

Bachelor Thesis

Identifying inappropriate comments in German-language online newspapers

Abinas Kuganathan, Jan Huber, Joel Hirzel

OST – Eastern Switzerland University of Applied Sciences
Department of Computer Science

Advisor: Prof. Dr. Daniel Patrick Politze
External Co-Examiner: Ramon Schildknecht
Internal Co-Examiner: Prof. Dr. Markus Stolze

June 16, 2022

Acknowledgement

We want to thank:

- Prof. Dr. Daniel Patrick Politze for his advice.
- Manuel Sutter for organizing the data transfer from 20 Minuten.
- Marco Di Bernardo for setting up the contract with 20 Minuten.
- Christian Spielmann for providing and supporting the server.
- Our families for the manual labeling of comments.
- AnneMarie O'Neill for proof-reading this thesis.

We would also like to thank our interview partners:

- Andreas Hobi, Chief of the Community-Team at Ringier
- Adrian Eng, Chief of Staff at Watson.ch
- Reto Stauffacher, Head of the Social Media Team at the Neue Zürcher Zeitung
- Jasmine Brönnimann, Product Manager at Tamedia

Abstract

In online discussions, users are called "trolls" when they provoke others, try to dominate discussions or manipulate the opinions of other users. With the rise of social media, trolling has become a prominent term. The discussion culture suffers from the presence of trolls. But there are also more extreme effects, such as paid propaganda trolls who aim to influence elections. Some trolls are easy to recognize because they obviously spread hate, while other trolls behave more subtly. Therefore, removing troll comments is laborious work.

This paper focuses on German-language user comments in (mostly Swiss) online newspapers. The goal is to develop classification algorithms that can automatically and reliably detect unwanted behavior in comments. In a first step, existing literature and solutions were analyzed and evaluated. In addition, it was examined how various Swiss newspapers deal with trolls. Subsequently, training data was collected and processed. Among other data, a labeled dataset of over 2 million comments was compiled by 20 Minuten. Classifiers were developed with different training data to detect three categories of trolls: hate trolls, off-topic trolls and state-linked propaganda trolls.

Hate trolls are detected with a recall of 84%, precision of 83% and accuracy of 83% using a combination of BERT models. Off-topic trolls are detected with a recall of 78%, precision of 83%, and accuracy of 80% mainly by calculating the cosine similarity from a comment to other comments and the article content. State-linked propaganda trolls are detected with a recall of 92%, precision of 90% and accuracy of 91% for training data from Twitter. For comments from 20 Minuten, a classifier can predict with a recall of 62%, precision of 76% and accuracy of 71% whether comments will be accepted or rejected by the moderation. The results have shown that a metadata-only approach is not feasible for the analysis.

To make the results and the algorithms accessible to lay users, a web application was developed. This proof of concept allows trying the classifiers using custom or existing comments.

These classifiers cannot completely replace the manual moderation process. However, the classifier can be used to support the human moderators. With an adjusted threshold, about 20% of the unwanted comments can be automatically detected with almost no false positives. In future research, it could be investigated how well the developed classifiers perform with different data from other domains. Furthermore, the analysis could be further extended by not only analyzing individual comments but accounts of comment authors (user-based approach compared to post-based approach).

Management Summary

Context

On most Swiss news websites, users can comment on articles. Usually, there are commenting guidelines and the comments go through a moderation-process before publication. Inappropriate comments are often referred to as "troll" comments. However, the definition of "online trolling" is ambiguous. Trolling can range from harmless jokes to bullying or state-sponsored propaganda.

Approach

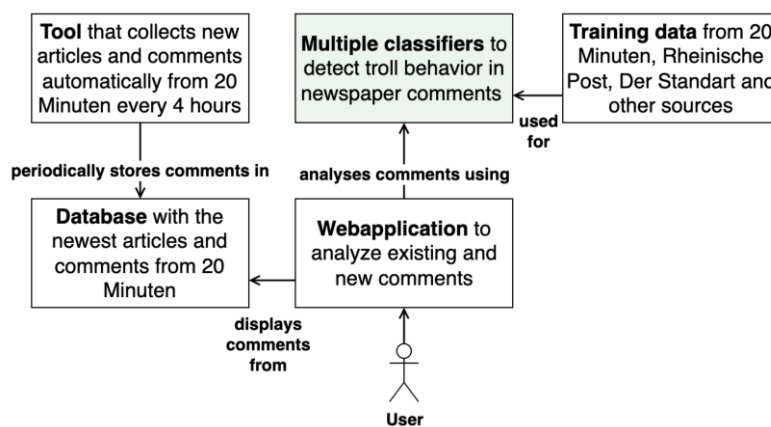


Figure 1: Overview of the project

To filter inappropriate comments, most newspapers in Switzerland rely heavily on manual moderation. In this thesis, several machine learning models were trained to detect inappropriate comments automatically.

Results

Multiple classification algorithms were developed to detect hate speech (83% correct results), off-topic comments (80% correct results) and state-linked propaganda (91% correct results). These classifiers were trained with data from different sources. For comments from the biggest Swiss

newspaper, 20 Minuten, the algorithms can correctly predict whether a comment will be accepted or rejected in 71% of the cases.

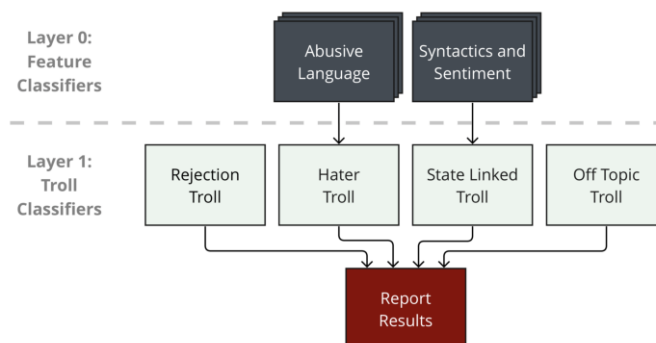


Figure 2: Overview of the classifiers

As a result, the classifiers could be used to support the human moderation team in their work. The performance is not sufficient to fully automate the moderation process. Instead, the algorithms can be used to automatically remove the most extreme comments: By adjusting the threshold, about 20% of the troll-comments can be automatically detected with almost no false positive.

To make the results accessible to lay users, a web application was developed. With the application, users can analyze their own comments with the algorithm or examine existing comments from 20minuten.ch.

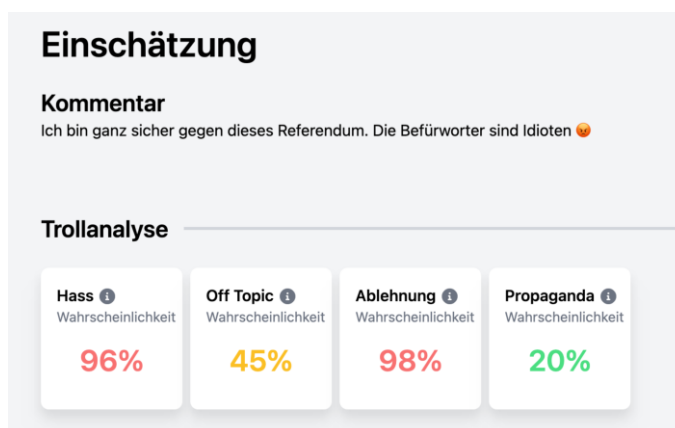


Figure 3: Screenshot of the web application showing multiple classifiers

Ideas for further research

In this work, mainly the content of the comments was analyzed. The algorithms are primarily specialized in detecting comments with inappropriate language. For future work, it would be interesting to focus on more subtle manipulation attempts and (paid) political trolls. A challenge in detecting such trolls is the lack of training data. Visualizations and pattern detection could be used to find suspicious patterns. Further research could also focus on the detection of troll accounts instead of individual troll comments.

Contents

List of Figures	vii
List of Tables	xi
List of Abbreviations	xiii
1 Introduction	1
1.1 Task Description	2
1.2 Statistical Hypothesis Test	4
2 Context Understanding	6
2.1 Definition of Internet Trolling	7
2.2 Trolls in Swiss Online Media	11
2.3 Overview of Existing Research	13
2.4 Existing Tools	14
3 Data Acquisition	17
3.1 Data Source Selection	18
3.2 Data Structure	21
3.3 20 Minuten Comment Importer Application	24
3.4 Manual Labeling of 20 Minuten Comments	33
3.5 Training Data	37
3.6 Import of '20 Minuten' Comments	40
3.7 Import of 'Der Standard' Comments	41
4 Data Modeling and Evaluation	42
4.1 Overall Classification Strategy	43
4.2 Rejection Words Classifier	44
4.3 Hate Troll Classifiers	50
4.4 State-Linked Troll Classifier	67
4.5 Off-Topic Troll Classifier	95
4.6 Metadata-Only Approach	105
4.7 Computing Performance Challenges	112
5 Application Proof of Concept	113
5.1 Functional Requirements	114
5.2 Non-Functional Requirements	115
5.3 Development Setup	116

5.4 Backend / API	117
5.5 Frontend	121
6 Conclusion	126
6.1 Summary	127
6.2 Validation of the Main Hypothesis	128
6.3 Outlook and Further Research	129
Bibliography	130
Glossary	136
A Unused Classifiers	139
A.1 Rejection Classifier Unused Building Blocks	140
A.2 Sarcasm Classifier Version 1	149
A.3 Comment Metadata Classifier for "20 Minuten"	157
B Code Structure	160
C Statistical Hypothesis Tests for the Metadata Classifier	166
D Statistical Tests and Feature Plots State-Linked Classifier	172
E Troll User Features	184
F Test Logs	186
G Literature Review on Data Analysis	190

List of Figures

1	Overview of the project	iii
2	Overview of the classifiers	iv
3	Screenshot of the web application showing multiple classifiers	iv
1.1	An example comment from 20minuten.ch with reactions from other users	2
2.1	Screenshot of a trolling forum	8
2.2	Example of a troll account on Twitter	9
2.3	Screenshot of the StopHateSpeech webapplication	14
2.4	Screenshot of the Perspective API demo	15
2.5	Screenshot of the Hamilton 2.0 Dashboard	15
2.6	Screenshot of the Trollfeed Dashboard	16
2.7	Screenshot of the Reddit Bot and Troll Dashboard	16
3.1	Comment from 20minuten.ch	21
3.2	Comment from Tagesanzeiger.ch	21
3.3	Comment from Blick.ch	21
3.4	Entity Relationship Diagram	22
3.5	Architecture of the comment-importer application	24
3.6	Import decisions	26
3.7	Automatic conversion from Java entity classes to database tables	27
3.8	Accessing the test data with the pgAdmin tool	27
3.9	A list of users with the most comments	28
3.10	A list of comments with the most negative reactions	28
3.11	A successful import report	29
3.12	An import report with errors	29
3.13	Java Docs for the importer application	30
3.14	Import statistics: imported articles per topic	31
3.15	Import statistics: comments per daytime	31
3.16	Import statistics: comments per positive reaction rate	32
3.17	Import statistics: average reactions per author	32
3.18	Data labeling setup	34
3.19	User interface of the labeling tool	35
3.20	Example of a rejected comment from the 20minuten.ch 2021 dump	37
3.21	Excerpt from FiveThirtyEight's troll dataset	38
3.22	Excerpt from crowd-voted comments	39
3.23	Transformation and import of the 20minuten.ch 2021 dump	40
3.24	Transformation and import of the one million posts corpus project database	41

