It is, nevertheless, most likely that the single most useful feature of the Wiki Dataset is related to blacklisted words, as agreed upon by 5 out of 5 ranking methods: For instance, the Gaussian SVM model consisting of only the per-words-ratio of blacklisted

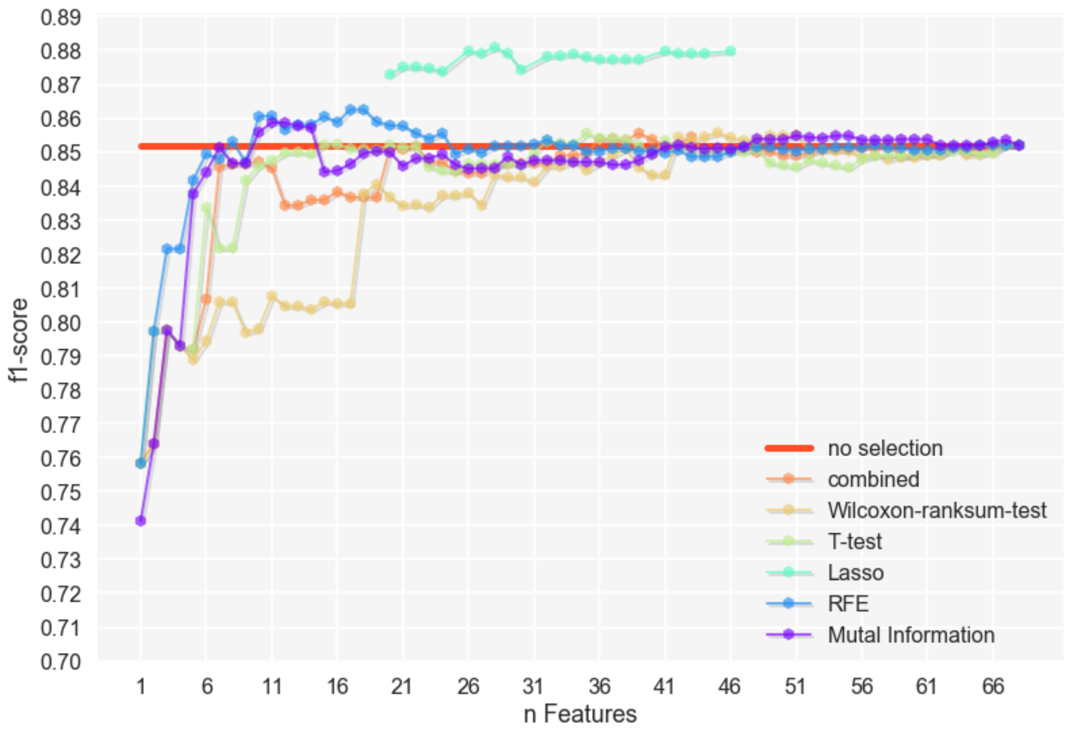


Figure 6.4: Feature selection results for the Wiki Dataset using Logistic Regression. The isolated markers that hover steadily, centered at the top immediately suggest, that the Lasso selection method is a class of its own: The best subset outperforms the baseline by almost 3 percent, using less than 30 features. However, the first to clearly outperform the full set is an RFE subset of 10 features. The least useful subsets belong to the ranksum selections.

words (60) accomplishes an impressive F_1-score of 81.4%, with a precision of 96.7% and recall of 70.2%. Looking at the smallest RFE subsets, we can see that the addition of the Negative-Subjectivity-Ratio (24) results in a steep rise to an F_1-score of 86% (where $n = 3$), while behind the scenes there is a shift in balance: Recall increases significantly from 70.2% to 82.1%, at the same time precision drops by 6.4% to 90.2%.

According to the additional criteria that aims to find a compact, yet high-quality subset (tables 6.2 and 6.3), we single out the final model; it consists of the leading 23 features of the combined ranking:

1. Blacklisted words count divided by the total word count (60)
2. Blacklisted words count divided by the number of sentences (56)
3. Blacklisted words count (40)
4. Long words count (1)
5. Long words count divided by the total word count (62)
6. Connectors (,;:–) ratio (51)
7. Ratio of negative subjective words and the total count of subjective words (24)
8. Highlighters (*'()") count (7)

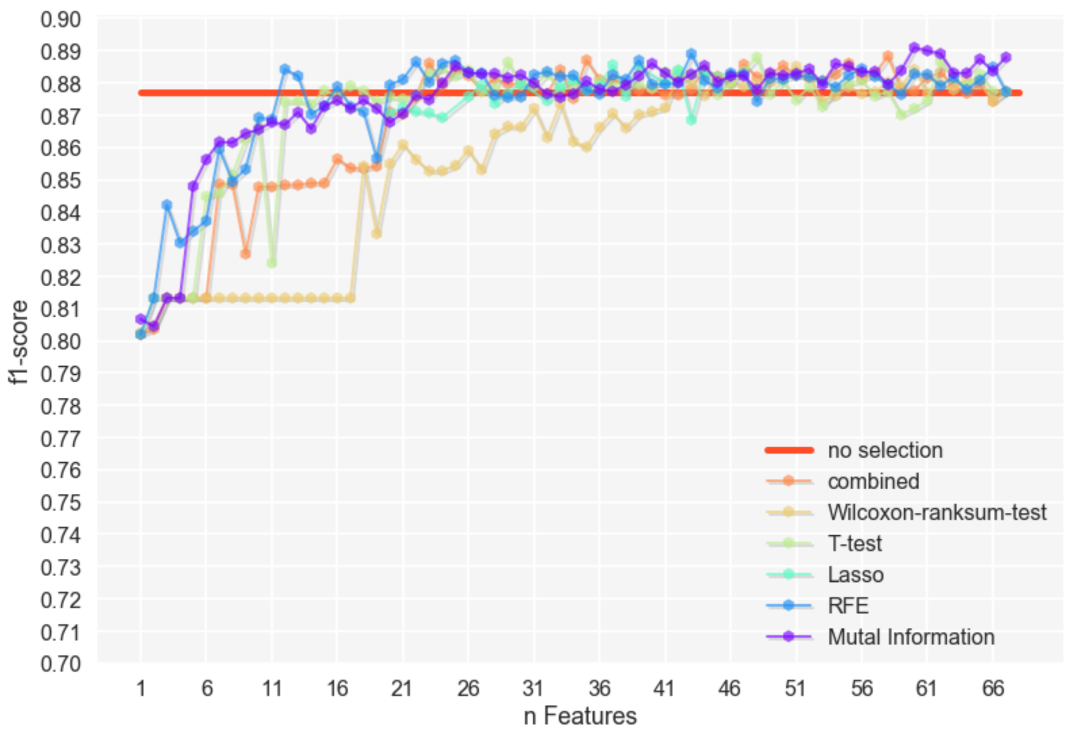


Figure 6.5: Feature Selection results for the Wiki Dataset using linear SVM. Here, the significance of the Lasso subsets has diminished to slightly below average. Only the ranksum subsets generously undercut their performance. The first peak above the baseline, using 12 Features, is again an RFE subset. Though it is important to note, that the baseline here is at almost 88%, surpassing the Logistic Regression baseline by 2.5% and performance, in general, has considerably improved in comparison to the Logistic Regression scores.

9. Maximum wordlength (12)
10. Maximum sentence length in words (18)
11. Average sentence length in words (19)
12. Period ratio (4)
13. Average wordlength (13)
14. Count of words that appear in an US English dictionary (43)
15. Total word count (10)
16. Present-Verbs-Ratio (32)
17. Past-Verbs-Ratio (33)
18. Whitespace ratio (6)
19. Median sentence length in words (20)
20. Count of ALL-CAPS words divided by the total word count (61)
21. Count of blacklist words written in ALL-CAPS (36)
22. Exclamation-Mark-Ratio (0)
23. Positive-Subjectivity-Ratio (23)

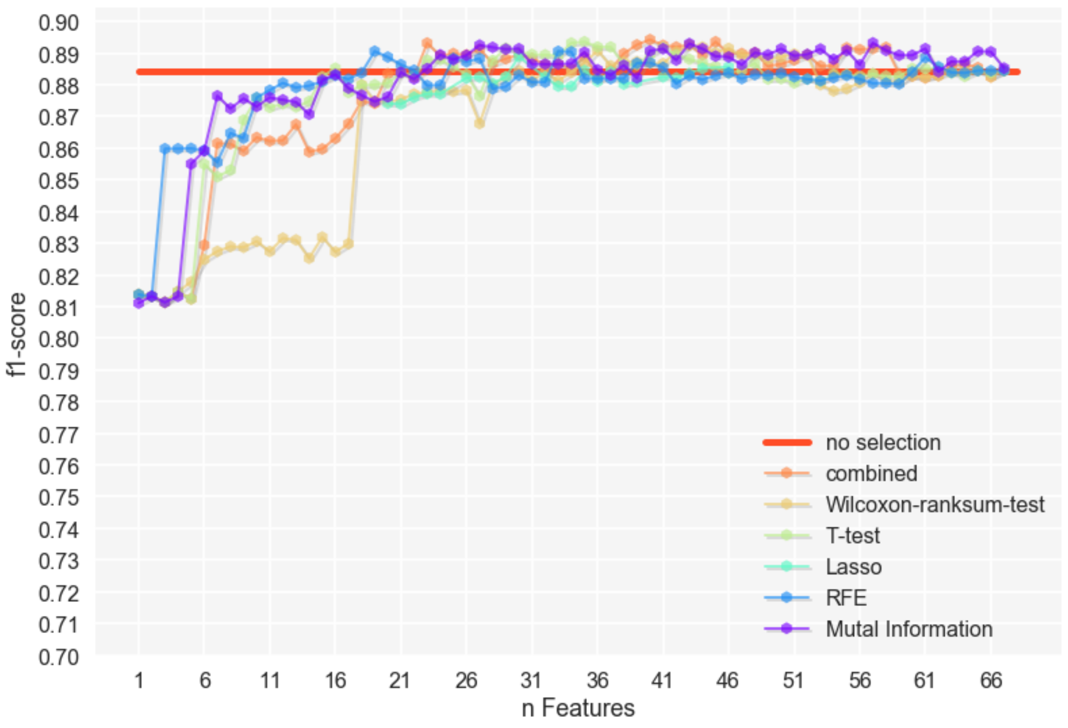


Figure 6.6: Feature selection results for the Wiki Dataset using Gaussian SVM. At first, the plot of the Linear SVM and the Gaussian SVM appear almost identical: The same pattern with the ranksum falling behind and the RFE cutting through the baseline first—here with a subset of 19 features. When examined more closely, we see that the overall performance has again shifted upward, with the baseline now at 88.4%. Moreover, a significant amount of subsets scores above 89%.

6.2 Results Martin Dataset

The Martin Corpus contains user comments collected from discussions below politial news articles. The timeframe in which these comments were posted is relatively short (4 months); the primary language used is American English. The average comment of the Martin Dataset consists of 47 words (median 42, minimum 3, maximum 182 words) contained in 3.5 sentences (median 3, minimum 1, maximum 17 sentences). We hold out a set of 258 samples for the final validation (table 6.4). The remainder of 776 samples is what we generate our feature selections out of.

6.2.1 Feature Selections

Appendix A.5 shows the complete feature rankings for the univariate methods and Recursive Feature Elimination. The entirety of the Lasso selections is listed in appendix A.6. Looking at figure 6.7, we can see that each univariate method arrives at a different ranking, though the differences seem moderate compared to the level of disagreement between the combined univariate methods and the RFE ranking (figure 6.8). Since the Lasso selections are not based on rankings, we use a Venn diagram (figure 6.9)

Table 6.2: Wiki Dataset. Comparison of the *baseline results* using the full set of 68 features, the *best results* in terms of the highest F_1-score and the *top choices* according to our additional heuristic that aims to minimize n.

	`gamma`	`C`	**subset**	n	**F_1-score**	**precision**	**recall**
Logistic Regression		1.0	baseline	68	0.852	0.8919	0.8155
		1.0	best	28	0.8809	0.9417	0.8274
		1.0	*top choice*	26	0.8797	0.9392	0.8274
Linear SVM		100.0	baseline	68	0.8772	0.9475	0.8167
		100.0	best	60	0.8909	0.9234	0.8607
		100.0	*top choice*	35	0.8871	0.9436	0.8369
Gaussian SVM	0.1	100.0	baseline	68	0.8844	0.9304	0.8429
	0.1	1000.0	best	40	0.8942	0.9305	0.8607
	1.0	100.0	*top choice*	23	0.8932	0.9338	0.856

Table 6.3: Wiki Dataset. Gaussian SVM, top 10 results sorted by feature subset size n.

n	**selection type**	**F_1-score**	**precision**	**recall**	**TN**	**FN**	**FP**	**TP**	`gamma`	`C`
23	combined	0.8932	0.9338	0.856	770	121	51	719	1.0	100.0
27	Mutual Information	0.8925	0.9395	0.85	775	126	46	714	0.1	1000.0
34	T-test	0.8932	0.9338	0.856	770	121	51	719	0.1	1000.0
35	T-test	0.8936	0.9304	0.8595	767	118	54	722	0.1	1000.0
39	combined	0.8926	0.9269	0.8607	764	117	57	723	0.1	1000.0
40	combined	0.8942	0.9305	0.8607	767	117	54	723	0.1	1000.0
43	Mutual Information	0.893	0.9237	0.8643	761	114	60	726	0.1	1000.0
43	combined	0.8926	0.9269	0.8607	764	117	57	723	0.1	1000.0
45	combined	0.8936	0.9304	0.8595	767	118	54	722	0.1	1000.0
57	Mutual Information	0.8932	0.9455	0.8464	780	129	41	711	0.1	100.0

Table 6.4: Martin Dataset, 3:1 separation in development set and validation set.

	OK	**aggressive**	**total**
dev set	390	386	776
val set	130	128	258
total	520	514	1034

to show where the univariate, the RFE and the Lasso selections intersect and which feature preferences are unique to a particular method. Here are the top ten features per selection method:

Filter Method: T-Test

1. Long words count (1)
2. Count of words that appear in an US English dictionary (43)
3. Total word count (10)

4. Lexical diversity or Type-to-Token Ratio (11)
5. Blacklisted words count (40)
6. Grammatical diversity or POS-Type-to-Word-Ratio (26)
7. Count of Urban-Dictionary-only words divided by the total word count (59)
8. Number of non-alphanumeric characters divided by the total character count (35)
9. Blacklisted words count divided by the number of sentences (56)
10. Ratio of negative subjective words and the total count of subjective words (24)

Filter Method: Wilcoxon-Ranksum-Test

1. Long words count (1)
2. Count of words that appear in an US English dictionary (43)
3. Total word count (10)
4. Lexical diversity or Type-to-Token Ratio (11)
5. POS-Type-to-Word-Ratio (26)
6. Count of Urban-Dictionary-only words divided by the total word count (59)
7. Number of non-alphanumeric characters divided by the total character count (35)
8. Average sentence length in words (19)
9. Maximum wordlength (12)
10. Maximum sentence length in words (18)

Filter Method: Mutual Information

1. Total word count (10)
2. Count of words that appear in an US English dictionary (43)
3. Blacklisted words count divided by the number of sentences (56)
4. Long words count (1)
5. Blacklisted words count (40)
6. Long words count divided by the total word count (62)
7. Count ALL-CAPS words divided by the total word count (61)
8. Blacklisted words count divided by the total word count (60)
9. Ratio of negative subjective words and the total count of subjective words (24)
10. Number of non-alphanumeric characters divided by the total character count (35)

Wrapper Method: Recursive Feature Elimination

1. Blacklisted words count (40)
2. Long words count (1)
3. Count of Urban-Dictionary-only words divided by the total word count (59)
4. Ratio of negative subjective words and the total count of subjective words (24)
5. Number of periods, question- and exclamation marks divided by the sentence count (46)
6. Average length of repeated punctuations (34)

7. Paragraph count divided by the sentence count (54)
8. Number of words modified ('fixed') during preprocessing (42)
9. Count ALL-CAPS words divided by the total word count (61)
10. Blacklisted words count divided by the number of sentences (56)

Embedded Method: Lasso

The following Lasso selection is the smallest of its kind; it contains 11 items sorted by index:

- Long words count (1)
- Question-Mark-Ratio (5)
- Paragraph count (8)
- Lexical diversity or Type-to-Token Ratio (11)
- Ratio of negative subjective words and the total count of subjective words (24)
- Count of blacklisted words written in ALL-CAPS (36)
- Noise count (38)
- Count of Urban-Dictionary-only words (39)
- Blacklisted words count (40)
- Unknown words count (41)
- Number of periods, question- and exclamation marks divided by the sentence count (46)

Rejected Features

Figure 6.9 shows the most relevant features determined by the intersection of prefered features across selection methods. Computing the intersections from the reverse end results in the least informative features across the six selections methods: The count of word lengthenings (17), the count of em dashes (25), the per-sentence-ratio of unknown words (57) and the ratio of indirect imperatives (65).

6.2.2 Subset Models Results

Training and testing of the models, based on each of the selections, produces 361 results per classifier—or 1083 results in total[2]—which are pictured in the following line graphs (please note that the y-axis is truncated).

Looking at the plots, the positive impact of feature selection is immediately apparent. For all three classifiers, the baseline is an F_1-score of roughly 77%. The best scores range from 81% up to almost 83%. Analogous to the Wiki Dataset, the RFE consistently is the first to cross the baseline, while the ranksum method seems to be particularly inefficient in creating useful small subsets. The first marker of the RFE selection (belonging to the count of blacklisted words feature (40)) is cut off in the plots to increase their readability, noting that the F_1-score for this one-feature selection is an identical 45 percent across all three classifiers. This is suprisingly low at first, however, considering that the precision

[2] $(5 \cdot (68 - 1)$ ranked selections $+ 25$ Lasso selections $+ 1$ full set$) \cdot 3$ classifiers $= 1083$ models

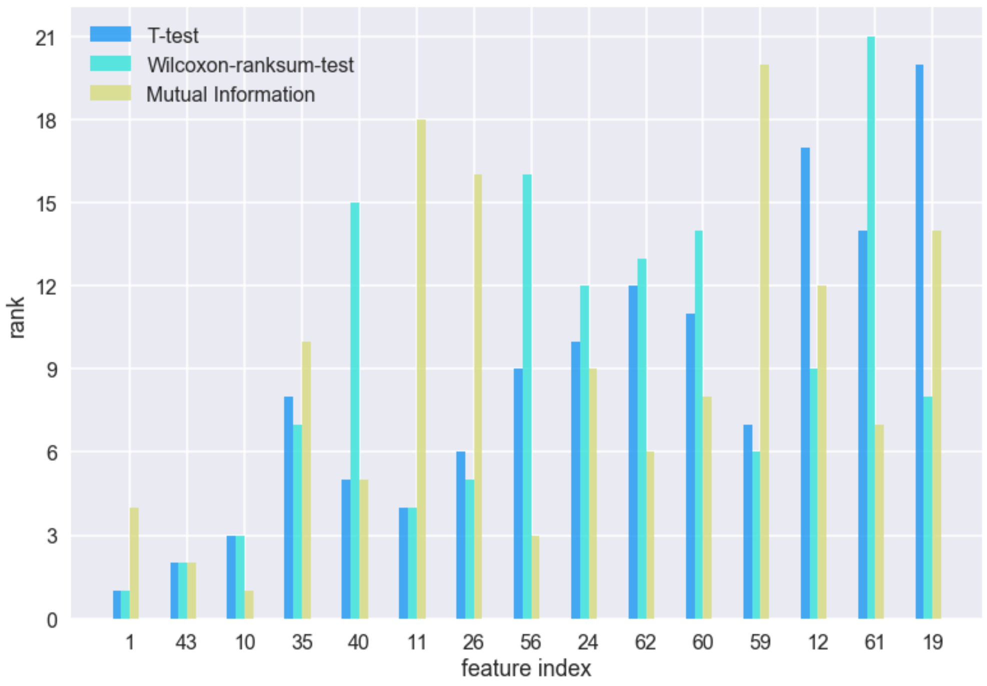


Figure 6.7: Martin Dataset. The plot shows the level of agreement within the univarate feature ranking for the combined top 15 features. While the T-test ranking and the ranksum ranking often correlate, the Mutual Information based ranking slightly tends towards being the outlier. Looking at the maximum rank on the y-axis, 21, it shows that the magnitude of disagreement in general is moderate, considering that the plot focusses on 15 out of 68 features.

rate for that same feature is almost 93%, while recall is below 30%, we can conclude that this feature may be in fact very useful if paired with complementary features. Also unexpectedly, the long words count (1) and the total word count (10) individually achieve an F_1-score of over 70%; both with a clear imbalance in favor of the recall value (ranging from 78 to 88%).

On close inspection of sharp jumps and drops a pattern emerges: The long words feature (1), as well as the Negative-Subjectivity-Ratio (24), individually lead to an upsurge once one of them is added to a model. The addition of the ratio of the Positive-Subjectivity-Ratio (23), on the other hand, more often than not is marked by a drop. Other features, such as the per-word-ratio of words found in the Urban Dictionary only (59) and the ratio of non-alphanumeric characters (35), tend to induce an upward trend, yet there is likely a dependence on the presence of one or more other features; the grammatical diversity feature (26), however, often leads to a decline in the linear classifiers, whereas its effect was more likely to be positive in the Gaussian models.

Ensuing this visual impression of the overall significance of feature selection on the Martin Dataset, table 6.5 outlines a direct comparison of the three learning algorithms. Interestingly, the results differ only marginally. More detailed listings of the results per classifier are enclosed in appendix A.3.

According to the additional criteria that aims to find a compact, yet high-quality

Table 6.5: Martin Dataset. Comparison of the *baseline results* using the full set of 68 features, the *best results* in terms of the highest F_1-score and the *top choices* according to our additional heuristic that aims to minimize n.

	`gamma`	`C`	**subset**	n	**F1-score**	**precision**	**recall**
Logistic Regression		1.0	baseline	68	0.7711	0.7934	0.75
		1.0	best	14	0.8189	0.8254	0.8125
		1.0	*top choice*	4	0.8142	0.824	0.8047
Linear SVM		100.0	baseline	68	0.7661	0.7917	0.7422
		100.0	best	17	0.8273	0.8512	0.8047
		100.0	*top choice*	10	0.8178	0.8487	0.7891
Gaussian SVM	0.1	10.0	baseline	68	0.7711	0.7934	0.75
	0.01	100.0	best	47	0.8207	0.8374	0.8047
	1.0	10.0	*top choice*	6	0.813	0.8475	0.7813

Table 6.6: Martin Dataset. Linear SVM, top 10 results sorted by feature subset size n.

n	**selection type**	F_1**-score**	**precision**	**recall**	**TN**	**FN**	**FP**	**TP**	`C`
10	Mutual Information	0.8178	0.8487	0.7891	112	27	18	101	100.0
15	Mutual Information	0.8211	0.8559	0.7891	113	27	17	101	100.0
17	Mutual Information	0.8273	0.8512	0.8047	112	25	18	103	100.0
21	T-test	0.8189	0.8254	0.8125	108	24	22	104	100.0
22	T-test	0.8171	0.814	0.8203	106	23	24	105	100.0
42	combined	0.8185	0.8092	0.8281	105	22	25	106	100.0
48	combined	0.8217	0.8154	0.8281	106	22	24	106	100.0
48	T-test	0.8185	0.8092	0.8281	105	22	25	106	100.0
59	combined	0.8178	0.8487	0.7891	112	27	18	101	10.0
59	Ranksum	0.8178	0.8487	0.7891	112	27	18	101	10.0

subset (tables 6.5 and 6.6), we single out the final model; it consists of the leading 10 features of the Mutual Information ranking. The selected core features are listed above in section 6.2.1. Considering the impressive result of the best RFE selections, we propose an alternative core selection consisting of slightly divergent 4–6 features (for instance, the per-word-ratio of Urban-Dictionary-only words (59) is a unique choice by the RFE; see also section 6.2.1).