4.1	Classifiers big picture	43
4.2	Rejection troll overview	44
4.3	ROC curve for different rejection classifiers	46
4.4	Confusion matrix for the rejection classifier (20 Minuten)	47
4.5	ROC curve for the rejection classifier	48
4.6	Hate troll overview	50
4.7	Combined dataset label count	51
4.8	Structure of the hate classifier	51
4.9	Logistic Regression <i>ROC</i>	52
4.10	Validation and training accuracy of unused classifier	52
4.11	Validation and training accuracy of unused classifier (2)	53
4.12	Validation and training loss of unused classifier	53
4.13	Hate RoBERTa classifier overview	54
4.14	Scatterplot of two dimensions reduced by PCA on train-set	54
4.15	Scatterplot of two dimensions reduced by PCA on test-set	55
4.16	Scree Plot of the reduced train-set	55
4.17	Confusion matrix for the <i>SVM</i>	56
4.18	German BERT hate speech classifier overview	57
4.19	Scatterplot of two dimensions reduced by PCA on train-set	57
4.20	Scatterplot of two dimensions reduced by PCA on test-set	58
4.21	Scree Plot of the reduced train-set	58
4.22	Confusion matrix: <i>KNN</i> classifier	59
4.23	German BERT Base hate classifier overview	60
4.24	Scatterplot of two dimensions reduced by PCA on train-set	60
4.25	Scatterplot of two dimensions reduced by PCA on test-set	61
4.26	Scree Plot of the reduced train-set	61
4.27	Confusion matrix: random forest classifier	62
4.28	Hate classifier ensemble overview	63
4.29	ROC curve of the ensemble classifiers	63
4.30	<i>ROC</i> : voting classifier	64
4.31	Confusion matrix: voting classifier	64
4.32	Precision recall curve: voting classifier	65
4.33	State-linked troll overview	67
4.34	Syntactics and sentiment classifier overview	68
4.35	State-linked troll classifier label counts	71
4.36	Dependency parsing sample graph	72
4.37	VIF plot	76
4.38	Feature selection scorings run	76
4.39	State-linked troll classifier structure	77
4.40	State-linked classifier ROC plots	78
4.41	Pairplots syntactic features	78
4.42	State-linked trolls usage of nouns (left) and adpositions (right)	79
4.43	State-linked trolls usage of links (left) and sarcasm (right)	79
4.44	State-linked trolls usage of reported speech (left) and emojis (right)	79
4.45	State-linked trolls usage of named person (left) and location (right) entities	80
4.46	Feature importance for the syntactic and sentiment subclassifier	80
4.47	State-linked classifier syntactic and sentiment confusion matrix	81
4.48	Sarcasm classifier version 2 overview	82
4.49	Non-sarcastic wordcloud (left) and sarcastic wordcloud (right)	84
4.50	Sarcasm classifier version 2: confusion matrix	85
4.51	Sarcasm classifier version 2 precision vs recall	85
4.52	Content meaning classifier overview	87
4.53	State-linked trolls non-fine-tuned model (left) and fine-tuned model (right)	88
4.54	Content meaning classifier: optimal number of principal components	89
4.55	VIF plot	89
4.56	State-linked classifier content meaning ROC plots	90
4.57	State-linked classifier contents meaning confusion matrix	90
4.58	Ensemble classifier overview	92

4.59	State-linked ensemble classifier ROC plots	92
4.60	State-linked classifier final confusion matrix	93
4.61	State-linked classifier precision vs recall	93
4.62	Off-Topic troll overview	95
4.63	Off-Topic comments from Der Standard with <i>SBERT</i> embedding similarities compared to the article	97
4.64	Off-Topic comments from Der Standard with Tfidf embedding similarities compared to the article	97
4.65	Off-Topic comments from 20 Minuten with <i>SBERT</i> embedding similarities compared to the article	98
4.66	Off-Topic comments from 20 Minuten with <i>SBERT</i> embedding similarities compared to other comments mean (left) and max (right)	99
4.67	Off-Topic comments from 20 Minuten with rate of article words in comment	100
4.68	Off-Topic comments from 20 Minuten with rate of article words in comment histogram (left) and <i>CDF</i> (right)	101
4.69	Off-Topic classifier final structure	102
4.70	Off-topic classifier ROC curves	102
4.71	Confusion matrix (left) and ROC curve (right) for the off-topic classifier	103
4.72	Confusion matrix (left) and ROC curve (right) for the off-topic classifier without comparison to other comments	103
4.73	Feature extraction using <i>SQL</i>	108
4.74	ROC curve for different metadata classifiers	109
4.75	Confusion matrix for the metadata classifier (der Standard)	110
5.1	Use cases	114
5.2	Hardware infrastructure for the project	116
5.3	Overview of the API	117
5.4	Explorable API	119
5.5	Screenshot: main page	121
5.6	Screenshot: analyzing a comment	122
5.7	Screenshot: an analyzed comment	122
5.8	Screenshot: an analyzed comment (2)	123
5.9	Screenshot: text-features of a comment	123
5.10	Screenshot: explore newssources	123
5.11	Screenshot: articles	124
5.12	Screenshot: comments	124
5.13	Screenshot: comment features	125
5.14	Screenshot: user features	125
A.1	Rejection classifier former overview	140
A.2	Rejection content meaning classifier former overview	142
A.3	Untrained <i>SBERT</i> (left) and fine tuned <i>SBERT</i> (right)	142
A.4	Rejection troll content meaning optimal number of principal components	143
A.5	ROC curve for different <i>SBERT</i> rejection classifiers	143
A.6	Rejection troll content meaning classifier	144
A.7	Rejection classifier former overview	145
A.8	ROC curves for different comment guideline classifiers	146
A.9	Comment guidelines classifier confusion matrix	146
A.10	Rejection classifier ensemble overview	147
A.11	ROC curves for rejection ensemble classifiers	147
A.12	Rejection troll ensemble confusion matrix	148
A.13	Rejection troll ensemble precision vs recall	148
A.14	Sarcasm classifier V1 overview	149
A.15	Sarcasm classifier <i>BERT</i> optimal number of principal components	152
A.16	Feature selection scorings	153
A.17	Sarcasm classifier model	154
A.18	Sarcasm classifier ROC curves	155
A.19	Sarcasm classifier: confusion matrix	155
A.20	Sarcasm classifier version 1 precision vs recall	156

A.21	ROC curve for different metadata classifiers	158
A.22	Confusion matrix for the metadata classifier (20 Minuten)	159
A.23	ROC curve for the metadata classifier (20 Minuten)	159
B.1	Overview of the repositories	162
B.2	Code structure of the 20min-JSON-to-database repository	163
B.3	Code structure of the 20min-Comment-to-database repository	163
B.4	Code structure of the troll analysis repository	164
B.5	Code structure of the troll api repository	164
B.6	Code structure of the troll frontend repository	165
F.1	Test logs: DateHelperTests	187
F.2	Test logs: StringHelperTests	187
F.3	Test logs: NewspaperRepositoryTests	187
F.4	Test logs: CommentConverterTests	187
F.5	Test logs: CommentFetcherTests	187
F.6	Test logs: RSSFeedsConverterTests	188
F.7	Test logs: RSSFeedsImporterTests	188
G.1	Density-reachability and connectedness	195
G.2	Kernel density	195
G.3	Noise threshold ξ	196
G.4	Linear decision boundary of LDA	199
G.5	Quadratic decision boundary of QDA	199
G.6	Non-Linear functions	200
G.7	Support Vector Machine with a polynomial kernel (left) and a radial kernel (right)	201
G.8	Split feature space (left) and the corresponding decision tree (right)	202
G.9	Flexibility vs. Interpretability	204
G.10	Sample ROC curve	206
G.11	Multi-Head attention (2)	209
G.12	Single sentence classification	210
G.13	BERT visualization using PCA	211
G.14	BERT visualization using UMAP	212

List of Tables

2.1	Example comments per category	7
3.1	Evaluation of 20 Minuten as a data source	18
3.2	Evaluation of Tagesanzeiger as a data source	18
3.3	Evaluation of Blick as a data source	19
3.4	Evaluation of Nau.ch as a data source	19
3.5	Evaluation of NZZ as a data source	19
3.6	Evaluation of SRF Online as a data source	20
3.7	Evaluation of Watson as a data source	20
4.1	Scores for the rejection words classifier	47
4.2	Rejection words classifier: test comments	48
4.3	Labels for the dataset (hate classifier)	51
4.4	RoBERTa: hyperparameter tuning results	56
4.5	Hate classifier: <i>SVM</i> scores	56
4.6	German BERT Hate: hyperparameter tuning results	59
4.7	German Bert hate classifier scores	59
4.8	German BERT Base: hyperparameter tuning results	62
4.9	German Bert Base classifier scores	62
4.10	Hate ensemble classifier scores	64
4.11	Hate troll classifier: test comments	65
4.12	Hate troll classifier: bias	66
4.13	Political terms used for the state-linked dataset	70
4.14	State-linked classifier: text cleaning	70
4.15	Features overview for the state-linked classifier	71
4.16	Syntactic dependency parsing labels (example)	72
4.17	Syntactic dependency parsing labels	73
4.18	Readability scores	74
4.19	State-linked classifier: optimal features sorted	77
4.20	State-linked classifier: hyperparameter tuning results	77
4.21	State-linked ensemble classifier scores	81
4.22	Sarcasm classifier version 2: optimal hyperparameters	84
4.23	Sarcasm classifier V2 scores	85
4.24	Sarcasm classifier: test comments	86
4.25	Features overview for the state-linked content meaning classifier	87
4.26	State-linked classifier: hyperparameter tuning results	89
4.27	State-linked content meaning classifier scores	90
4.28	State-linked ensemble classifier scores	93