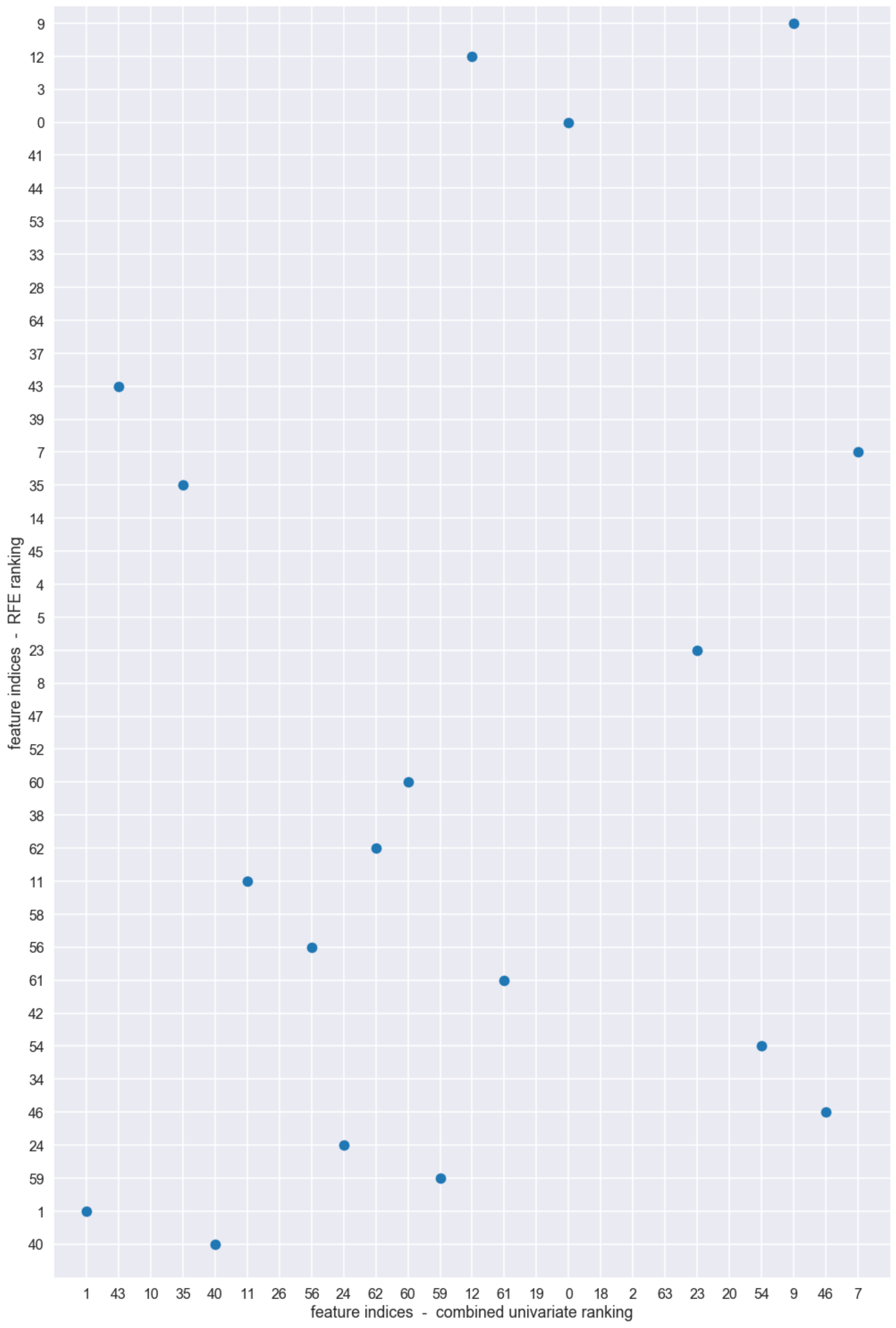


Figure 6.8: Martin Dataset. A comparison of the RFE and combined univariate ranking shows a low level of general agreement on the importance of the single features. The scatterplot encompasses the ranked top 25 combined univariate features on the x-axis and the RFE ranking (top 60 features) on the y-axis. A correlation coefficient of 0.19 confirms the impression that there is no linear relationship between the RFE and univariate selections.

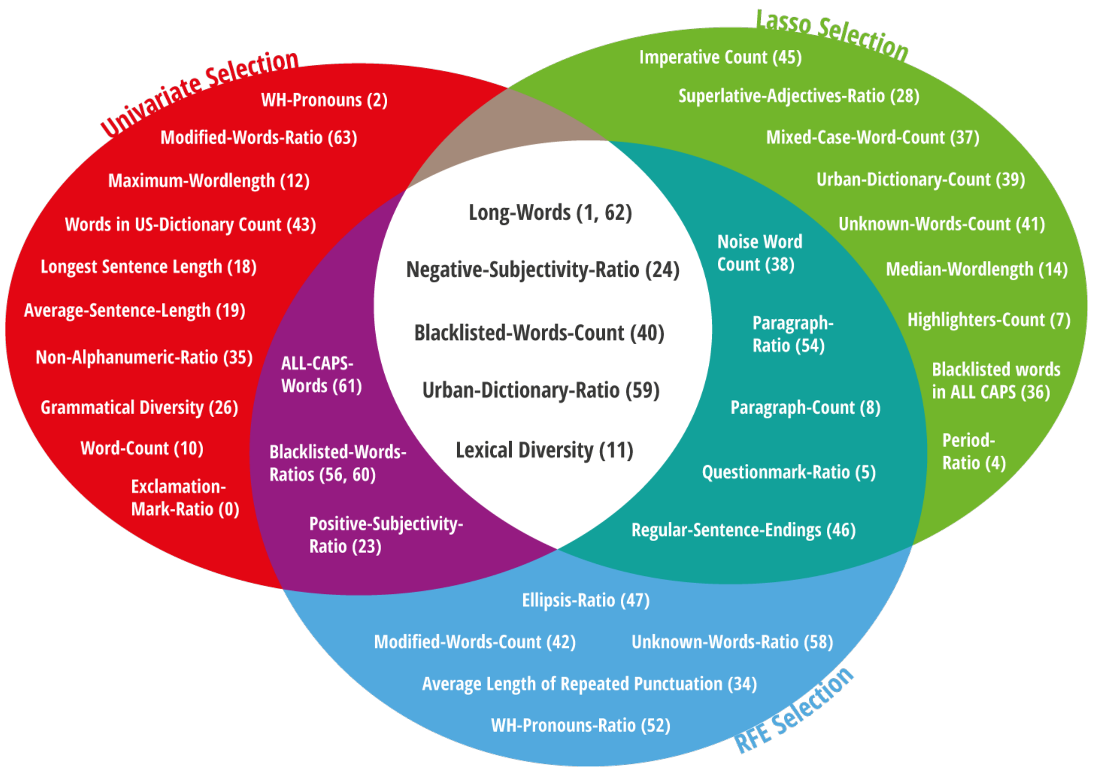

Univariate Selection
WH-Pronouns (2)
Modified-Words-Ratio (63)
Maximum-Wordlength (12)
Words in US-Dictionary Count (43)
Longest Sentence Length (18)
Average-Sentence-Length (19)
Non-Alphanumeric-Ratio (35)
Grammatical Diversity (26)
Word-Count (10)
Exclamation-Mark-Ratio (0)
Lasso Selection
Imperative Count (45)
Superlative-Adjectives-Ratio (28)
Mixed-Case-Word-Count (37)
Urban-Dictionary-Count (39)
Unknown-Words-Count (41)
Median-Wordlength (14)
Highlighters-Count (7)
Blacklisted words in ALL CAPS (36)
Period-Ratio (4)
Long-Words (1, 62)
Negative-Subjectivity-Ratio (24)
Blacklisted-Words-Count (40)
Urban-Dictionary-Ratio (59)
Lexical Diversity (11)
ALL-CAPS-Words (61)
Blacklisted-Words-Ratios (56, 60)
Positive-Subjectivity-Ratio (23)
Noise Word Count (38)
Paragraph-Ratio (54)
Paragraph-Count (8)
Questionmark-Ratio (5)
Regular-Sentence-Endings (46)
Ellipsis-Ratio (47)
Modified-Words-Count (42)
Unknown-Words-Ratio (58)
Average Length of Repeated Punctuation (34)
WH-Pronouns-Ratio (52)
RFE Selection

Figure 6.9: Martin Dataset. Venn diagram of the top 20 features per selection type. The univariate selection, based on the combined ranking, and the Lasso selection agree on 6 features only—which are in fact the core features shared by all. The RFE places itself in the center; it shares 10 features with the univariate selection and has 11 features in common with the Lasso selection. The 6 features all selections share relate to lexical diversity (11), longs words (1, 62), negative subjective words (24), blacklisted words (40) and words found in the Urban Dictionary only (59).

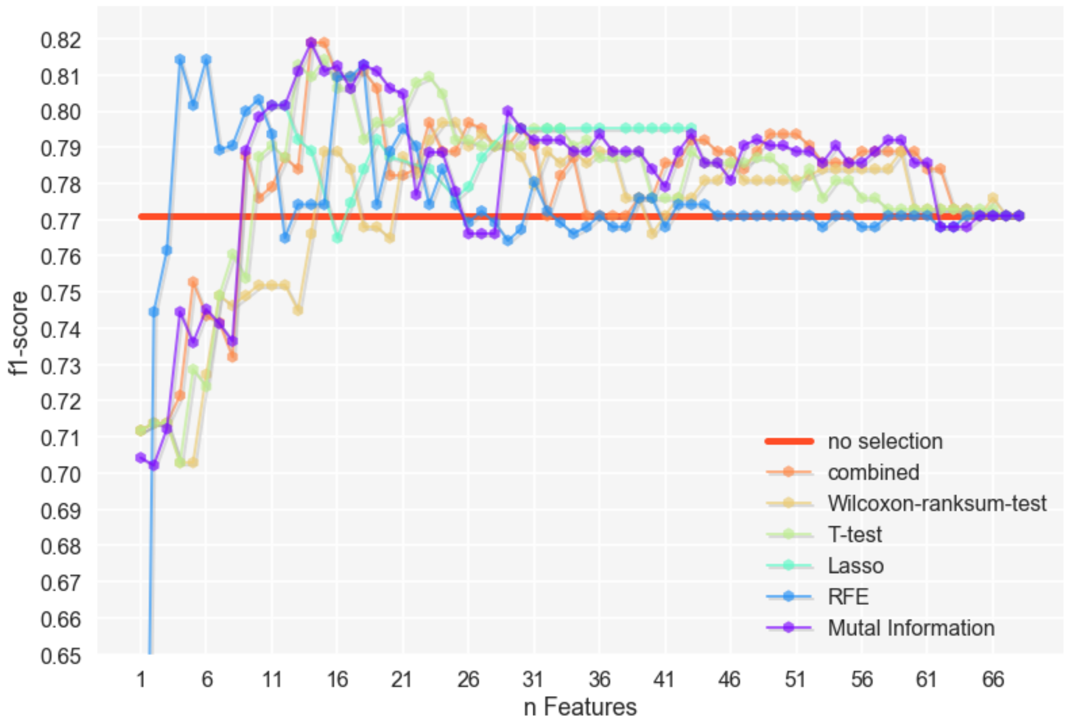


Figure 6.10: Feature subset results for the Martin Dataset using Logistic Regression. The RFE line is the first to surpass the baseline using only four features: The count of blacklisted words (40) and long words (1), the per-word-ratio of Urban-Dictionary-only words (59), and the Negative-Subjectivity-Ratio (24). Strikingly, this small selection scored the second best result for this particular classifier and also keeps up remarkably well with the best scores of the other two classifiers. Also noteworthy: The 19 highest scoring selections use less than 20 features, the 48 top selections all use less than 30 features.

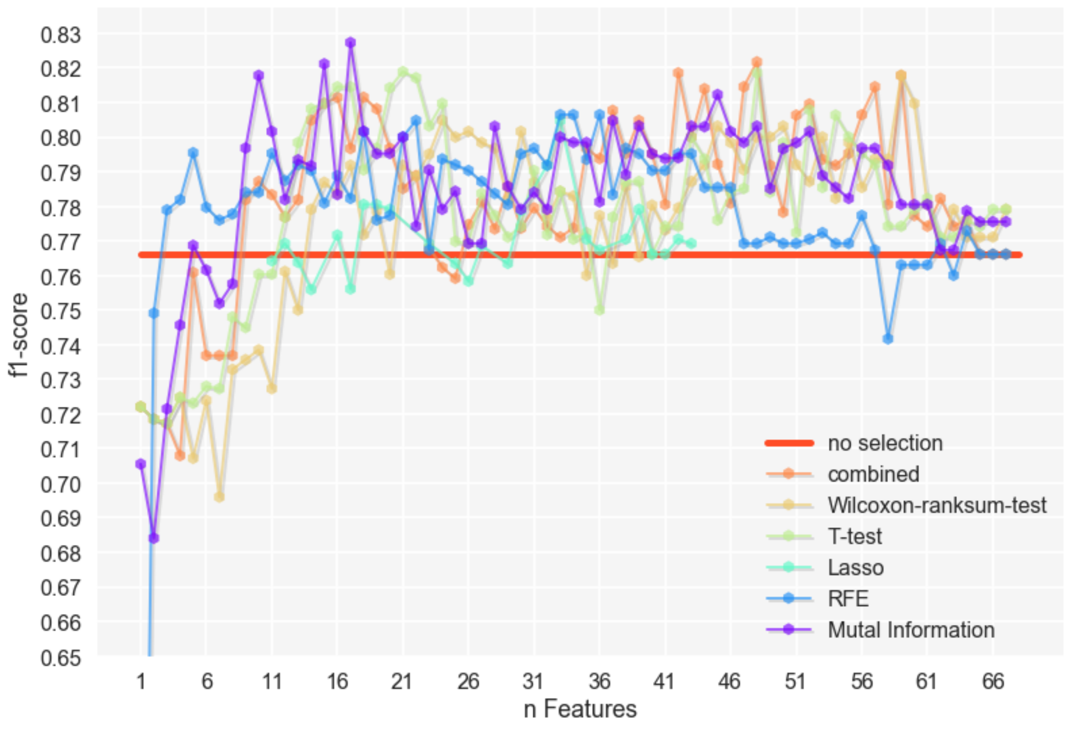


Figure 6.11: Feature subset results for the Martin Dataset using linear SVM. The RFE line is again the first to cross the baseline—here using only three features (blacklisted words (40), long words (1) and the ratio of Urban-Dictionary-only words (59))! Though with a few features more, the erratic Mutual Information markers accomplish the highest first three peaks among the top 10 selections.