4.29	State-linked ensemble classifier: test comments	94
4.30	Feature overview for the off-topic classifier	96
4.31	Embedding variants	96
4.32	Off-topic classifier scores	103
4.33	Simple off-topic classifier scores	104
4.34	Features of time behavior	106
4.35	Features for user interaction	106
4.36	Features of comment content	106
4.37	Other features	107
4.38	Brainstormed features	108
4.39	Scores for the metadata classifier (Der Standard)	110
5.1	Comparison of technologies for the API	118
A.1	20 Minuten comments guidelines coverage	141
A.2	Rejection troll content meaning classifier scores	144
A.3	Features overview for the comment guidelines classifier	145
A.4	Rejection troll comment guidelines classifier scores	146
A.5	Rejection troll content meaning classifier scores	148
A.6	Considered features for the sarcasm classifier	150
A.7	Sarcasm classifier: optimal features sorted	153
A.8	Sarcasm classifier: hyperparameter tuning results	154
A.9	Sarcasm classifier: scores	156
A.10	Sarcasm classifier: test comments	156
A.11	Scores for the metadata classifier (20 Minuten)	158
B.1	Distribution of the code per task	161
B.2	Distribution of the code per language	161
E.1	Features for a user	185
G.1	Sample confusion matrix	205

List of Abbreviations

- API** Application Programming Interface. 2, 15, 16, 18–20, 25–27, 37, 112, 113, 115, 117–120
- AUC** Area Under (the *ROC*) Curve. 52, 136, 155
- BERT** Bidirectional Encoder Representations from Transformers. 44, 51, 63, 87, 96, 104, 112, 141, 209
- CDF** Cumulative Distribution Function. ix, 101
- CSV** Comma-Separated values. 112, 191
- DOM** Document Object Model. 191
- EDA** Exploratory Data Analysis. 192
- EM** Expectation Maximization Algorithm. 194
- HTML** Hypertext Markup Language. 191
- HTQL** Hypertext Query Language. 191
- JSON** JavaScript Object Notation. 18, 19, 25, 115, 118
- KNN** K-Nearest-Neighbor. viii, 56, 59, 62, 77, 89, 154, 203
- NP** Nondeterministic Polynomial Runtime. 192
- NZZ** Neue Zürcher Zeitung. 9
- PCA** Principal Component Analysis. 54, 57, 60, 88, 142, 151
- POS** Part of Speech. 84
- QDA** Quadratic Discriminant Analysis. 56, 59, 62, 77, 88, 89, 151, 154, 199
- RAM** Random-Access Memory. 112
- RoBERTa** Robustly Optimized BERT Pre-training Approach. 58, 61, 62, 96

- ROC** Receiver Operating Characteristic. viii, xiii, 47, 52, 63, 64, 103, 136, 137, 143, 154, 155
- RSS** Residual Sum Squares. 193
- SBERT** Sentence BERT. ix, 37, 96–100, 103, 104, 142, 150, 151
- SQL** Structured Query Language. 27, 41, 115, 157
- SVM** Support Vector Machine. viii, xi, 52, 56, 59, 62, 77, 89, 154
- TF-IDF** Term Frequency–Inverse Document Frequency. 52, 96
- URL** Uniform Resource Locator. 29, 40

CHAPTER 1

Introduction

This chapter is intended to introduce the goal and main hypothesis of this thesis. Furthermore, the statistical hypothesis tests used later in this work are discussed.

1.1 Task Description

1.1.1 Problem

The project deals with two concrete problems:

- For newspapers, moderating comments is a time-consuming and often a manual process.
- For the readers of newspaper articles, it is difficult to recognize troll comments and troll users as such.

1.1.2 Goal

The main goal of this project is to detect so called "troll comments" using machine learning algorithms. A definition of trolling for this project is given in Section 2.1. Also, this project wants to embed comments in a larger context by displaying additional information for each comment and for each author of a comment.

1.1.3 Main Hypothesis

The main hypothesis of this thesis is as follows:

The proposed classifiers are helpful in supporting the manual moderation of inappropriate content in newspaper comments.

The final assessment of this hypothesis can be found at the end in Section 6.2.

1.1.4 Target Group

The target group for the tool are non-specialists who are interested in the topic of trolling and would like to learn more about it. The project is intended to create awareness for the problem and also appeal to people who are not technically experienced.

However, the insights gathered during the work should also be of interest to newspapers and provide ideas on how to recognize trolls in an automated way.

1.1.5 Process

In a first step, some background information about trolling and social hacking should be gathered to get a better understanding of the approaches that "internet trolls" use. The scope of the project should then be further specified. In particular, it should be decided where exactly to look for occurrences of suspicious activities.

Some features have to be created to rate individual comments as unsuspecting or suspicious (feature engineering). The content and the metadata of a comment (username, time and date, reactions on the comment) can provide information.

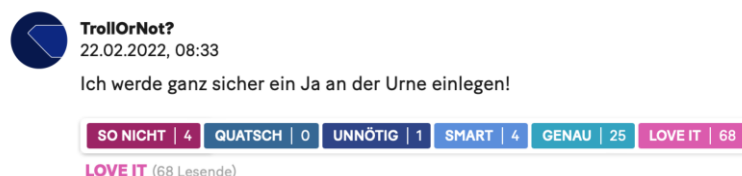


Figure 1.1: An example comment from 20minuten.ch with reactions from other users

Next, a set of newspaper comments should be collected. Multiple approaches are possible, namely periodic web scraping, access via *API* or as a static dump of old comments. A combination of multiple data sources is possible.

The collected data must be put into a structured format like a database. If there is labeled training data available, it can be used to train a classification algorithm.

The result of the analysis should give indications if a specific comment is inappropriate or not. The results should be presented in an easy-to-understand form to the user. The user should be able to see additional information for each comment. The information should help the users to better understand the context in which the comment was written and help to check the credibility of a comment.

1.1.6 Reference Study

This project aims to build upon existing scientific knowledge and use it for a concrete example. The meta study "A Survey on Troll Detection" summarized the findings of numerous research papers from this topic. "A Survey on Troll Detection" was a reference guide for this project [1].

1.2 Statistical Hypothesis Test

This section discusses the hypothesis tests used in Chapter 4 to validate the meaningfulness of the individual features.

1.2.1 Two Sampled t -Test

A t -test is a statistical test that is used to compare the means of two groups [2] [3]. This test can be performed when the two variances are assumed to be equal or unequal [4]:

Equal Variances

The t -test in case of two similar variances is calculated as [2]:

$$t = \frac{\bar{X} - \bar{Y}}{S_{(1+2)} \sqrt{\left(\frac{1}{n_X} + \frac{1}{n_Y}\right)}} \quad (1.1)$$

where $S_{(1+2)}$ is the pooled standard deviation [2]:

$$S_{(1+2)} = \sqrt{\frac{(n_X - 1)S_X^2 + (n_Y - 1)S_Y^2}{n_X + n_Y - 2}}$$

and \bar{X} and \bar{Y} are the sample means:

$$\bar{X} = \frac{1}{n_X} \sum_{i=1}^{n_X} x_i$$

$$\bar{Y} = \frac{1}{n_Y} \sum_{i=1}^{n_Y} y_i$$

and S_X^2 and S_Y^2 are the sample variances:

$$S_X^2 = \frac{\sum_{i=1}^{n_X} (x_i - \bar{X})^2}{n_X - 1}$$

$$S_Y^2 = \frac{\sum_{i=1}^{n_Y} (y_i - \bar{Y})^2}{n_Y - 1}$$

and n_X and n_Y are the sample sizes. The degree of freedom is calculated as [4]:

$$df = n_X + n_Y - 2 \quad (1.2)$$

Unequal Variances

If the variances are not similar, the t -statistic of the t -test is a Welch's t -test [2] [4]:

$$t = \frac{\bar{X} - \bar{Y}}{\sqrt{\frac{S_X^2}{n_X} + \frac{S_Y^2}{n_Y}}} \quad (1.3)$$

This statistics has a different degree of freedom which is calculated based on the Welch-Satterthwaite equation [2][5]:

$$df = \frac{\left(\frac{S_X^2}{n_X} + \frac{S_Y^2}{n_Y}\right)^2}{\frac{(S_X^2/n_X)^2}{n_X - 1} + \frac{(S_Y^2/n_Y)^2}{n_Y - 1}} \quad (1.4)$$

Assumptions of t -Tests

A t -test assumes the following conditions:

- The two samples for comparison must be independently sampled from the same population.
- The two samples have to satisfy the conditions of normality [2].
- The variances can be equal or unequal, but the appropriate formula must be employed as described previously.

Shapiro's test (Subsection 1.2.3) can be used to verify the assumption of normality and Levene's test (Subsection 1.2.4) can be performed to check the similarity of variances [2].

1.2.2 Kolmogorov–Smirnov Test

A KS-test is used to compare a sample with a reference probability (one-sample KS-test), or to compare two samples (two-sample KS-test) [6]. The latter is used extensively in Chapter 4. In contrast to the t -test, the KS-test does not assume a normal distribution of the samples [6].

The Kolmogorov-Smirnov statistic to test whether two distributions differ is defined as [6]:

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \quad (1.5)$$

where F_X and F_Y are the empirical distribution functions of the two samples X and Y and sup is the supremum function [6]. The closer $D_{X,Y}$ is to 0, the more likely it is that the two samples were drawn from the same distribution [6].

Generally, the null hypothesis is that the two distributions are identical [7]. The null hypothesis is rejected at significance level α if

$$D_{X,Y} > \sqrt{-\ln\left(\frac{\alpha}{2}\right)} * \frac{1}{2} * \sqrt{\frac{n_X + n_Y}{n_X * n_Y}}$$

Because the KS-test does not assume a specific distribution, a sufficiently large sample size is needed to properly reject the null hypothesis [7] [6]. This is why the sample sizes are used as a normalization factor in the equation above.

Cramér et al. have shown that a corresponding p-value can be calculated, which then can be used to accept or reject the null hypothesis [8]. The null hypothesis can be rejected if the p-value is smaller than the critical value [7].

1.2.3 Shapiro-Wilk Test

The Shapiro-Wilk test calculates a W -statistic. It indicates whether a random sample of data points follows a normal distribution [9]. A small w -statistic is evidence of departure from normal distribution [9]. The null hypothesis that the data is drawn from a normal distribution can be rejected if the p-value is greater than the significance level [10].

1.2.4 Levene's Test

The Levene's test is used to test whether two samples have equal variances [11] [12]. Generally the null hypothesis is that the variances are equal [12]. In addition to the test statistic W , the p-value can be calculated, whereby a small p-value suggests that the populations do not have equal variances [13].

1.2.5 Significance Level

The hypothesis tests performed in this thesis generally assume a significance level of 5%.

CHAPTER **2**

Context Understanding

This chapter clarifies the definition of internet trolling. Examples are used to show where trolls are active (especially in Switzerland) and the techniques they use to achieve their goals. Finally, existing solutions and related work is discussed.

2.1 Definition of Internet Trolling

2.1.1 General Definition

The definition of the term "trolling" is ambiguous. An Internet troll in the classic sense is a person who tries to stir up conflicts with provocative comments. A troll acts with destructive methods to harm a community:

“A troll is someone who systematically spreads negativity, disfavor, or bad mood. Trolls can be characterized by their destructive behavior.“

(Andreas Hobi - Head of the Blick Community Team)

A troll can either be a software (chatbot) or a real person.