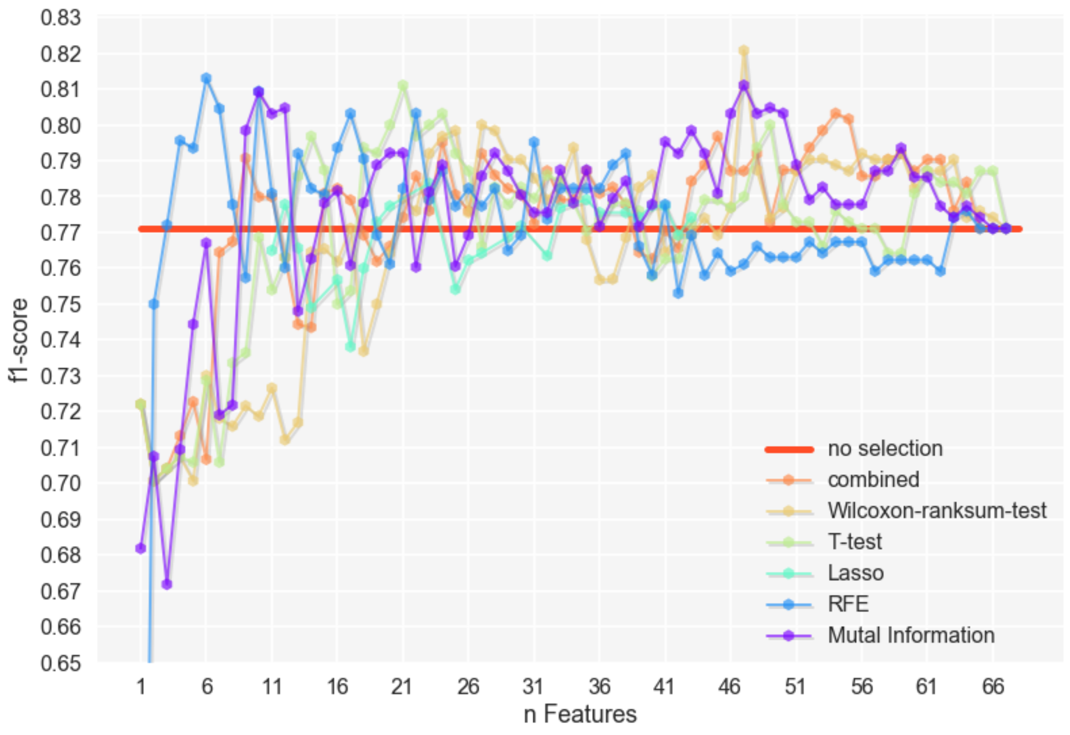


Figure 6.12: Feature subset results for the Martin Dataset using Gaussian SVM. Again, the first peaks beyond the baseline belong to RFE subsets. The second peak is already a major one with only six features in total, pushing the RFE onto the second place overall once more: The count of blacklisted words (40) and long words (1), the per-word-ratio of Urban-Dictionary-only words (59), the Negative-Subjectivity-Ratio (24) and two punctuation features—the ratio of 'sentence endings' (46) and the average length of repeated punctuation (34). The Mutual Information scores for selections between 10 and 12 features perform almost as good.

Chapter 7

Discussion and Conclusion

This thesis explored the impact of multiple feature selection techniques on an aggressive language model with an initial set of 68 features. The features relate to grammar, diversity, basic text statistics, punctuation as well as lexical and stylistic properties.

We applied our feature selection approach to two different datasets: The Wiki Dataset—which is based on a publicly available corpus and amounts to roughly 6 600 samples—and the Martin Dataset, which is based on a corpus that was specifically developed for this research and consists of more than 1k samples in total.

Best Feature Selection Techniques

We employed five different feature selection techniques—the T-test, the Wilcoxon-ranksum-test, Mutual Information, Recursive Feature Elimination and the Lasso. Two patterns that consistently emerged where the RFE's strength in small subsets and the humble results of the ranksum subsets.

Best Models

The three learning algorithms that were tested in the process—Logistic Regression, linear SVM and Gaussian SVM—performed similarly well, though the SVMs fared slightly better in general. The best performing model overall has a precision of 93%, recall of 86.1% and F_1-score of 89.4%; it encompasses 40 features and was trained and tested on the Wiki Dataset, using a Gaussian SVM. The best performing model that was trained and tested on the much smaller Martin Dataset uses a linear SVM, has a precision of 85.1%, recall of 80.5% and F_1-score of 82.7%; it encompasses 17 features.

Best Features

Altogether, the results show that less than a quarter of all the available features are needed to reach or surpass the performance of the full set. Lexical and basic text statistic features yield the biggest impact overall. The three most powerful predictors are the number of blacklisted words, the number (or per-word-ratio) of long words and the ratio of negative subjective words; yet it should be noted that their strength is to be seen with regard to their interaction in the model—in conjunction the features achieve a more balanced precision and recall.

7.1 Common Thread

The two corpora presumably differ noticeably given their unrelated origin, so the question of whether there is a common thread—and therefore a likely general trend in written aggressive language—is inevitable. Indeed, there is a significant overlap of core features. The top features both datasets have in common are

- the count, the per-sentence- and per-word-ratio of blacklisted words (40, 56, 60),
- the count and per-word-ratio of long words (1, 62),
- the total word count (10),
- the count of words that exist in an US-English dictionary (43),
- the ratio of subjective words with negative polarity (24) and
- the per-word-ratio of words written in all capital letters (61).

An advantage of these core features is, that they are relatively simple to compute. Only the extraction of the Negative-Subjectivity-Ratio (24) requires POS tagged words, since the polarity of a word may in fact depend on the POS type (unfortunately, POS tagging is a major bottleneck of the system).

Beside these core features, other common influential predictors in both datasets are: the Exclamation-Mark-Ratio (0), the highlighters count (7), the maximum wordlength (12), the maximum sentence length (18), the average sentence length (19), the per-word-ratio of words that are part of the Urban Dictionary (but not a regular US-English dictionary), the ratio of non-alphanumeric characters (35), the grammatical diversity (26), and the Past-Verbs-Ratio (33).

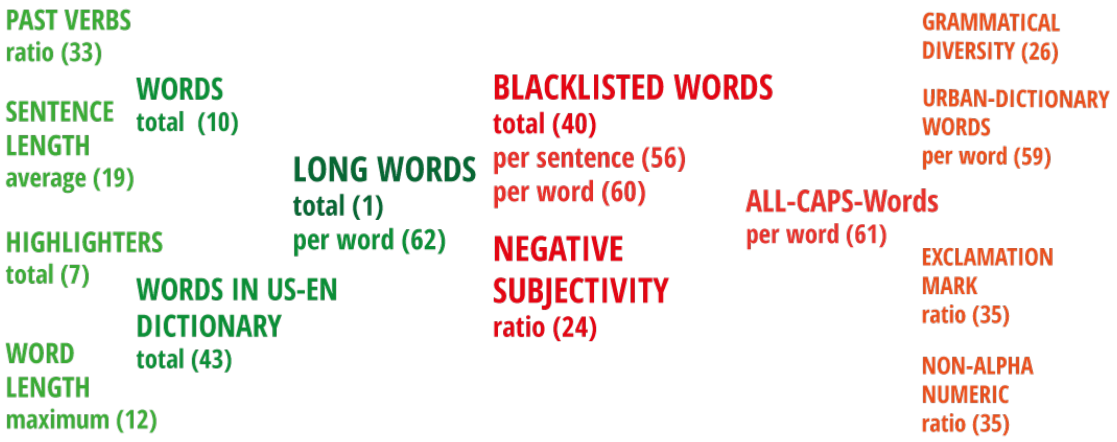


Figure 7.1: Common influential predictors of both datasets. The most informative features are located in the center. Green means that a high value of the feature is linked with non-aggressive comments, red features on the other hand indicate aggression if the feature's value is high.

7.2 Implications

New insights and deeper understanding are a main benefit of data analysis. Here is what we can learn from the data:

In order to compose a perfectly aggressive comment, the distributions of the most informative features suggest a liberal use of blacklisted words—at least one offensive term per every other sentence, preferably more. Fewer words in total are better suited in communicating the emotion. Sentences may, on average, amount to up to 10 words and contain plenty of negative, subjective words at best. Positive subjective words are to be used sparingly if at all; the same applies to words that consist of more than 7 letters—avoid! Writing about past events may convey consideration, thus it is advisable to stick to the present. There is little use for *elitist* punctuation style: Quotation marks, parenthesis, commas, colons or dashes might accidentally soften the tone. Ideally, all punctuation is reduced to exclamation marks. Lastly, a consistently activated caps lock key goes a long way.

7.3 Outlook

The analysis of the two different datasets using new ideas (e.g., count of long words, sub-categorization of punctuation, Urban Dictionary integration) has provided relevant insights, though there remains plenty of room for future exploration.

In terms of the underlying data, using samples with more subtle aggression (e.g., absence of blacklisted words) would be a consequential next step.

The surprisingly predictive long words feature offers itself for further research: Experiments with varying length-thresholds (the current threshold of seven letters is an intuitive choice that has not been systematically optimized), analysis of type or semantic information captured specifically from long words, limitation of the long words feature to proper words (according to a regular dictionary).

Finally, new feature ideas suited for user comments that directly relate to a specific article: The words of the article may serve as an encapsulated dictionary on its own. The article-dictionary could prevent named entities[1] from being misinterpreted as unknown words. It could also be utilized to measure if a comment bears an obvious reference to the article.

[1]The term *Named Entity* refers to specific people, locations, products, etc.

Appendix A

Feature Vectors, Additional Results

A.1 Feature Vector Samples

This section demonstrates what a feature vector looks like. Note, however, that the representation of the subsequent feature vectors is a transposed view of the actual datasets (in order to fit the content onto the page): The rows represent the independent features and the columns are the feature vectors where each belongs to its own sample comment; in general, we look at the columns as the independent features and the rows as the samples.

A.2 Feature Rankings and Selections

The two tables per dataset show the complete feature rankings for the univariate methods, their combined ranking and Recursive Feature Elimination.

A.3 Additional Results per Classifier

The following tables offer more details on the performance per dataset and classifier. Each table contains the 50 best results, ranked by the F_1-score.

A.4 Feature Indices and Names

Table A.13 pairs feature indices with short, descriptive names. It serves as a helpful look up table when manually analyzing the results.

Table A.1: Wikipedia Talk Corpus, feature vectors based on sample comments (tab. 4.2).

rev_id feature	**103624**	**297866**	**93890**	**360594**	**24464221**	**24777589**	**24509777**	**10144987**
0	0.0	0.0	0.0	0.333	0.6	0.25	0.0	0.0
1	10.0	4.0	1.0	9.0	1.0	10.0	1.0	9.0
2	0.0	0.0	0.0	2.0	4.0	14.0	0.0	8.0
3	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
4	0.8	1.0	1.0	0.667	0.2	0.333	1.0	0.833
5	0.2	0.0	0.0	0.0	0.2	0.417	0.0	0.167
6	0.17	0.174	0.114	0.189	0.178	0.177	0.156	0.194
7	0.0	1.0	0.0	2.0	0.0	4.0	0.0	0.0
8	4.0	4.0	0.0	4.0	0.0	2.0	2.0	2.0
9	6.0	4.0	1.0	4.0	4.0	6.0	1.0	6.0
10	65.0	17.0	5.0	62.0	25.0	86.0	6.0	136.0
11	0.723	1.0	1.0	0.774	0.92	0.756	1.0	0.654
12	12.0	12.0	14.0	12.0	11.0	10.0	13.0	11.0
13	4.892	5.529	6.0	4.41	4.16	4.198	6.0	4.037
14	5.0	4.0	4.0	4.0	4.0	4.0	5.5	4.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0
17	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
18	30.0	8.0	5.0	41.0	10.0	25.0	6.0	34.0
19	10.833	4.25	5.0	15.5	6.25	14.333	6.0	22.667
20	5.5	4.0	5.0	10.0	7.0	11.5	6.0	22.5
21	4.0	1.0	5.0	1.0	1.0	4.0	6.0	10.0
22	0.123	0.118	0.4	0.167	0.16	0.195	0.333	0.199
23	0.023	0.029	0.2	0.033	0.04	0.037	0.0	0.037
24	0.031	0.0	0.0	0.025	0.06	0.091	0.25	0.044
25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26	0.277	0.647	1.0	0.306	0.64	0.326	1.167	0.154
27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
28	0.125	0.0	0.0	0.0	0.0	0.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31	0.0	0.0	0.5	0.154	0.143	0.071	0.0	0.139
32	0.727	1.0	0.5	0.846	0.857	0.643	0.0	0.556
33	0.273	0.0	0.0	0.0	0.0	0.286	0.0	0.306
34	0.0	0.0	0.0	0.0	3.0	2.0	0.0	0.0
35	0.019	0.06	0.032	0.042	0.063	0.085	0.053	0.032
36	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
37	0.0	0.0	0.0	0.0	0.0	2.0	0.0	1.0
38	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
39	3.0	1.0	0.0	3.0	2.0	1.0	1.0	2.0
40	0.0	0.0	0.0	0.0	1.0	2.0	0.0	4.0
41	5.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0
42	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
43	60.0	16.0	5.0	60.0	24.0	82.0	6.0	134.0
44	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
45	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
46	0.833	0.75	1.0	0.75	1.25	2.0	1.0	1.0
47	0.0	0.0	0.0	0.0	0.0	0.167	0.0	0.0
48	0.0	0.0	1.0	0.5	0.25	0.167	0.0	0.833
49	0.0	0.0	0.0	0.0	0.25	0.0	0.0	0.0
50	0.333	0.0	0.0	0.5	0.25	0.5	0.0	0.5
51	0.167	0.5	0.0	1.75	0.25	1.833	1.0	1.833
52	0.0	0.0	0.0	0.5	1.0	2.333	0.0	1.333
53	0.0	0.0	0.0	0.0	0.25	0.5	0.0	0.0
54	0.667	1.0	0.0	1.0	0.0	0.333	2.0	0.333
55	0.5	0.25	0.0	0.75	0.5	0.167	1.0	0.333
56	0.0	0.0	0.0	0.0	0.25	0.333	0.0	0.667
57	0.833	0.25	0.0	0.25	0.25	0.0	0.0	0.0
58	0.077	0.059	0.0	0.016	0.04	0.0	0.0	0.0
59	0.046	0.059	0.0	0.048	0.08	0.012	0.167	0.015
60	0.0	0.0	0.0	0.0	0.04	0.023	0.0	0.029
61	0.0	0.059	0.0	0.032	0.04	0.0	0.0	0.015
62	0.154	0.235	0.2	0.145	0.04	0.116	0.167	0.066
63	0.0	0.059	0.0	0.0	0.0	0.0	0.0	0.0
64	0.0	0.0	0.0	0.016	0.0	0.0	0.0	0.0
65	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
66	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
67	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table A.2: Martin Corpus, feature vectors based on sample comments (table 4.3).

feature \ *.txt	**0___113**	**0___116**	**0___140**	**0___406**	**1___067**	**1___191**	**1___644.txt**	**1___694.txt**
0	0.0	0.0	0.0	0.0	0.333	0.0	0.0	0.0
1	0.0	1.0	15.0	3.0	1.0	6.0	4.0	2.0
2	0.0	0.0	2.0	0.0	0.0	4.0	0.0	1.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	1.0	1.0	0.833	1.0	0.667	1.0	0.0	1.0
5	0.0	0.0	0.167	0.0	0.0	0.0	0.0	0.0
6	0.22	0.156	0.164	0.179	0.2	0.173	0.179	0.198
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	5.0	2.0	6.0	2.0	2.0	3.0	1.0	3.0
10	47.0	6.0	77.0	47.0	26.0	73.0	29.0	26.0
11	0.766	1.0	0.74	0.723	0.923	0.753	0.897	0.808
12	7.0	8.0	14.0	12.0	8.0	10.0	10.0	9.0
13	3.34	4.0	4.792	4.0	3.731	4.384	4.345	3.769
14	3.0	2.5	5.0	3.0	4.0	4.0	4.0	3.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	15.0	4.0	20.0	43.0	21.0	37.0	29.0	9.0
19	9.4	3.0	12.833	23.5	13.0	24.333	29.0	8.667
20	9.0	3.0	16.0	23.5	13.0	19.0	29.0	9.0
21	2.0	2.0	1.0	4.0	5.0	17.0	29.0	8.0
22	0.277	0.4	0.186	0.091	0.269	0.271	0.179	0.154
23	0.106	0.0	0.05	0.034	0.115	0.043	0.071	0.038
24	0.043	0.3	0.029	0.023	0.0	0.057	0.054	0.019
25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26	0.34	1.167	0.312	0.34	0.577	0.288	0.483	0.5
27	0.0	0.0	0.0	0.25	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31	0.214	0.5	0.0	0.417	0.0	0.167	0.286	0.0
32	0.714	0.5	0.545	0.5	0.667	0.611	0.714	0.8
33	0.071	0.0	0.455	0.083	0.333	0.222	0.0	0.2
34	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0
35	0.037	0.111	0.037	0.046	0.04	0.027	0.053	0.03
36	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
37	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
38	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
39	0.0	1.0	7.0	0.0	1.0	2.0	5.0	0.0
40	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0
41	0.0	0.0	1.0	0.0	0.0	0.0	0.0	2.0
42	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
43	47.0	6.0	75.0	44.0	26.0	70.0	28.0	24.0
44	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
45	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
46	1.0	1.5	1.0	1.0	1.5	1.0	0.0	1.0
47	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
48	0.6	0.5	0.0	2.5	0.0	1.0	2.0	0.0
49	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50	1.0	0.0	1.0	4.0	0.5	1.667	3.0	0.0
51	0.2	0.0	1.167	2.0	0.5	1.0	6.0	0.0
52	0.0	0.0	0.333	0.0	0.0	1.333	0.0	0.333
53	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0
54	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
55	0.0	0.5	1.167	0.0	0.5	0.667	5.0	0.0
56	0.0	0.0	0.0	0.0	0.5	0.0	1.0	0.0
57	0.0	0.0	0.167	0.0	0.0	0.0	0.0	0.667
58	0.0	0.0	0.013	0.0	0.0	0.0	0.0	0.077
59	0.0	0.167	0.091	0.0	0.038	0.027	0.172	0.0
60	0.0	0.0	0.0	0.0	0.038	0.0	0.034	0.0
61	0.0	0.167	0.013	0.0	0.0	0.0	0.0	1.0
62	0.0	0.167	0.195	0.064	0.038	0.082	0.138	0.077
63	0.0	0.0	0.0	0.0	0.038	0.0	0.0	0.0
64	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
65	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
66	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
67	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table A.3: Wiki Dataset, feature rankings.

rank	**t-test**	**ranksum**	**MI**	**combined**	**RFE**
1	60	60	56	60	60
2	40	56	60	56	40
3	56	40	40	40	24
4	1	1	62	1	56
5	62	12	24	62	7
6	24	51	51	51	51
7	4	62	61	24	44
8	7	13	1	7	45
9	61	18	13	12	62
10	51	19	7	18	61
11	10	7	19	19	64
12	0	33	36	4	23
13	18	10	12	13	35
14	43	20	18	43	16
15	33	32	32	10	47
16	6	43	20	32	1
17	12	6	4	33	9
18	41	24	43	6	38
19	19	41	6	20	32
20	36	4	59	61	3
21	32	23	58	36	5
22	20	57	0	0	4
23	44	54	23	23	8
24	23	48	35	41	49
25	53	14	33	44	67
26	48	31	44	54	37
27	54	26	10	57	20
28	49	0	14	48	26
29	13	58	26	26	36
30	57	36	37	35	10
31	64	44	54	31	54
32	35	9	46	14	39
33	26	11	55	58	2
34	9	64	31	11	19
35	37	39	57	64	28
36	11	49	41	59	29
37	31	35	11	37	17
38	67	50	22	53	12
39	45	2	48	9	53
40	59	45	45	45	33
41	29	34	64	49	43
42	2	61	53	39	52
43	66	53	52	50	59
44	39	37	67	67	55
45	14	52	50	29	41
46	28	8	29	52	50
47	47	59	39	46	48
48	58	27	9	28	18
49	5	29	8	2	42
50	50	46	65	66	58
51	27	28	28	55	0
52	52	67	66	22	30
53	65	5	17	8	15
54	16	55	16	34	27
55	21	21	21	27	65
56	17	66	38	21	66
57	3	22	42	65	11
58	42	42	30	16	21
59	30	30	3	5	31
60	22	16	34	17	22
61	46	17	63	42	14
62	34	63	49	30	63
63	38	47	15	47	6
64	63	3	25	3	13
65	55	65	27	38	57
66	15	38	47	63	46
67	8	15	5	15	34
68	25	25	2	25	25

Table A.4: Wiki Dataset, 23 Lasso selections ranging from 20 to 46 features.

n	Selected Features																																												
20		1	2			7	8	9				16		20	22		24								36		38	40		44	47				51			56			60	61	62	64		
21		1	2			7	8	9				16		20	22		24							35	36		38	40		44	47				51			56			60	61	62	64		
22		1	2			7	8	9		11		16		20	22		24							35	36		38	40		44	47				51			56			60	61	62	64		
23		1	2			7	8	9		11		16		20	22	23	24							35	36		38	40		44	47				51			56			60	61	62	64		
24		1	2		5	7	8	9		11		16		20	22	23	24							35	36		38	40		44	47				51			56			60	61	62	64		
26		1	2		5	7	8	9		11		16		20	22	23	24				29			35	36		38	40		44	47		49		51			56			60	61	62	64		
27		1	2		5	7	8	9		11		16		20	22	23	24			28	29			35	36		38	40		44	47		49		51			56			60	61	62	64		
28		1	2		5	7	8	9		11		16		20	22	23	24	26		28	29			35	36		38	40		44	47		49		51			56			60	61	62	64		
29		1	2		5	7	8	9		11		16		20	22	23	24	26		28	29			35	36		38	40		44	47		49		51			56			60	61	62	64		67
30		1	2		5	7	8	9		11		16		20	22	23	24	26		28	29	32		35	36		38	40		44	47		49		51			56			60	61	62	64		67
32		1	2	4	5	7	8	9		11		16		20	22	23	24	26		28	29	32	33	35	36		38	40		44	47		49		51			56			60	61	62	64		67
33		1	2	4	5	7	8	9		11		16		20	22	23	24	26		28	29	32	33	35	36	37	38	40		44	47		49		51			56			60	61	62	64		67
34		1	2	4	5	7	8	9		11	12	16		20	22	23	24	26		28	29	32	33	35	36	37	38	40		44	47		49		51			56			60	61	62	64		67
35		1	2	4	5	7	8	9		11	12	16		20	22	23	24	26		28	29	32	33	35	36	37	38	40	43	44	47		49		51			56			60	61	62	64		67
36		1	2	4	5	7	8	9		11	12	16		20	22	23	24	26		28	29	32	33	35	36	37	38	40	43	44	47		49		51			56		59	60	61	62	64		67
37		1	2	4	5	7	8	9		11	12	16	17	20	22	23	24	26		28	29	32	33	35	36	37	38	40	43	44	47		49		51			56		59	60	61	62	64		67
38		1	2	4	5	7	8	9		11	12	16	17	20	22	23	24	26		28	29	32	33	35	36	37	38	40	43	44	47	48	49		51			56		59	60	61	62	64		67
39		1	2	4	5	7	8	9		11	12	16	17	20	22	23	24	26		28	29	32	33	35	36	37	38	40	43	44	47	48	49		51	52		56		59	60	61	62	64		67
41	0	1	2	4	5	7	8	9	10	11	12	16	17	20	22	23	24	26		28	29	32	33	35	36	37	38	40	43	44	47	48	49		51	52		56		59	60	61	62	64		67
42	0	1	2	4	5	7	8	9	10	11	12	16	17	20	22	23	24	26		28	29	32	33	35	36	37	38	40	43	44	47	48	49		51	52		56	58	59	60	61	62	64		67
43	0	1	2	4	5	7	8	9	10	11	12	16	17	20	22	23	24	26	27	28	29	32	33	35	36	37	38	40	43	44	47	48	49		51	52		56	58	59	60	61	62	64		67
44	0	1	2	4	5	7	8	9	10	11	12	16	17	20	22	23	24	26	27	28	29	32	33	35	36	37	38	40	43	44	47	48	49	50	51	52		56	58	59	60	61	62	64		67
46	0	1	2	4	5	7	8	9	10	11	12	16	17	20	22	23	24	26	27	28	29	32	33	35	36	37	38	40	43	44	47	48	49	50	51	52	53	56	58	59	60	61	62	64	66	67

Table A.5: Martin Dataset, feature rankings.