2.1.2 Definition of "inappropriate behavior / trolling" for the scope of this project

Various studies were examined to find out how trolls are described. Each of the studies used a custom definition of "troll behavior". Here are some words used by other studies to describe troll comments: Aggressive, Angry, Anti-Social, Controversial, Deceptive, Destructive, Disrespectful, Discriminatory, Harassing, Insulting, Off-Topic, Offending, Offensive, Propagandistic, Provocative, Repetitive, Sparking debate, Senseless, Spam [1].

For this project, the following three categories are described as "trolling":

- **Hate comments:** This is a very broad term and it includes many of the terms discussed above (Aggressive, Anti-Social, Disrespectful, Discriminatory, Harassing, Insulting, Offending, Offensive, etc.).
- **Off-topic comments:** This includes comments that have no relation to the article or other comments, comments that are senseless or comments that contain advertisements.
- **State-linked propaganda comments:** This is the most difficult group of trolls to detect since not every comment that spreads the views or the propaganda of a state is necessarily a paid troll. Professional trolling is described in more detail in Section 2.1.4.

Here is an example comment for each category:

Table 2.1: Example comments per category

Hate	"Du bist ein Idiot!" (comment from 20minuten.ch)
Off-topic	"Im herbst ist die zeitumstellung falls es europa noch gibt" (20min.ch for the article "Ab Freitag gilt normale Lage - doch was ist im Herbst?")
State-linked propaganda	"Ich glaube eher, dass diese Eskalation von der Ukraine gewollt war, um den Westen weiter gegen Russland aufzubringen. Denn eines kann man wohl sagen, die Ukraine sucht irgendwie die militärische Auseinandersetzung. Aber nicht alleine, deswegen wird versucht die Nato und andere Staaten da reinzuziehen." (propaganda troll discovered by Twitter)

Also see Section 4.1.2 for a more detailed description of those three categories.

2.1.3 Related Definitions

- **Internet manipulation:** Trolling is a type of Internet manipulation. In addition, other techniques exist to manipulate the public opinion. This includes the use of automatic scripts and bots (social bots, votebots, clickbots) [14].
- **Influence Engineering:** This term describes a field of research in which the influence of social media on people's behavior is studied [15].
- **Sock-puppet Armies:** A group of sponsored trolls.
- **Bot:** A computer program that performs automated tasks. A chatbot is able to write messages that appear to come from a human.
- **Fake Account:** A social media account of someone pretending to be someone else.
- **Information Operator:** Twitter uses this term to define state-sponsored professional trolling campaigns to influence elections or to spread political views.

2.1.4 Professional Trolling

Many troll-comments can be manually recognized as such, for example when a comment contains hate speech. But there are also more subtle trolls: Professional trolls are paid by a client to conduct propaganda with their internet comments [16]. Such professional trolls can often not be identified by looking at a single comment.

Supposedly, the people who run professional trolling operations are often political groups, intelligence services or military institutions.



Figure 2.1: Screenshot of a trolling forum [17]

Professional trolls can have different goals [18]. These goals often include to:

- Influence the public opinion
- Polarize the society
- Silence political opponents
- Harm a competitor
- Strengthen the reputation of someone or something
- etc.

Examples of Professional Trolling Groups

Here are some examples of professional trolling:

- The 50 Cent Army spreads internet comments on behalf of China [19].
- North Korea relies on internet trolls to weaken the morale in South Korea [20].
- In Russia, the "Agency for Internet Research" is in charge of writing comments in the government's interest [21].
- The American Pentagon operates software to create fake internet identities. [22].

A whole series of other examples from various states can be found on Wikipedia¹.

Many countries have taken countermeasures to protect themselves from trolling campaigns. However, this is not an easy task, since professional trolls cannot always be clearly distinguished from normal users [23].

Platforms affected by professional trolling

Trolls are active wherever users can create their own posts on the Internet. This includes forums, games and social media such as Twitter and Facebook. On encyclopedias, trolls try to spread a certain view by editing articles. Such attempts at manipulation are particularly difficult to detect. Trolls often use true information that is presented in a misleading context [24].

The following screenshot shows a troll account on Twitter. The troll pretends to be a conservative American woman. In reality, this person does not exist. Her comments are created to polarize the public [25]:



Figure 2.2: Example of a troll account on Twitter

News sites also provide popular ground for trolls in their comment sections. For example, the *NZZ* (Neue Zürcher Zeitung) has temporarily closed its comment sections due to the low quality of comments [26].

¹https://en.wikipedia.org/wiki/State-sponsored_Internet_propaganda accessed on 07.04.2022

2.1.5 Manipulation Techniques

Some troll comments cannot only be detected from the content alone. There are many subtle manipulation techniques:

- **Spreading of black propaganda:** Creating posts that appear to come from someone else with the goal of harming that opponent.
- **Shilling:** Publishing a positive post about a person or organization without disclosing a personal connection to that person or organization.
- **Sock-puppeting:** Creating posts under a false identity.
- **Astroturfing:** Giving credibility to a statement by creating the impression that the statement is supported by grassroots participants from a movement.
- **Spreading of misinformation / Fake News:** The spreading of false information.
- **Creation of Fake Likes:** Manipulative liking of posts to give the impression a post is highly popular or unpopular.
- **Spinning:** Consciously (mis-)interpreting an event in a manipulative way.
- **Vote Brigading:** Manipulating an online poll by crowd voting, i.e. mobilizing many people.

2.1.6 Psychological Mechanisms

Manipulative posts can have multiple effects:

- The **Nasty Effect** describes that people evaluate content differently depending on the context surrounding this content. Regarding this project, this means that a newspaper article is judged differently depending on the comments made on this article.
- A second effect is the **spiral of silence**. The theory states that people are reluctant to express their opinion if they feel that their opinion does not fit in with the prevailing public opinion [27]. Trolls use this effect to silence political opponents.
- Related is the **Bandwagon Effect**. It describes that if a post initially receives some upvotes or downvotes, this trend is additionally strengthened in the course of further voting [28].

2.2 Trolls in Swiss Online Media

2.2.1 Contacting different newspapers

Several newspapers were contacted to find out how they perceive the trolling problem in general and what countermeasures they are taking.

2.2.2 Blick

According to Andreas Hobi, the number of non-professional trolls on `Blick.ch` has increased in recent years. Internally, an AI is used that automatically suggests comments to be removed. With the help of different thresholds, the comments are split into three groups: accepted, undecided, and rejected. Accepted comments are automatically published. The undecided comments have to be manually examined by the moderation team. About 10% of all comments are sorted out directly by the system. These are around 40'000 comments per month. This sounds like a huge number, but it should be noted that besides troll comments, the system also filters comments written in foreign languages and very short comments that do not contribute to a constructive discussion. Debatable comments are allowed in some cases. The comment analysis from `Blick.ch` only focuses on textual features. Metadata is not analyzed.

“The program automatically deletes the worst comments. This includes comments from "trolls", which in most cases contain keywords that are recognized by the algorithm.“

(Andreas Hobi - Head of the Blick Community Team)

Currently, the Russia-Ukraine conflict dominates the headlines. This has supposedly led to Russian trolls appearing in the comment sections of Swiss online media. Various newspapers have noticed this, including `Blick.ch`. However, it is not 100% clear which users are professional trolls, since they are difficult to distinguish from regular pro-Russian comments. `Blick.ch` does not have an automated system to detect professional trolls, so they often have to remove comments by hand.

“Usually, propaganda trolls are registered under a male name and completely focus on one topic, such as the Ukraine crisis.“

(Andreas Hobi - Head of the Blick Community Team)

If a comment looks suspicious and the corresponding account had just been created recently, the comment is deleted. Since this is tedious work, an automated solution would be helpful.

2.2.3 Watson

Watson distinguishes between three categories of rejected comments:

- unobjective comments (in German: "unsachlich")
- off-topic comments
- guideline-violating comments

To support moderation, Watson uses a list of "bad words" to detect inappropriate comments. Each comment is moderated manually:

“Wir haben die Erfahrung gemacht, dass das menschliche Auge die meisten Trolle am effektivsten enttarnt.“

(Adrian Eng, Chief of staff at Watson)

Adrian Eng also mentions the role of the community. The community helps Watson by reporting inappropriate comments. Watson does not use automated tools to analyze comments.

2.2.4 Neue Zürcher Zeitung

In contrast to `Watson.ch`, the Neue Zürcher Zeitung uses artificial intelligence to analyze comments. They use the tool "Conversario", which is widely used by German media. Conversario advertises that it recognizes hate comments and spam. Among other things, the artificial intelligence recognizes toxic words or questionable links and also analyzes the spelling and grammar of the comments. The metadata of the comments is not analyzed in depth. Reto Stauffacher, Head of the Social Media Team, writes that this tool makes the moderation-work much easier. On the Neue Zürcher Zeitung's Facebook page, the artificial intelligence automatically deletes some comments. On `NZZ.ch`, every automatically rejected comment is checked again by an employee.

According to Reto Stauffacher, it is difficult to detect comments from professional trolls. He notes that not everyone who has a "strange" opinion is automatically a troll. In case of doubt, comments are published as long as they do not violate the guidelines.

Reto Stauffacher says that he observes rather few professional troll attacks aimed at the Neue Zürcher Zeitung. Professional trolls have to be removed manually.

2.2.5 20 Minuten

About a third of the comments from `20Minuten.ch` gets rejected by moderation. 20 Minuten uses a manual review of the comments as well as automated classification. For the automated classification, Google's Perspective API is used as described in Section 2.4.3

2.3 Overview of Existing Research

2.3.1 A Survey on Troll Detection

The 2020 study "A Survey on Troll Detection" evaluated existing research [1].

It should be noted that all the projects studied have chosen a different definition of "trolling". Nevertheless, the projects often choose similar approaches to detect trolls in a given domain. A common approach is to use natural language processing techniques such as splitting posts into n-grams for further analysis. Other approaches are more related to the semantics of a text. Sentiment analysis is a very commonly used tool. Metadata about individual users or comments has also been used in projects to detect trolls. The social interactions between different users (for example, likes and dislikes per comment) are a particularly powerful feature to detect trolls. For some domains, trolls can often be identified based on their habits: for example, trolls tend to write shorter comments [29] or comments that are harder to read [30]. It turns out that in many domains, a combination of different techniques is often capable of detecting trolls very well.

In research, a distinction is made between different methods: post-based methods, thread-based methods and user-based methods.

- With post-based methods, comments are analyzed independently from one another. Here, the semantics of a comments plays a particularly important role. In many projects, comments are transformed into a multidimensional vectors to train classification models.
- With thread-based methods, the analysis happens on a higher level. Not only the comments are analyzed individually, but entire conversations.
- User-based methods classify entire users as "troll" or "non-troll" rather than individual comments. Many of these classifiers achieve an accuracy of more than 80%.

It is also noteworthy that the research results for one domain often transferred well to other domains.

2.3.2 Distinguishing Characteristics of this thesis

This project combines several techniques used in existing research and tries to build upon commonly used techniques. These techniques include:

- Metadata analyses
- Natural language processing methods
- Sentiment analyses
- Semantic analyses
- Combination of post-based and thread-based methods

Multiple classifiers are trained for this project. Each classifier is specialized on a different category of troll. Thanks to the collaboration with 20 Minuten, significantly more training data is available for this project than for similar projects.

2.4 Existing Tools

2.4.1 Analyzed Projects

There is a lot of scientific work that aims to detect influence campaigns and hate speech. The next sections present some existing solutions. The requirement for a project to be included in this list was primarily the frontend and usability for lay users. All the solutions presented want to raise awareness for the topic "trolling". The tools allow users to interactively discover the problem of online manipulation.

2.4.2 StopHateSpeech

StopHateSpeech is an organization that wants to protect people from online harassment². They use an algorithm to detect hate speech in online discussions. Among other data, the algorithm was trained with comments from `Blick.ch`. Participants can help to improve the algorithm by classifying comments. There is also a "counter speech platform" that allows participants to intervene directly in ongoing discussions. This platform gives participants tips on how to react correctly to hate speech. This is done in collaboration with ETH Zurich and the University of Zurich, which are testing various counter speech strategies for their effectiveness. Anyone can register on the site and join the project.

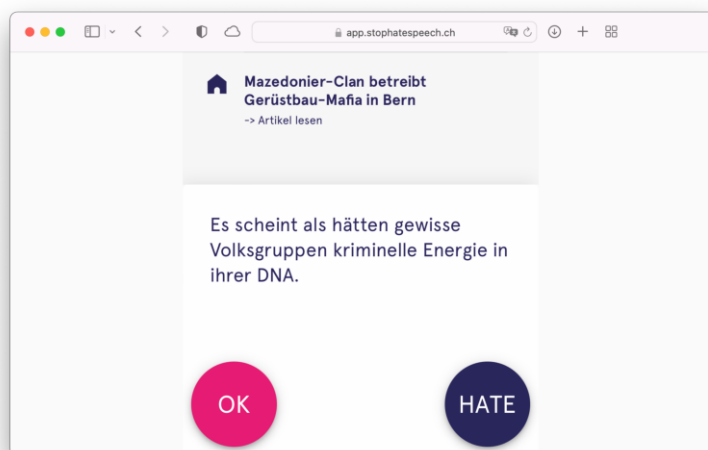


Figure 2.3: Screenshot of the StopHateSpeech webapplication

²<https://stophatespeech.ch>

2.4.3 Perspective API

Jigsaw is part of google. The unit creates solutions that explore threats to open societies, such as online trolling. One of the tools created by Jigsaw is Perspective. The goal of Perspective is to detect toxic comments³.

The tool is open for developers, which means that everyone can use the public API to check whether a comment is toxic or not. Only the content of the comment itself is used for the assessment using machine learning. In a demo-webapp, anyone can enter a comment and check the "toxicity-level" that is calculated for a comment. Some newspapers (including 20 Minuten) use the Perspective API as an integration to their comment moderation system [31].

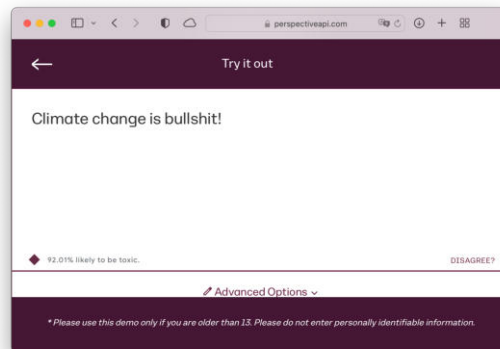


Figure 2.4: Screenshot of the Perspective API demo

2.4.4 Hamilton 2.0

Hamilton 2.0 is an interactive dashboard. It contains various metrics and charts of state-activity on Twitter and YouTube. Only accounts that are under government control or close to a government are considered for the analysis. However, this does not necessarily mean that all posted content is propaganda material. Nevertheless, the tool provides an interesting insight into which narratives are currently being spread. The tool uses auto-translation to translate the content and then analyzes it with natural language processing [32].



Figure 2.5: Screenshot of the Hamilton 2.0 Dashboard

³<https://perspectiveapi.com/>

2.4.5 Trollfeed

This project is less about analysis and more about countermeasures against online hate. The organization behind the tool, Detect Than Act, is supported by the European Union. Trollfeed scans social media for toxic messages and provides users with a dashboard to react to those messages. The tool works in different languages and uses artificial intelligence for the classification of the messages. It was trained with thousands of offensive words. Users can respond to toxic messages using some pre-selected visual memes or text. Also, messages can be reported if they violate the law [33].

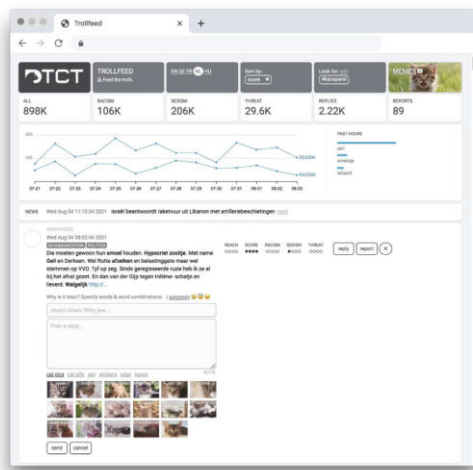


Figure 2.6: Screenshot of the Trollfeed Dashboard

2.4.6 Reddit Bot and Troll Dashboard

Unfortunately, the "Reddit Bot and Troll Dashboard" is not online anymore. Still, it is an interesting reference for this project as it is open source and well documented.

The dashboard shows posts from Reddit and classifies each post as "normal user", "possible bot" or "possible troll". The Reddit *API* is used to receive data about comments and users. The Reddit *API* offers a broad range of metadata, which helped the developers to create a decision tree classifier with good accuracy. The initial training dataset is publicly available from Reddit and includes some known bots and suspected trolls. Python's *scikit* library was used for the machine learning algorithms.

Numerous different metrics were used as input for the algorithm. It turned out that the "average difference ratio for the most recent posts of a user" was the most important metric. This means that bots and trolls often copy-pasted similar comments to different locations. Another essential metric was the reactions that a comment received [34].

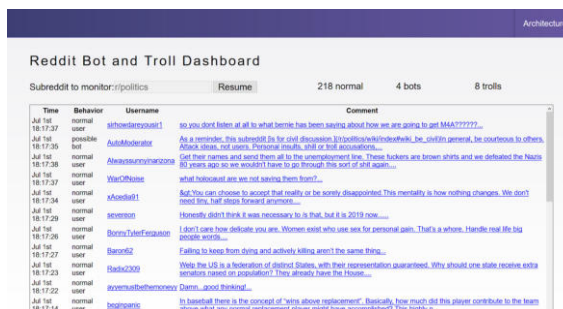


Figure 2.7: Screenshot of the Reddit Bot and Troll Dashboard

CHAPTER 3

Data Acquisition

Data collection is the first and an essential step in data analysis. This chapter describes how the data is gathered for this study.

3.1 Data Source Selection

3.1.1 Criteria

First, a source for comments was selected.

This thesis relies on data from user comments of newspaper articles, including the corresponding metadata. Only German-language newspapers from Switzerland with a supra-regional reference were considered as possible data source. The following newspapers were evaluated: 20 Minuten, Tagesanzeiger, Blick, Nau, NZZ, SRF Online, Watson. It was assessed how well a data retrieval via an *API* or web-scraping is technically feasible and what kind of data can be obtained via the source. Some newspapers were contacted directly with the question whether they could send us data on the comments.

3.1.2 Evaluated Newspapers

20 Minuten

Table 3.1: Evaluation of 20 Minuten as a data source

Publisher	Tamedia / TxGroup
Print run in 2018	569'618 [35]
Paywall	No
API	There is an undocumented internal <i>JSON API</i> . The data contains all comments, usernames, and reactions.
Example API-URL	https://api.20min.ch/comment/v1/comments?tenantId=6&contentId=424276129278
Web Scraping	Rather cumbersome. The comments are not loaded all at once. The HTML structure is consistent.
Direct Contact	20 Minuten offered a dump of comments from 2021.

Tagesanzeiger

Table 3.2: Evaluation of Tagesanzeiger as a data source

Publisher	Tamedia / TxGroup
Print run in 2018	213'738 [35]
Paywall	No
API	There is an undocumented internal <i>JSON api</i> . The data contains all comments, usernames, and reactions.
Example API-URL	https://www.tagesanzeiger.ch/disco-api/v1/comments/article/488506735755/newestFirst/
Web Scraping	Well feasible. The comments are loaded all at once and the HTML structure has a very clean layout.
Direct Contact	The Tagesanzeiger has offered a dump of comments for multiple Tamedia / TxGroup publications.

Blick

Table 3.3: Evaluation of Blick as a data source

Publisher	Ringier
Print run in 2018	231'235 [35]
Paywall	No
API	There is an undocumented internal <i>JSON API</i> . The data contains all comments, usernames, and reactions. Compared to the other APIs, additional data is returned for the individual comment-authors, for example, gender.
Example API-URL	https://community.ws.blick.ch/community/comment/?page=2&discussion_type_id=17279551
Web Scraping	Rather difficult. The comments are reloaded. Moreover, the CSS classes have automatically generated names.
Direct Contact	Blick offered to send a dump of 2.5 million comments, including both accepted and deleted comments with metadata. However, it transpired that it would take too long to complete a contract.

Nau

Table 3.4: Evaluation of Nau.ch as a data source

Publisher	Nau media AG
Print run in 2018	(No Print, only Online)
Paywall	No
API	There is an undocumented internal <i>JSON API</i> . The data contains all comments, usernames, and reactions. It is possible to post anonymous comments without a user account, which makes analysis difficult.
Example API-URL	https://api.nau.ch/frontpage/articles/66136162/comments
Web Scraping	Well doable. The HTML is structured in a very understandable way.
Direct Contact	Nau did not respond to our contact request.

NZZ

Table 3.5: Evaluation of NZZ as a data source

Publisher	Neue Zürcher Zeitung
Print run in 2018	143'009 [35]
Paywall	Yes
Direct Contact	NZZ did not respond to our contact request.

SRF Online

Table 3.6: Evaluation of SRF Online as a data source

Publisher	SRF
Print run in 2018	(No Print, only Online)
Paywall	No
API	The comments are returned as HTML, which makes reading them difficult. Most articles can not be commented or are rarely commented. Compared to other newspapers, comments can only be liked, but not disliked.
Example API-URL	https://www.srf.ch/comments/18866096/3/13/html
Web Scraping	Rather difficult to do.
Direct Contact	SRF was not asked directly for data.