rank	t-test	ranksum	MI	combined	RFE
1	1	1	10	1	40
2	43	43	43	43	1
3	10	10	56	10	59
4	11	11	1	35	24
5	40	26	40	40	46
6	26	59	62	11	34
7	59	35	61	26	54
8	35	19	60	56	42
9	56	12	24	24	61
10	24	18	35	62	56
11	60	20	63	60	58
12	62	24	12	59	11
13	0	62	46	12	62
14	61	60	19	61	38
15	18	40	7	19	60
16	2	56	26	0	52
17	12	2	33	18	47
18	54	23	11	2	8
19	63	52	8	63	23
20	19	9	59	23	5
21	42	61	47	20	4
22	34	55	23	54	45
23	46	0	0	9	14
24	20	21	32	46	35
25	23	51	54	7	7
26	53	50	9	33	39
27	58	33	48	55	43
28	4	32	22	8	37
29	49	48	2	32	64
30	47	13	55	47	28
31	52	7	39	49	33
32	8	54	13	42	53
33	3	4	44	52	44
34	38	49	45	48	41
35	45	63	49	4	0
36	55	42	50	13	3
37	9	16	18	45	12
38	16	45	51	50	9
39	32	8	20	53	21
40	7	6	3	3	15
41	66	31	53	58	10
42	36	53	42	34	27
43	33	34	58	44	30
44	13	47	4	51	22
45	17	3	67	39	2
46	48	5	6	38	55
47	50	39	31	16	26
48	44	44	27	6	67
49	28	28	38	21	13
50	5	46	29	31	63
51	57	27	37	66	51
52	37	37	52	22	29
53	39	58	28	28	50
54	27	66	66	5	20
55	30	29	41	27	16
56	41	38	65	37	17
57	6	17	5	36	57
58	14	36	64	29	18
59	21	22	34	17	36
60	31	14	57	57	65
61	29	30	36	41	19
62	22	57	14	67	49
63	15	41	15	14	32
64	67	64	16	30	31
65	25	25	21	64	66
66	51	15	25	65	48
67	64	67	30	15	6
68	65	65	17	25	25

Table A.6: Martin Dataset, 25 Lasso selections ranging from 11 to 43 features.

n	Selected Features																																										
11	1			5		8		11					24									36		38	39	40	41				46												
12	1			5		8		11					24									36		38	39	40	41				46										62		
13	1			5		8		11		14			24									36		38	39	40	41				46										62		
14	1			5	7	8		11		14			24									36		38	39	40	41				46										62		
16	1			5	7	8		11		14			24									36	37	38	39	40	41				46					54					62		
17	1		4	5	7	8		11		14			24									36	37	38	39	40	41				46					54					62		
18	1		4	5	7	8		11		14			24									36	37	38	39	40	41				46					54		59			62		
19	1		4	5	7	8		11		14			24									36	37	38	39	40	41			45	46					54		59			62		
20	1		4	5	7	8		11		14			24			28						36	37	38	39	40	41			45	46					54		59			62		
23	1		4	5	7	8		11		14		23	24			28						36	37	38	39	40	41			45	46	47				54		59		61	62		
25	1		4	5	7	8		11		14		23	24			28						36	37	38	39	40	41			45	46	47		52		54		59		61	62	64	
26	1		4	5	7	8		11		14		23	24			28					34	36	37	38	39	40	41			45	46	47		52		54		59		61	62	64	
27	1		4	5	7	8		11		14		23	24			28					34	36	37	38	39	40	41	42		45	46	47		52		54		59		61	62	64	
29	1		4	5	7	8	9	11		14		23	24			28				33	34	36	37	38	39	40	41	42		45	46	47		52		54		59		61	62	64	
30	1		4	5	7	8	9	11		14		23	24			28				33	34	36	37	38	39	40	41	42		45	46	47		52		54		59	60	61	62	64	
32	1		4	5	7	8	9	11		14		23	24		27	28		30		33	34	36	37	38	39	40	41	42		45	46	47		52		54		59	60	61	62	64	
33	1		4	5	7	8	9	11		14		23	24		27	28		30		33	34	36	37	38	39	40	41	42	44	45	46	47		52		54		59	60	61	62	64	
35	1		4	5	7	8	9	11		14		23	24	26	27	28		30		33	34	36	37	38	39	40	41	42	44	45	46	47		52		54	58	59	60	61	62	64	
36	1		4	5	7	8	9	11		14		23	24	26	27	28		30		33	34	36	37	38	39	40	41	42	44	45	46	47	51	52		54	58	59	60	61	62	64	
38	1		4	5	7	8	9	11		14	21	23	24	26	27	28	29	30		33	34	36	37	38	39	40	41	42	44	45	46	47	51	52		54	58	59	60	61	62	64	
39	1		4	5	7	8	9	11		14	21	23	24	26	27	28	29	30		33	34	36	37	38	39	40	41	42	44	45	46	47	51	52		54	58	59	60	61	62	64	67
40	1		4	5	7	8	9	11		14	21	23	24	26	27	28	29	30		33	34	36	37	38	39	40	41	42	44	45	46	47	51	52	53	54	58	59	60	61	62	64	67
41	1	2	4	5	7	8	9	11		14	21	23	24	26	27	28	29	30		33	34	36	37	38	39	40	41	42	44	45	46	47	51	52	53	54	58	59	60	61	62	64	67
42	1	2	4	5	7	8	9	11	13	14	21	23	24	26	27	28	29	30		33	34	36	37	38	39	40	41	42	44	45	46	47	51	52	53	54	58	59	60	61	62	64	67
43	1	2	4	5	7	8	9	11	13	14	21	23	24	26	27	28	29	30	31	33	34	36	37	38	39	40	41	42	44	45	46	47	51	52	53	54	58	59	60	61	62	64	67

Table A.7: Wiki Dataset, Logistic Regression, top 50 results sorted by F_1-score.

n	selection type	F_1-score	precision	recall	TN	FN	FP	TP
28	Lasso	0.8809	0.9417	0.8274	778	145	43	695
41	Lasso	0.8798	0.9229	0.8405	762	134	59	706
46	Lasso	0.8798	0.9229	0.8405	762	134	59	706
26	Lasso	0.8797	0.9392	0.8274	776	145	45	695
42	Lasso	0.8791	0.9228	0.8393	762	135	59	705
43	Lasso	0.8791	0.9228	0.8393	762	135	59	705
44	Lasso	0.8791	0.9228	0.8393	762	135	59	705
27	Lasso	0.879	0.9391	0.8262	776	146	45	694
29	Lasso	0.879	0.9391	0.8262	776	146	45	694
34	Lasso	0.8789	0.9239	0.8381	763	136	58	704
33	Lasso	0.8784	0.9227	0.8381	762	136	59	704
32	Lasso	0.8782	0.9238	0.8369	763	137	58	703
35	Lasso	0.878	0.9204	0.8393	760	135	61	705
36	Lasso	0.8773	0.9203	0.8381	760	136	61	704
37	Lasso	0.8773	0.9203	0.8381	760	136	61	704
38	Lasso	0.8773	0.9203	0.8381	760	136	61	704
39	Lasso	0.8773	0.9203	0.8381	760	136	61	704
21	Lasso	0.8751	0.9473	0.8131	783	157	38	683
22	Lasso	0.8751	0.9473	0.8131	783	157	38	683
23	Lasso	0.8747	0.9448	0.8143	781	156	40	684
30	Lasso	0.8742	0.9267	0.8274	766	145	55	695
24	Lasso	0.8737	0.9409	0.8155	778	155	43	685
20	Lasso	0.8728	0.9421	0.8131	779	157	42	683
17	RFE	0.8625	0.9372	0.7988	776	169	45	671
18	RFE	0.8625	0.9372	0.7988	776	169	45	671
11	RFE	0.8607	0.9445	0.7905	782	176	39	664
10	RFE	0.8605	0.9458	0.7893	783	177	38	663
15	RFE	0.8605	0.9407	0.7929	779	174	42	666
19	RFE	0.859	0.9163	0.8083	759	161	62	679
16	RFE	0.8588	0.9318	0.7964	772	171	49	669
11	Mutual Information	0.8586	0.943	0.7881	781	178	40	662
12	Mutual Information	0.8586	0.943	0.7881	781	178	40	662
13	RFE	0.8581	0.9366	0.7917	776	175	45	665
14	RFE	0.8581	0.9366	0.7917	776	175	45	665
20	RFE	0.8579	0.9139	0.8083	757	161	64	679
13	Mutual Information	0.8577	0.9442	0.7857	782	180	39	660
21	RFE	0.8577	0.906	0.8143	750	156	71	684
14	Mutual Information	0.8571	0.9429	0.7857	781	180	40	660
12	RFE	0.8566	0.9415	0.7857	780	180	41	660
10	Mutual Information	0.8558	0.9414	0.7845	780	181	41	659
45	Ranksum	0.8557	0.9045	0.8119	749	158	72	682
22	RFE	0.8555	0.9056	0.8107	750	159	71	681
24	RFE	0.8555	0.9056	0.8107	750	159	71	681
35	T-test	0.8555	0.9101	0.8071	754	162	67	678
39	combined	0.8555	0.9101	0.8071	754	162	67	678
51	Ranksum	0.855	0.9	0.8143	745	156	76	684
49	Ranksum	0.8548	0.9011	0.8131	746	157	75	683

Table A.8: Wiki Dataset, Linear SVM, top 50 results sorted by F_1-score.

n	selection type	F_1-score	precision	recall	TN	FN	FP	TP	C
60	Mutual Information	0.8909	0.9234	0.8607	761	117	60	723	100.0
61	Mutual Information	0.89	0.9254	0.8571	763	120	58	720	100.0
62	Mutual Information	0.889	0.9276	0.8536	765	123	56	717	100.0
43	RFE	0.889	0.9166	0.8631	755	115	66	725	100.0
58	combined	0.8883	0.9511	0.8333	785	140	36	700	100.0
67	Mutual Information	0.8879	0.9196	0.8583	758	119	63	721	100.0
48	T-test	0.8879	0.9252	0.8536	763	123	58	717	100.0
65	Mutual Information	0.8874	0.9296	0.8488	767	127	54	713	100.0
35	combined	0.8871	0.9436	0.8369	779	137	42	703	100.0
39	RFE	0.887	0.9273	0.85	765	126	56	714	100.0
25	RFE	0.8869	0.9388	0.8405	775	134	46	706	100.0
62	T-test	0.8866	0.914	0.8607	753	117	68	723	100.0
22	RFE	0.8865	0.925	0.8512	763	125	58	715	100.0
29	T-test	0.8864	0.9318	0.8452	769	130	52	710	100.0
23	combined	0.8861	0.9459	0.8333	781	140	40	700	100.0
55	combined	0.886	0.9183	0.856	757	121	64	719	100.0
24	RFE	0.886	0.9352	0.8417	772	133	49	707	100.0
40	Mutual Information	0.886	0.9352	0.8417	772	133	49	707	100.0
54	Mutual Information	0.8859	0.9411	0.8369	777	137	44	703	100.0
47	combined	0.8857	0.926	0.8488	764	127	57	713	100.0
37	Lasso	0.8856	0.9215	0.8524	760	124	61	716	100.0
44	Mutual Information	0.8853	0.9584	0.8226	791	149	30	691	100.0
50	combined	0.8853	0.9351	0.8405	772	134	49	706	100.0
25	Mutual Information	0.8852	0.941	0.8357	777	138	44	702	100.0
51	Ranksum	0.8851	0.9363	0.8393	773	135	48	705	100.0
55	Mutual Information	0.8851	0.9363	0.8393	773	135	48	705	100.0
39	Lasso	0.885	0.9203	0.8524	759	124	62	716	100.0
66	RFE	0.8849	0.927	0.8464	765	129	56	711	100.0
56	RFE	0.8844	0.9148	0.856	754	121	67	719	100.0
24	T-test	0.8843	0.9545	0.8238	788	148	33	692	100.0
52	Mutual Information	0.8843	0.9315	0.8417	769	133	52	707	100.0
12	RFE	0.8842	0.9213	0.85	760	126	61	714	1000.0
60	Ranksum	0.8841	0.9169	0.8536	756	123	65	717	100.0
25	combined	0.8841	0.9507	0.8262	785	146	36	694	100.0
33	combined	0.884	0.9635	0.8167	795	154	26	686	100.0
44	combined	0.884	0.9397	0.8345	776	139	45	701	100.0
26	T-test	0.884	0.9457	0.8298	781	143	40	697	100.0
42	Lasso	0.8839	0.9235	0.8476	762	128	59	712	100.0
59	Mutual Information	0.8839	0.9349	0.8381	772	136	49	704	100.0
42	T-test	0.8838	0.9409	0.8333	777	140	44	700	100.0
66	Mutual Information	0.8837	0.942	0.8321	778	141	43	699	100.0
47	T-test	0.8835	0.9432	0.831	779	142	42	698	10.0
57	Mutual Information	0.8835	0.9432	0.831	779	142	42	698	10.0
32	RFE	0.8835	0.9325	0.8393	770	135	51	705	100.0
44	T-test	0.8835	0.9325	0.8393	770	135	51	705	100.0
56	Mutual Information	0.8834	0.9506	0.825	785	147	36	693	100.0
41	T-test	0.8833	0.9337	0.8381	771	136	50	704	100.0
26	Mutual Information	0.8832	0.9582	0.819	791	152	30	688	100.0
26	RFE	0.8832	0.9348	0.8369	772	137	49	703	10.0
62	combined	0.8831	0.9135	0.8548	753	122	68	718	100.0

Table A.9: Wiki Dataset, Gaussian SVM, top 50 results sorted by F_1-score.