Watson

Table 3.7: Evaluation of Watson as a data source

Publisher	FixxPunkt AG
Print run in 2018	(No Print, only Online)
Paywall	No
API	There is no <i>API</i> to access comments individually.
Web Scraping	Rather difficult to do. The comments are post-loaded. The HTML is structured consistently.
Direct Contact	Watson did not respond to the request.

3.1.3 Selected Datasource for Analysis

20 Minuten was selected as the main data source. It is the largest newspaper in Switzerland, with a large volume of comments. It would have been possible to include other platforms from Tamedia / TxGroup as well, however this idea was discarded to lower the complexity of the project. Comments from 20 Minuten are gathered using two different methods:

- Recent comments are periodically fetched via *API*. Since there is no payment barrier and a simple *API*, it is easy to access the comments. The tool used to fetch the comments is described in Section 3.3. Those comments are primarily used for the frontend.
- 20 Minuten provided a dump of comments from 2021. This dump is described in Section 3.5.2. Those comments are primarily used for the training of classification algorithms.

3.2 Data Structure

3.2.1 Unified Data Structure

A database was created to store the comments. It was a goal that the data structure is not only compatible with comments from 20 Minuten but compatible with comments from different sources. Looking at the comment sections and API's from different Swiss newspapers, it turned out that they all seem to work with a similar structure for their comments and reactions to comments. Here are some examples of how comments from different news sources look like:

A comment from 20min:



Figure 3.1: Comment from 20minuten.ch

A comment from Tagesanzeiger



Figure 3.2: Comment from Tagesanzeiger.ch

A comment from Blick:

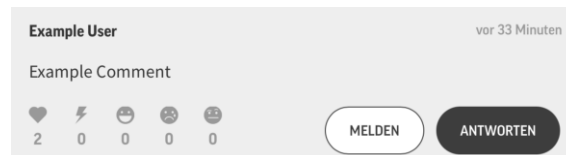


Figure 3.3: Comment from Blick.ch

3.2.2 Entity Relationship Diagram

Because every newspaper evaluated was working with the same concepts (comments, replies, users, reactions...), it was possible to find a common data structure. The structure is flexible enough that it can contain comments from different news sources:

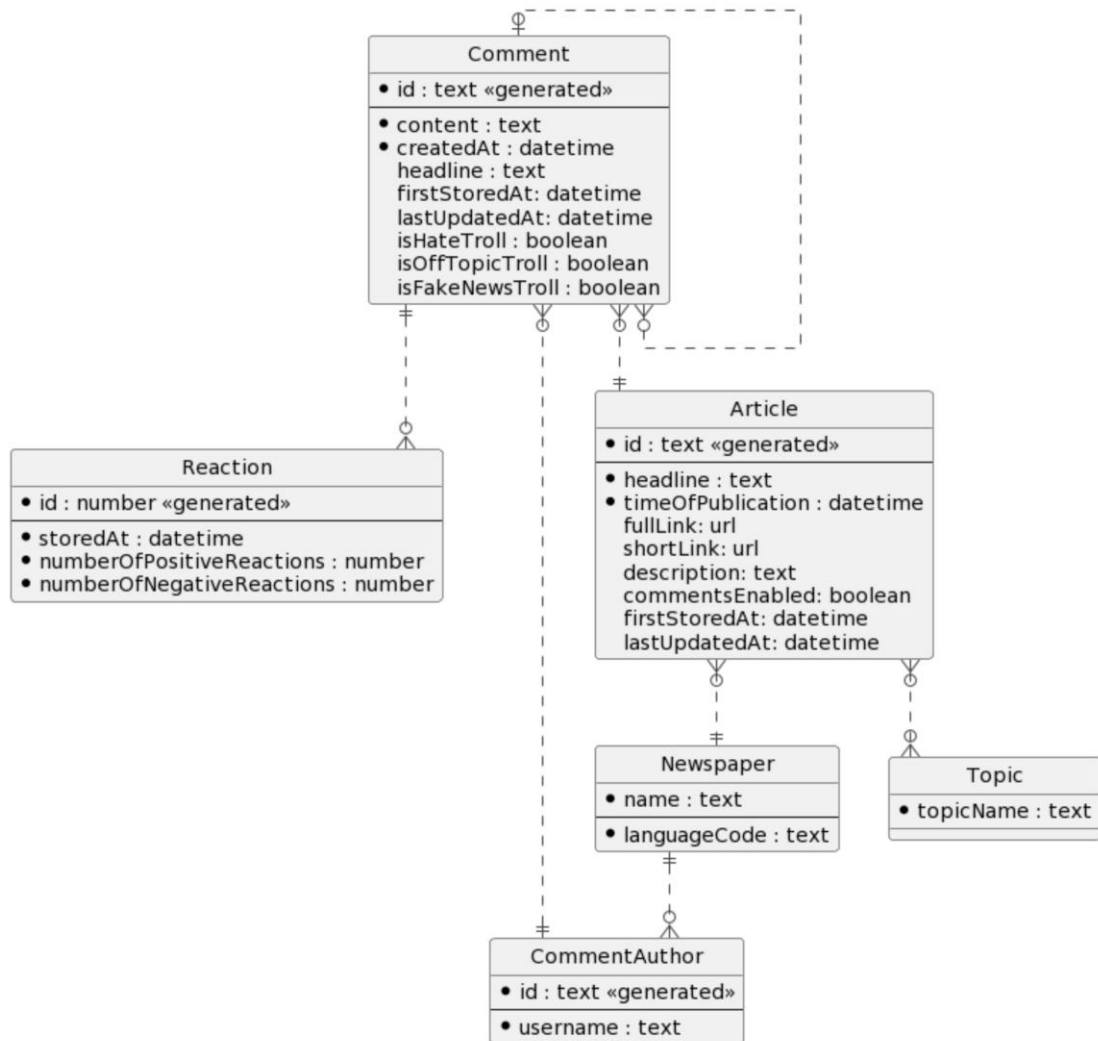


Figure 3.4: Entity Relationship Diagram

Explanation of the diagram:

- A dot means that an attribute is non-optional.
- Every entity has a unique primary key.
- Foreign keys are not explicitly listed as attributes in the image but only as a relationship between two entities.

Entity "Comment"

- Every comment can be in relation to another comment. This is the case if a comment is a reply to another comment. If a comment does not reference another comment, it means that this comment is at the top-level of comment hierarchy.
- Each comments references exactly one author and one article.
- A comment can be labeled as "troll" comment or "non-troll" comment. This attribute can also be `null` for unlabeled data. In the real database there are additional labels as well which are not included in this image for simplicity.

Entity "CommentAuthor"

- The CommentAuthor entity only contains the username.
- It would be helpful to store additional data about authors, such as the date of registration or the ip address. However, it is unlikely that this kind of data is available to work with.

Entity "Article"

- An article contains the headline and the time of publication.
- The content of the article itself is not part of the data structure, as this would drastically increase the complexity of the data structure and the size of the data.
- However, articles can have a description containing a lead or an excerpt of the article.
- Each article is associated with exactly one newspaper.

Entity "Topic"

- Each article can have one or many associated topics. For example "corona" and "international". Each topic can be associated with multiple articles as well which means that a relationship-table is used.

Entity "Newspaper"

- The structure can contain different newspapers as the source of articles.

Entity "Reaction"

- On every news source considered, users can react to comments. This is either a "like" or "dislike" button, or a bigger range of reactions that can be selected.
- To be compatible with different news sources, the reactions in this data structure are simplified. The data structure only stores the number of positive and negative reactions.
- News sources like 20 Minuten however have a broader range of reactions. Those reactions have to be mapped as either "positive" or "negative". For example, "Love it", "Smart" and "Genau" would count as positive reactions, whereas "Unnötig", "Quatsch", "So nicht" would count as negative reactions.
- Every comment can refer to multiple "Reaction" entities. This might look confusing at first. The idea is that reactions for the same comment can be stored multiple times for different times and dates. This allows to analyze how the reactions to a comment change during the day.

Implicit Assumptions

Here are some implicit conditions defined for the data:

- A comment can only reference another comment if both comments reference the same article.
- A comment cannot reference itself.
- A comment author who is associated with a newspaper can only comment on articles of the same newspaper.

3.3 20 Minuten Comment Importer Application

3.3.1 Summary

An application to periodically fetch recent comments from 20Minuten.ch was created for two reasons:

- Setting up a contract and organizing a data delivery by 20 Minuten took some time. The Importer Application was therefore used to collect a first dataset for experimenting and compiling analysis.
- The collected comments are used for the application proof of concept which was created later (see Chapter 5). Legally, it is not possible to use the data from the 2021 dump to display on the frontend. Also for the users it is more compelling to see the most recent comments.

First, the Java Spring Boot application was used to load comments from selected articles into the database. Later, the application was extended further to periodically load comments for the newest articles available online. "20 Minuten" was chosen as the data source but the tool was built in a modular way that would allow it to connect to other data sources as well.

3.3.2 Architecture

To build the importer, the Spring Boot Framework for Java was used, and PostgreSQL for the database. This was a very pragmatic decision, since Spring makes loading and storing data from and to a database effortless and requires little code. Furthermore, Spring Boot was chosen because the initial setup effort was small (convention over configuration principle). A comparison of different evaluated technologies can be found in Section ??.

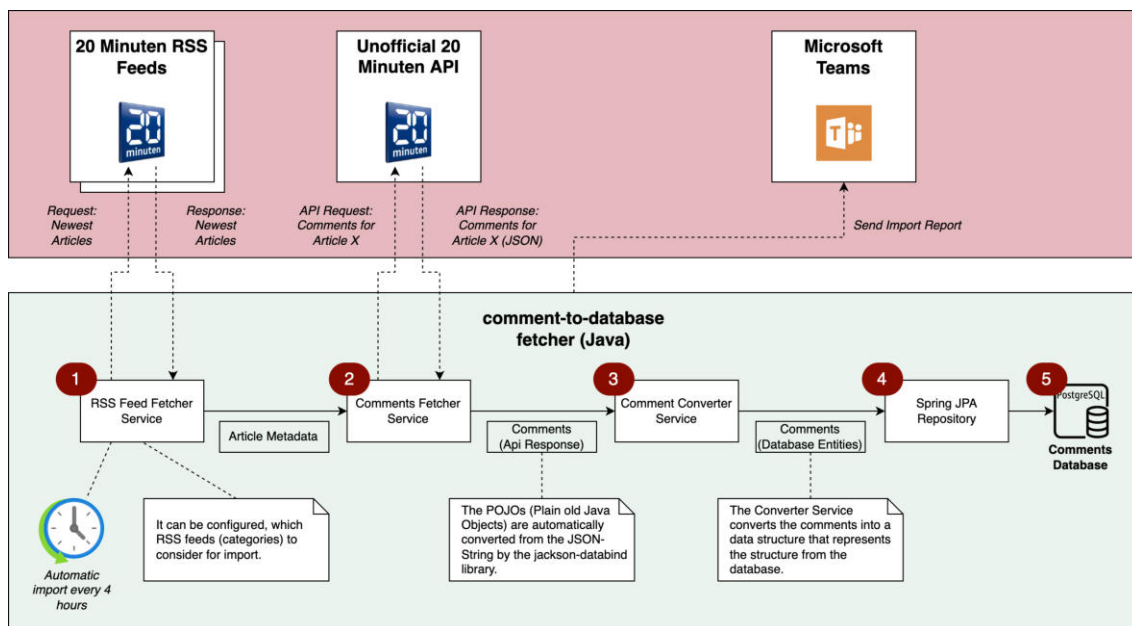


Figure 3.5: Architecture of the comment-importer application

3.3.3 Technology

Here are some libraries, frameworks and tools used for the importer:

- **Maven:** To manage dependencies
- **Lombok:** A library to generate some boilerplate Java code automatically (setters, getters, constructors, etc.)
- **Rome:** To parse RSS feeds
- **Spring Web:** To make *API*-Requests over HTTPS
- **Spring JPA:** For the auto-generation of database repositories and persistence of Java entity classes to the database
- **Jackson Databind:** To automatically populate Java objects with the data from a *JSON* string

An `RSSFeedFetcher` Service loads a list of the newest articles from `20minuten.ch`. In the configuration, it can be defined which RSS feeds to ignore and which RSS feeds to consider for import. The `FetcherService` then loads the comments from the individual articles as a *JSON* string from the 20 Minuten *API*. Using the Jackson library, this *JSON* String is then automatically filled into Java objects to make the data easier to access. Next, the data is converted into the required uniform structure for the database. This database structure is described in Section 3.2.2. In Java, the structure for the database was mapped using Java entity classes. A code-first approach was used for the database: The database tables are automatically generated from the Java entity classes. The *object relational mapper* stores and loads data from the database, so the code does not contain any SQL-Statement.

In the following subsections, the components will be explained in more detail.

3.3.4 Periodic Import Scheduler

Using a `Spring Scheduler`, the import of new articles and comments is started every four hours:

```
1 /**
2  * Periodically import new articles and comments from 20 Minuten into the database
3  *
4  * @return a List of imported or skipped articles and articles that could not be
5  *         ↪ imported because of an error
6  */
7 @Scheduled(cron="0 0 1,5,9,13,17,21 * * ?") //every day at 1:00, 5:00, 9:00, 13:00,
8     ↪ 17:00, 21:00
9 public List<ArticleImportResult> importCommentsFromNewArticles() {...}
```

Listing 3.1: Periodic imports of comments

Comment-Reactions for the same comment can be imported multiple times. This allows to analyze for each comment how the distribution of "likes" and "dislikes" has changed during the day.

20 Minuten closes the comment section for articles some time after publication. Therefore, the tool periodically only imports articles that are less than two day old. Older articles and comments are imported as well but only once (not periodically), because the comments for those articles will not change anymore:

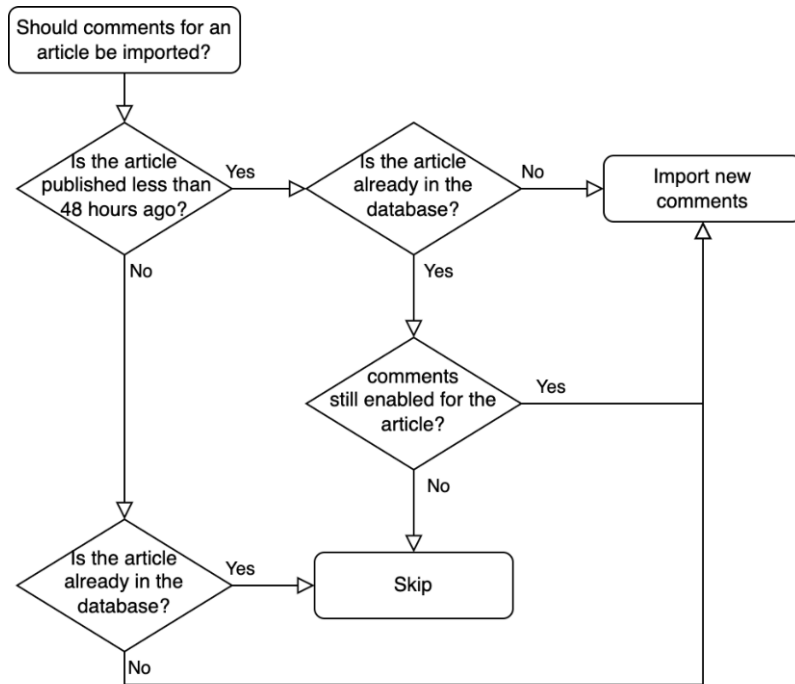


Figure 3.6: Import decisions

3.3.5 RSS Fetching

20 Minuten has an RSS Feed to fetch the newest articles available online¹. There is a main feed that contains articles from all categories and also individual feeds for specific topics (for example regional news, international news, sports, and so on). The application uses individual feeds because it allows finer control over which feeds should be imported and which feeds should be ignored. Also, the usage of the individual feeds made it possible to store the "topics" of an article as an attribute. This information is not available on the main feed. It can happen that the same article is posted in two different feeds (for example an article might be in the "international" feed as well as the "coronavirus" feed). The application detects such duplicates and can save multiple topics per article while the article is only imported once.

The RSS feeds contains additional metadata about each article, for example a headline, a short description, a link and a publication date. This metadata is stored as well.

3.3.6 20 Minuten API

The 20 Minuten *API* is undocumented. However, an analysis of the browser network traffic revealed the usage of the *API*. The *API* is used to fetch the comments for a given article.

For each comment, the content of the comment, the username and the date of creation are returned. In addition, the number of reactions from other users to the comment are returned as well ("awesome", "bad", "nonsense", "unnecessary", "smart", "exact"). Each comment contains a list of reply-comments.

3.3.7 Comments and Reaction Fetching

The *API* call takes place inside the `CommentFetcherService`. It was considered that errors can occur during this step, for example because the *API* has changed or because there is no connection to the server of 20 Minuten. Custom exception classes were defined for the error handling.

¹<https://partner-feeds.20min.ch/rss/20minuten>

3.3.8 Comments and Reaction Fetching Transformation

The *API* response is represented with the classes `ApiResponseCommentList`, `ApiResponseComment` and `ApiResponseCommentReaction`.

These objects are converted by the `ConverterService` into the entity classes needed for the database (`Article`, `Comment`, `CommentAuthor`, `Newspaper`, `Reaction`, `Topic`).

Basically, the structure of the data remains similar. One challenge was the recursive relationship between comments.

3.3.9 Data Storage

Thanks to the use of Spring JPA, hardly any custom code was needed to create the repository. The repository is responsible for the communication with the database and the conversion of Java entity objects into *SQL* statements. The repository only needs to be defined as an interface. The actual implementation is automatically created by Spring JPA.

3.3.10 Database

The database tables are automatically generated from the entity classes (code-first-approach):

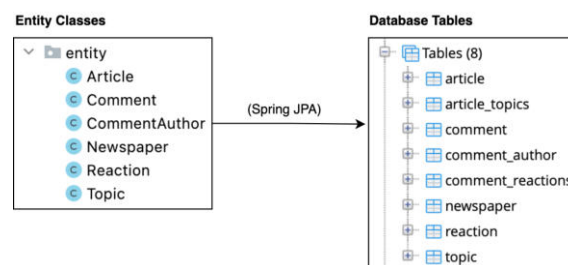


Figure 3.7: Automatic conversion from Java entity classes to database tables

The tool is not hardwired to the Postgres system. Other database systems like MySQL or SQLite could be connected by a simple change in the configuration.

The collected data can now be queried via the database:

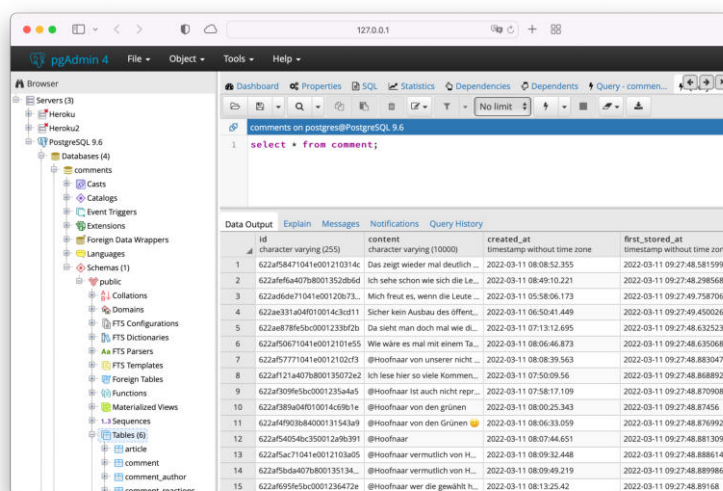


Figure 3.8: Accessing the test data with the pgAdmin tool

Simple analyses of the data are already possible just by querying the database. Here are some examples:

Show users with the most comments

```

1 SELECT count(*) as amount_of_comments, comment_author_username
2 FROM comment
3 GROUP BY comment_author_username
4 ORDER BY amount_of_comments DESC

```

Listing 3.2: SQL query: Show the list of users that write the most comments

	amount_of_comments bigint	comment_author_username character varying (255)
1	65	Ex...
2	45	Y...
3	25	S...