n	**selection type**	**F_1-score**	**precision**	**recall**	**TN**	**FN**	**FP**	**TP**	`gamma`	`C`
40	combined	0.8942	0.9305	0.8607	767	117	54	723	0.1	1000.0
35	T-test	0.8936	0.9304	0.8595	767	118	54	722	0.1	1000.0
45	combined	0.8936	0.9304	0.8595	767	118	54	722	0.1	1000.0
57	Mutual Information	0.8932	0.9455	0.8464	780	129	41	711	0.1	100.0
23	combined	0.8932	0.9338	0.856	770	121	51	719	1.0	100.0
34	T-test	0.8932	0.9338	0.856	770	121	51	719	0.1	1000.0
43	Mutual Information	0.893	0.9237	0.8643	761	114	60	726	0.1	1000.0
39	combined	0.8926	0.9269	0.8607	764	117	57	723	0.1	1000.0
43	combined	0.8926	0.9269	0.8607	764	117	57	723	0.1	1000.0
27	Mutual Information	0.8925	0.9395	0.85	775	126	46	714	0.1	1000.0
41	combined	0.8925	0.928	0.8595	765	118	56	722	0.1	1000.0
43	Ranksum	0.8925	0.928	0.8595	765	118	56	722	0.1	1000.0
36	T-test	0.8919	0.9325	0.8548	769	122	52	718	0.1	1000.0
37	T-test	0.8919	0.9325	0.8548	769	122	52	718	0.1	1000.0
42	combined	0.8919	0.9268	0.8595	764	118	57	722	0.1	1000.0
44	Ranksum	0.8919	0.9268	0.8595	764	118	57	722	0.1	1000.0
28	Mutual Information	0.8918	0.9394	0.8488	775	127	46	713	0.1	1000.0
58	combined	0.8918	0.9394	0.8488	775	127	46	713	0.1	100.0
41	T-test	0.8918	0.9279	0.8583	765	119	56	721	0.1	1000.0
55	combined	0.8917	0.9406	0.8476	776	128	45	712	0.1	100.0
29	T-test	0.8915	0.9359	0.8512	772	125	49	715	0.1	1000.0
46	Ranksum	0.8915	0.9246	0.8607	762	117	59	723	0.1	1000.0
57	combined	0.8914	0.9429	0.8452	778	130	43	710	0.1	100.0
61	Mutual Information	0.8914	0.9429	0.8452	778	130	43	710	0.1	100.0
30	Mutual Information	0.8914	0.937	0.85	773	126	48	714	0.1	1000.0
41	Mutual Information	0.8914	0.9256	0.8595	763	118	58	722	0.1	1000.0
44	Mutual Information	0.8914	0.9256	0.8595	763	118	58	722	0.1	1000.0
50	Mutual Information	0.8914	0.9256	0.8595	763	118	58	722	0.1	1000.0
27	combined	0.8913	0.9441	0.844	779	131	42	709	0.1	1000.0
53	Mutual Information	0.8913	0.9441	0.844	779	131	42	709	0.1	100.0
29	Mutual Information	0.8912	0.9382	0.8488	774	127	47	713	0.1	1000.0
56	combined	0.8911	0.9393	0.8476	775	128	46	712	0.1	100.0
55	Mutual Information	0.8908	0.9416	0.8452	777	130	44	710	0.1	100.0
58	Mutual Information	0.8908	0.9416	0.8452	777	130	44	710	0.1	100.0
36	Ranksum	0.8908	0.9245	0.8595	762	118	59	722	0.1	1000.0
40	Mutual Information	0.8907	0.9255	0.8583	763	119	58	721	0.1	1000.0
19	RFE	0.8906	0.9501	0.8381	784	136	37	704	0.1	1000.0
65	Mutual Information	0.8906	0.9381	0.8476	774	128	47	712	0.1	100.0
33	RFE	0.8905	0.9323	0.8524	769	124	52	716	0.1	1000.0
66	Mutual Information	0.8904	0.9392	0.8464	775	129	46	711	0.1	100.0
30	T-test	0.8904	0.9334	0.8512	770	125	51	715	0.1	1000.0
34	RFE	0.8904	0.9334	0.8512	770	125	51	715	0.1	1000.0
42	Ranksum	0.8904	0.9277	0.856	765	121	56	719	0.1	1000.0
35	Mutual Information	0.8903	0.9288	0.8548	766	122	55	718	0.1	1000.0
48	Mutual Information	0.8901	0.9416	0.844	777	131	44	709	0.1	100.0
45	Ranksum	0.8901	0.9244	0.8583	762	119	59	721	0.1	1000.0
48	combined	0.8901	0.9244	0.8583	762	119	59	721	0.1	1000.0

Table A.10: Martin Dataset, Logistic Regression, top 50 results sorted by F_1-score.

n	selection type	F_1-score	precision	recall	TN	FN	FP	TP
14	Mutual Information	0.8189	0.8254	0.8125	108	24	22	104
14	combined	0.8189	0.8254	0.8125	108	24	22	104
15	combined	0.8189	0.8254	0.8125	108	24	22	104
4	RFE	0.8142	0.824	0.8047	108	25	22	103
6	RFE	0.8142	0.824	0.8047	108	25	22	103
15	T-test	0.8142	0.824	0.8047	108	25	22	103
13	T-test	0.8127	0.8293	0.7969	109	26	21	102
18	Mutual Information	0.8127	0.8293	0.7969	109	26	21	102
18	RFE	0.8127	0.8293	0.7969	109	26	21	102
16	Mutual Information	0.8125	0.8125	0.8125	106	24	24	104
13	Mutual Information	0.811	0.8175	0.8047	107	25	23	103
15	Mutual Information	0.811	0.8175	0.8047	107	25	23	103
18	combined	0.811	0.8175	0.8047	107	25	23	103
19	Mutual Information	0.811	0.8175	0.8047	107	25	23	103
14	T-test	0.8095	0.8226	0.7969	108	26	22	102
16	combined	0.8095	0.8226	0.7969	108	26	22	102
16	RFE	0.8095	0.8226	0.7969	108	26	22	102
17	combined	0.8095	0.8226	0.7969	108	26	22	102
17	RFE	0.8095	0.8226	0.7969	108	26	22	102
23	T-test	0.8095	0.8226	0.7969	108	26	22	102
22	T-test	0.8078	0.811	0.8047	106	25	24	103
16	T-test	0.8063	0.816	0.7969	107	26	23	102
17	T-test	0.8063	0.816	0.7969	107	26	23	102
19	combined	0.8063	0.816	0.7969	107	26	23	102
20	Mutual Information	0.8063	0.816	0.7969	107	26	23	102
17	Mutual Information	0.8062	0.8	0.8125	104	24	26	104
21	Mutual Information	0.8048	0.8211	0.7891	108	27	22	101
24	T-test	0.8048	0.8211	0.7891	108	27	22	101
10	RFE	0.8031	0.8095	0.7969	106	26	24	102
5	RFE	0.8016	0.8145	0.7891	107	27	23	101
11	Mutual Information	0.8016	0.8145	0.7891	107	27	23	101
12	Mutual Information	0.8016	0.8145	0.7891	107	27	23	101
11	Lasso	0.8016	0.7984	0.8047	104	25	26	103
12	Lasso	0.8016	0.7984	0.8047	104	25	26	103
9	RFE	0.8	0.8031	0.7969	105	26	25	102
21	T-test	0.8	0.8031	0.7969	105	26	25	102
29	Mutual Information	0.8	0.8197	0.7813	108	28	22	100
10	Mutual Information	0.7984	0.808	0.7891	106	27	24	101
19	T-test	0.7969	0.7969	0.7969	104	26	26	102
20	T-test	0.7969	0.7969	0.7969	104	26	26	102
23	combined	0.7968	0.813	0.7813	107	28	23	100
24	Ranksum	0.7968	0.813	0.7813	107	28	23	100
25	Ranksum	0.7968	0.813	0.7813	107	28	23	100
26	combined	0.7968	0.813	0.7813	107	28	23	100
35	Lasso	0.7953	0.8016	0.7891	105	27	25	101
36	Lasso	0.7953	0.8016	0.7891	105	27	25	101

Table A.11: Martin Dataset, Linear SVM, top 50 results sorted by F_1-score.

n	selection type	F_1-score	precision	recall	TN	FN	FP	TP	C
17	Mutual Information	0.8273	0.8512	0.8047	112	25	18	103	100.0
48	combined	0.8217	0.8154	0.8281	106	22	24	106	100.0
15	Mutual Information	0.8211	0.8559	0.7891	113	27	17	101	100.0
21	T-test	0.8189	0.8254	0.8125	108	24	22	104	100.0
48	T-test	0.8185	0.8092	0.8281	105	22	25	106	100.0
42	combined	0.8185	0.8092	0.8281	105	22	25	106	100.0
10	Mutual Information	0.8178	0.8487	0.7891	112	27	18	101	100.0
59	combined	0.8178	0.8487	0.7891	112	27	18	101	10.0
59	Ranksum	0.8178	0.8487	0.7891	112	27	18	101	10.0
22	T-test	0.8171	0.814	0.8203	106	23	24	105	100.0
16	T-test	0.8145	0.8417	0.7891	111	27	19	101	10.0
17	T-test	0.8145	0.8417	0.7891	111	27	19	101	10.0
47	combined	0.8145	0.8417	0.7891	111	27	19	101	100.0
57	combined	0.8145	0.8417	0.7891	111	27	19	101	10.0
20	T-test	0.8142	0.824	0.8047	108	25	22	103	10.0
44	combined	0.814	0.8077	0.8203	105	23	25	105	100.0
45	Mutual Information	0.8123	0.797	0.8281	103	22	27	106	100.0
16	combined	0.8115	0.8534	0.7734	113	29	17	99	10.0
18	combined	0.8115	0.8534	0.7734	113	29	17	99	10.0
24	T-test	0.8097	0.8403	0.7813	111	28	19	100	100.0
15	combined	0.8097	0.8403	0.7813	111	28	19	100	10.0
15	T-test	0.8097	0.8403	0.7813	111	28	19	100	10.0
60	Ranksum	0.8097	0.8403	0.7813	111	28	19	100	10.0
52	combined	0.8095	0.8226	0.7969	108	26	22	102	10.0
14	T-test	0.8082	0.8462	0.7734	112	29	18	99	10.0
19	combined	0.8082	0.8462	0.7734	112	29	18	99	10.0
52	T-test	0.8078	0.811	0.8047	106	25	24	103	10.0
37	combined	0.8077	0.7955	0.8203	103	23	27	105	100.0
33	RFE	0.8065	0.8333	0.7813	110	28	20	100	100.0
34	RFE	0.8065	0.8333	0.7813	110	28	20	100	10.0
36	RFE	0.8065	0.8333	0.7813	110	28	20	100	10.0
56	combined	0.8065	0.8333	0.7813	110	28	20	100	10.0
54	T-test	0.8063	0.816	0.7969	107	26	23	102	10.0
51	combined	0.8063	0.816	0.7969	107	26	23	102	100.0
24	Ranksum	0.8049	0.839	0.7734	111	29	19	99	10.0
14	combined	0.8049	0.839	0.7734	111	29	19	99	100.0
22	RFE	0.8048	0.8211	0.7891	108	27	22	101	10.0
33	Lasso	0.8048	0.8211	0.7891	108	27	22	101	100.0
37	Mutual Information	0.8048	0.8211	0.7891	108	27	22	101	100.0
39	combined	0.8048	0.8211	0.7891	108	27	22	101	100.0
23	T-test	0.8032	0.8264	0.7813	109	28	21	100	100.0
39	Mutual Information	0.8032	0.8264	0.7813	109	28	21	100	100.0
43	Mutual Information	0.8031	0.8095	0.7969	106	26	24	102	100.0
45	Ranksum	0.8031	0.8095	0.7969	106	26	24	102	100.0
48	Mutual Information	0.8031	0.8095	0.7969	106	26	24	102	10.0
50	Ranksum	0.8031	0.8095	0.7969	106	26	24	102	10.0
28	Mutual Information	0.8031	0.7939	0.8125	103	24	27	104	100.0
44	Mutual Information	0.803	0.766	0.8438	97	20	33	108	100.0
18	Mutual Information	0.8017	0.8509	0.7578	113	31	17	97	1.0
11	Mutual Information	0.8016	0.8319	0.7734	110	29	20	99	100.0

Table A.12: Martin Dataset, Gaussian SVM, top 50 results sorted by F_1-score.

n	selection type	F_1-score	precision	recall	TN	FN	FP	TP	gamma	C
47	Ranksum	0.8207	0.8374	0.8047	110	25	20	103	0.01	100.0
6	RFE	0.813	0.8475	0.7813	112	28	18	100	1.0	10.0
21	T-test	0.811	0.8175	0.8047	107	25	23	103	0.1	10.0
47	Mutual Information	0.811	0.8175	0.8047	107	25	23	103	0.01	1000.0
10	RFE	0.8093	0.8062	0.8125	105	24	25	104	1.0	10.0
10	Mutual Information	0.8092	0.791	0.8281	102	22	28	106	0.1	1000.0
49	Mutual Information	0.8048	0.8211	0.7891	108	27	22	101	0.01	1000.0
12	Mutual Information	0.8047	0.8047	0.8047	105	25	25	103	0.1	1000.0
7	RFE	0.8045	0.7754	0.8359	99	21	31	107	10.0	1.0
22	RFE	0.8032	0.8264	0.7813	109	28	21	100	0.1	10.0
50	Mutual Information	0.8032	0.8264	0.7813	109	28	21	100	0.01	1000.0
54	combined	0.8032	0.8264	0.7813	109	28	21	100	0.01	1000.0
17	RFE	0.8031	0.8095	0.7969	106	26	24	102	0.01	1000.0
24	T-test	0.8031	0.8095	0.7969	106	26	24	102	0.1	10.0
46	Mutual Information	0.8031	0.8095	0.7969	106	26	24	102	0.01	1000.0
48	Mutual Information	0.8031	0.8095	0.7969	106	26	24	102	0.01	1000.0
11	Mutual Information	0.8031	0.7939	0.8125	103	24	27	104	0.1	1000.0
55	combined	0.8016	0.8319	0.7734	110	29	20	99	0.01	1000.0
20	T-test	0.8	0.7879	0.8125	102	24	28	104	1.0	1.0
23	T-test	0.8	0.7879	0.8125	102	24	28	104	1.0	1.0
27	Ranksum	0.8	0.8197	0.7813	108	28	22	100	0.1	100.0
49	T-test	0.8	0.8197	0.7813	108	28	22	100	0.01	1000.0
9	Mutual Information	0.7984	0.7923	0.8047	103	25	27	103	0.1	1000.0
53	combined	0.7984	0.7923	0.8047	103	25	27	103	0.01	100.0
43	Mutual Information	0.7984	0.808	0.7891	106	27	24	101	0.01	1000.0
25	Ranksum	0.7984	0.825	0.7734	109	29	21	99	0.01	1000.0
28	Ranksum	0.7984	0.825	0.7734	109	29	21	99	0.01	1000.0
22	T-test	0.7969	0.782	0.8125	101	24	29	104	1.0	1.0
14	T-test	0.7968	0.813	0.7813	107	28	23	100	0.1	100.0
45	combined	0.7968	0.813	0.7813	107	28	23	100	0.01	1000.0
24	Ranksum	0.7967	0.8305	0.7656	110	30	20	98	0.01	1000.0
4	RFE	0.7956	0.7466	0.8516	93	19	37	109	100.0	0.1
41	Mutual Information	0.7953	0.8016	0.7891	105	27	25	101	0.01	1000.0
24	combined	0.7952	0.8182	0.7734	108	29	22	99	0.01	1000.0
31	RFE	0.7952	0.8182	0.7734	108	29	22	99	0.01	1000.0
16	RFE	0.7937	0.8065	0.7813	106	28	24	100	0.01	1000.0
18	T-test	0.7937	0.8065	0.7813	106	28	24	100	0.1	100.0
34	Ranksum	0.7937	0.8065	0.7813	106	28	24	100	0.01	1000.0
48	T-test	0.7937	0.8065	0.7813	106	28	24	100	0.01	1000.0
52	combined	0.7937	0.8065	0.7813	106	28	24	100	0.01	1000.0
5	RFE	0.7935	0.8235	0.7656	109	30	21	98	0.1	1000.0
59	Mutual Information	0.7935	0.8235	0.7656	109	30	21	98	0.1	10.0
19	T-test	0.7922	0.7953	0.7891	104	27	26	101	0.1	100.0
20	Mutual Information	0.7922	0.7953	0.7891	104	27	26	101	0.01	1000.0
21	Mutual Information	0.7922	0.7953	0.7891	104	27	26	101	0.01	1000.0
28	Mutual Information	0.7922	0.7953	0.7891	104	27	26	101	0.1	100.0
13	RFE	0.792	0.8115	0.7734	107	29	23	99	0.01	1000.0

Table A.13: Feature indices and short names.

index	feature name	index	feature name
0	exclamation_ratio	40	blacklisted_words_count
1	long_words_count	41	unkown_word_count
2	wh_pronouns	42	modified_words_count
3	ellipsis_count	43	in_dictionary
4	period_ratio	44	polite_words_count
5	questionmark_ratio	45	imperative_count
6	whitespace_ratio	46	per_sent___sent_endings
7	highlighters_count	47	per_sent___ellipses
8	paragraph_count	48	per_sent___modal_verbs
9	sentence_count	49	per_sent___imperatives
10	word_count	50	per_sent___one_char_word
11	type_to_token_ratio	51	per_sent___connectors
12	max_wordlength	52	per_sent___wh_pronouns
13	average_wordlength	53	per_sent___rep_punc
14	median_wordlength	54	per_sent___paragraphs
15	spaced_words_count	55	per_sent___urbanDict_only
16	edit_distance	56	per_sent___blacklisted_words
17	lengthening_counts	57	per_sent___unknown_words
18	longest_sent_len	58	per_words___unknown_words
19	avg_sent_len	59	per_words___urbanDict_only
20	median_sent_len	60	per_words___blacklisted_words
21	shortest_sent_len	61	per_words___all_caps
22	subj_ratio	62	per_words___long_words
23	subj_pos_ratio	63	per_words___modified_words
24	subj_neg_ratio	64	per_words___numbers
25	mdash_count	65	per_imperatives___imperative_indirect
26	pos_type_ratio	66	per_imperatives___imperative_strength
27	JJR_ratio	67	per_imperatives___imperative_polite
28	JJS_ratio		
29	RBR_ratio		
30	RBS_ratio		
31	modal_ratio		
32	present_ratio		
33	past_ratio		
34	rep_punc_len_avg		
35	non_alphanum_ratio		
36	blacklist_CAPS		
37	mixed_case_word_count		
38	noise_count		
39	urbanDict_only		

References

Literature

[1] Ashley Anderson et al. “The “Nasty Effect:” Online Incivility and Risk Perceptions of Emerging Technologies”. *Journal of Computer-Mediated Communication* 19.3 (2014), pp. 373–387 (cit. on p. 1).

[2] Ingrid Brodnig. *Hass im Netz. Was wir gegen Hetze, Mobbing und Lügen tun können.* Wien: Christian Brandstätter Verlag, 2016 (cit. on pp. 1, 2).

[3] Samuel Brody and Nicholas Diakopoulos. “Cooooooooooooooollllllllllllll!!!!!!!!!!!!!! Using Word Lengthening to Detect Sentiment in Microblogs”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* Edinburgh, UK: Association for Computational Linguistics, 2011, pp. 562–570 (cit. on p. 25).

[4] Pete Burnap and Matthew Williams. “Hate speech, machine classification and statistical modelling of information flows on Twitter: Interpretation and communication for policy decision making”. In: *Proceedings of Internet, Policy & Politics 2014.* Oxford, UK, 2014, pp. 1–18. URL: http://orca.cf.ac.uk/65227/ (cit. on pp. 7, 8, 10).

[5] Ying Chen et al. “Detecting Offensive Language in Social Media to Protect Adolescent Online Safety”. In: *Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust.* SOCIALCOM-PASSAT ’12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 71–80 (cit. on pp. 7, 8, 10).

[6] Isabelle Guyon and André Elisseeff. “An Introduction to Variable and Feature Selection”. *Journal of Machine Learning* 3 (Mar. 2003), pp. 1157–1182 (cit. on pp. 26, 27, 45).

[7] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. *A Practical Guide to Support Vector Classification.* Tech. rep. Taipei 106, Taiwan: Department of Computer Science, National Taiwan University, 2003. URL: http://www.csie.ntu.edu.tw/~cjlin/papers.html (cit. on p. 29).

[8] Till Krause and Hannes Grassegger. “Inside Facebook”. *Süddeutsche Zeitung* (Dec. 2016). URL: http://www.sueddeutsche.de/digital/exklusive-sz-magazin-recherche-inside-facebook-1.3297138 (cit. on p. 2).

[9] Chikashi Nobata et al. “Abusive Language Detection in Online User Content”. In: *Proceedings of the 25th International Conference on World Wide Web*. WWW ’16. Montreal, Quebec, Canada: International World Wide Web Conferences Steering Committee, 2016, pp. 145–153 (cit. on pp. 7–10).

[10] Amir H. Razavi et al. “Offensive Language Detection Using Multi-level Classification”. In: *Proceedings of the 23rd Canadian Conference on Advances in Artificial Intelligence*. AI’10. Ottawa, Canada: Springer-Verlag, 2010, pp. 16–27 (cit. on pp. 7, 8, 10).

[11] Ellen Spertus. “Smokey: Automatic Recognition of Hostile Messages”. In: *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*. AAAI’97/IAAI’97. Providence, Rhode Island: AAAI Press, 1997, pp. 1058–1065 (cit. on pp. 8, 10, 11).

[12] William Warner and Julia Hirschberg. “Detecting Hate Speech on the World Wide Web”. In: *Proceedings of the Second Workshop on Language in Social Media*. LSM ’12. Montreal, Canada: Association for Computational Linguistics, 2012, pp. 19–26 (cit. on pp. 7, 8, 10).

[13] Ellery Wulczyn, Nithum Thain, and Lucas Dixon. “Ex Machina: Personal Attacks Seen at Scale”. In: *Proceedings of the 26th International Conference on World Wide Web*. WWW ’17. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017, pp. 1391–1399 (cit. on pp. 7, 9–11).

Online sources

[14] Jamie Bartlett et al. *Anti-Social Media*. 2014. URL: https://www.demos.co.uk/files/DEMOS_Anti-social_Media.pdf (cit. on p. 11).

[15] Katie Benner. *Twitter Adds New Ways to Curb Abuse and Hate Speech*. Nov. 2016. URL: https://www.nytimes.com/2016/11/16/technology/twitter-adds-new-ways-to-curb-abuse-and-hate-speech.html (cit. on p. 2).

[16] Maeve Duggan. *Online Harassment 2017*. July 2017. URL: http://www.pewinternet.org/2017/07/11/online-harassment-2017/ (cit. on pp. 1, 2).

[17] Melissa Eddy and Mark Scott. *Delete Hate Speech or Pay Up, Germany Tells Social Media Companies*. June 2017. URL: https://www.nytimes.com/2017/06/30/business/germany-facebook-google-twitter.html (cit. on p. 2).

[18] Andy Greenberg. *Now Anyone Can Deploy Google’s Troll-Fighting AI*. 2017. URL: https://www.wired.com/2017/02/googles-troll-fighting-ai-now-belongs-world/ (cit. on p. 2).

[19] Boris Hänßler. *Am Anfang war das Pöbeln – Geschichte des Shitstorms*. Jan. 2014. URL: http://www.spiegel.de/einestages/vom-flamewar-zum-shitstorm-geschichte-des-internet-gepoebels-a-953266.html (cit. on p. 7).

[20] Ed Ho. *Our Latest Update on Safety*. 2017. URL: https://blog.twitter.com/official/en_us/topics/product/2017/our-latest-update-on-safety.html (cit. on p. 2).

[21] Rashmi Jain. *SVM and Parameter Tuning in Python and R*. URL: http://blog.hackerearth.com/simple-tutorial-svm-parameter-tuning-python-r (cit. on p. 30).

[22] Ankur Lal. *Text Moderation – A Use Case on Detecting Offensive Text Using NLP*. Nov. 2015. URL: http://www.hcltech.com/blogs/engineering-and-rd-services/text-moderation-use-case-detecting-offensive-text-using-nlp (cit. on pp. 10, 19).

[23] OpenCV. *Introduction to Support Vector Machines*. URL: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html (cit. on p. 30).

[24] *RBF SVM parameters*. URL: http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html (cit. on p. 29).

[25] Janko Roettgers. *Instagram Starts Using Artificial Intelligence to Moderate Comments. Is Facebook Up Next?* June 2017. URL: http://variety.com/2017/digital/news/instagram-ai-machine-learning-facebook-filters-1202482031/ (cit. on p. 2).

[26] Gerry Smith and Jeremy Kahn. *Comments Welcome at N.Y. Times, With Troll-Blocking Tech Tool*. 2017. URL: https://www.bloomberg.com/news/articles/2017-06-13/comments-welcome-at-n-y-times-with-troll-blocking-tech-tool (cit. on p. 2).

[27] Nicholas Thompson. *Instagram Unleashes an AI System to Blast Away Nasty Comments*. June 2017. URL: https://www.wired.com/story/instagram-launches-ai-system-to-blast-nasty-comments (cit. on p. 2).

[28] Ellery Wulczyn, Nithum Thain, and Lucas Dixon. *Wikipedia Talk Labels: Personal Attacks*. Feb. 2017. URL: https://figshare.com/articles/Wikipedia_Talk_Labels_Personal_Attacks/4054689 (cit. on pp. 9, 16).

Check Final Print Size

— Check final print size! —

width = 100mm
height = 50mm

— Remove this page after printing! —