Figure 3.9: A list of users with the most comments

Show comments with negative reactions

```

1 SELECT comment.content, 1.0 * amount_of_positive_reactions /
   ↪ amount_of_negative_reactions as positive_reaction_ratio
2 FROM reaction
3 JOIN comment ON comment.id = reaction.comment_id
4 WHERE (amount_of_negative_reactions > 0)
5 AND (amount_of_negative_reactions + amount_of_positive_reactions) > 10
6 ORDER BY positive_reaction_ratio ASC

```

Listing 3.3: SQL query: Show comments with more than 10 reactions and the largest proportion of negative reactions

	content character varying (10000)	positive_reaction_ratio numeric
1	@Telefonhörer Meinen Sie so ein Kraftwerk wie in Tschernobil, dass in den falschen Händen zur Wa...	0.00000000000000000000
2	@meineMeinung1 SVP und rechts haben auch noch nie Probleme gelöst.	0.00000000000000000000
3	Gratuliere allen die auf Elektro umgestiegen sind. Ich leider noch nicht.	0.00000000000000000000
4	@Mjolnir / 45 Minuten zur Arbeit ist etwas weit, meinst du nicht auch ? Kann man ändern !	0.00000000000000000000

Figure 3.10: A list of comments with the most negative reactions

3.3.11 Reports and Logs

After each run (every 4 hours = 6 times per day), the tool creates a report about the imported articles and comments. This report is sent to a Microsoft Teams Channel using a webhook. As the tool runs silently in the background on the server, the reports allow us to check if everything works fine:

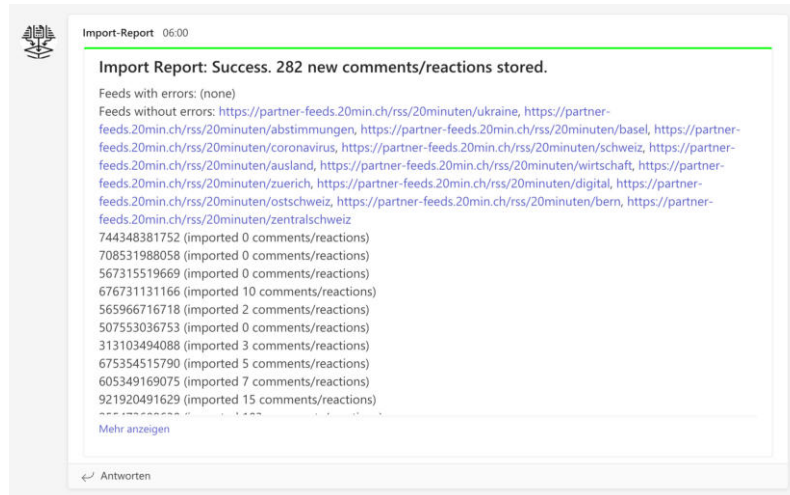


Figure 3.11: A successful import report

3.3.12 Exception Handling

The importer was built in a robust manner. It will still continue to run, even if some of the feeds, articles, or comments cannot be fetched or parsed. It was important to prevent the tool from crashing, for example, if a feed *URL* changes. In the case of an error, a report-notification with the description of the error is sent:

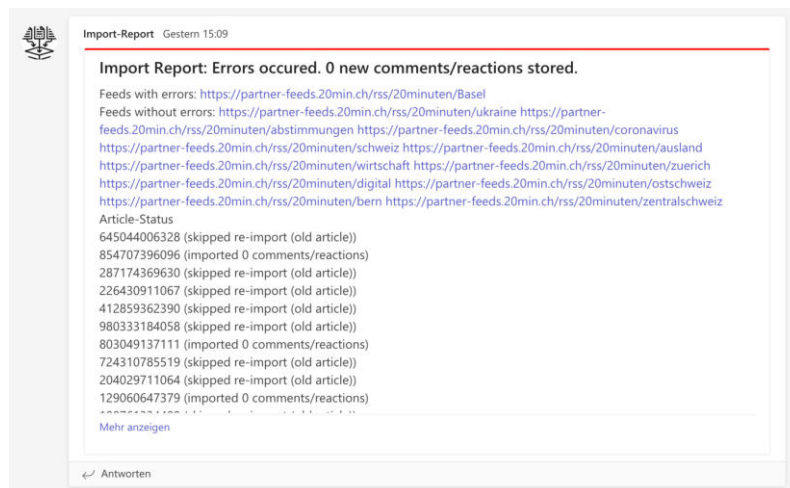


Figure 3.12: An import report with errors

3.3.13 Documentation

The Java classes and public methods are documented using Java Docs:



Method Details

fetchCommentsFromApi

```
public ApiResponseCommentList fetchCommentsFromApi(ArticleMetadata articleMetadata,
                                                    int commentLimit)
    throws CannotFetchFromApiException,
           CannotConvertApiResponseToObjects
```

Fetches a list of comment from the 20 Minuten Api for a given article

Parameters:
articleMetadata - additional metadata about the article
commentLimit - maximum amount of comments to fetch

Returns:
 an object containing the api response with a list of comments

Throws:
CannotFetchFromApiException - if the api cannot be accessed
CannotConvertApiResponseToObjects - if the api response can not be converted

Figure 3.13: Java Docs for the importer application

An installation guide can be found in Appendix ??.

3.3.14 Configuration

The importer uses a configuration file to specify the database connection and other settings.

Some settings to be configured include:

- maximum number of comments to fetch per article
- maximum number of articles to fetch per run
- maximum age of an article in hours to be imported periodically
- which rss feeds to include in the import and which feeds to skip

3.3.15 Automated tests

Critical parts of the importer are covered using automated unit tests. There are 36 tests in total, a detailed test log can be found in Section F.1.

The following tools and frameworks were used to write the tests:

- **JUnit:** As testing framework
- **H2 Database:** In-Memory database to prevent the tests from changing the real database
- **Mockito:** To create mock dependencies for isolated tests

3.3.16 Import Statistics

All figures and statistics mentioned refer to the imported comments until Mai 30, 2022.

In total, 2'414 articles were imported from different categories:

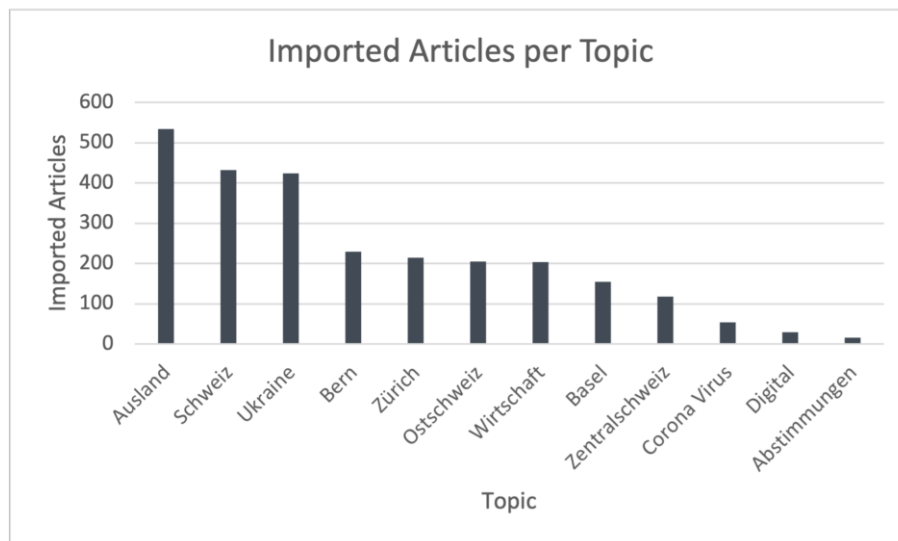


Figure 3.14: Import statistics: imported articles per topic

Those articles include a total number of 104'942 imported comments, written by 14'769 comment authors. 42% of all the comments are replies to other comments. All comments together received a total number of 7'625'481 reactions, 68% of all reactions were positive.

Some topics are more controversial than others. Users are more likely to comment on articles about politics, international news or economy than other topics. Regional articles have fewer comments in average. An article about the speed limits on the streets received the most comments, 827 in total.

It does not come as a surprise that most of the comments are either written in the morning, in the evening or at noon. There are less comments published during working-hours and at night:

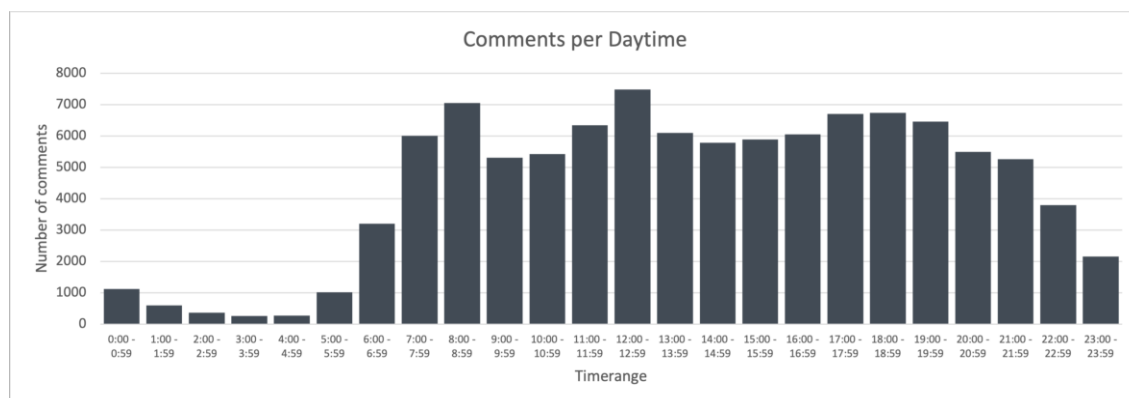


Figure 3.15: Import statistics: comments per daytime

Most of the comments receive positive reactions from the community. The following chart shows the 'positive reaction rate' for the comments. A positive reaction rate of 70% means that a comment received 70% positive reactions (likes) and 30% negative reactions (dislikes). Only comments with more than 10 reactions in total were included in this chart:

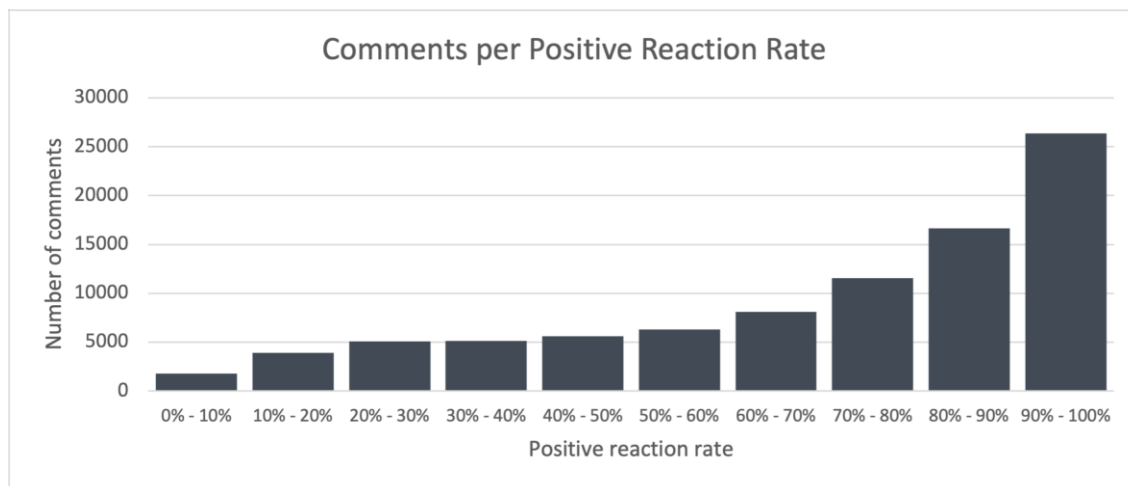


Figure 3.16: Import statistics: comments per positive reaction rate

Most comment authors have a positive average reaction rate. However, a small number of comment authors writes comments that are not popular in the community. Only comments with more than 10 reactions and only comment authors with more than 5 comments were included in this chart:

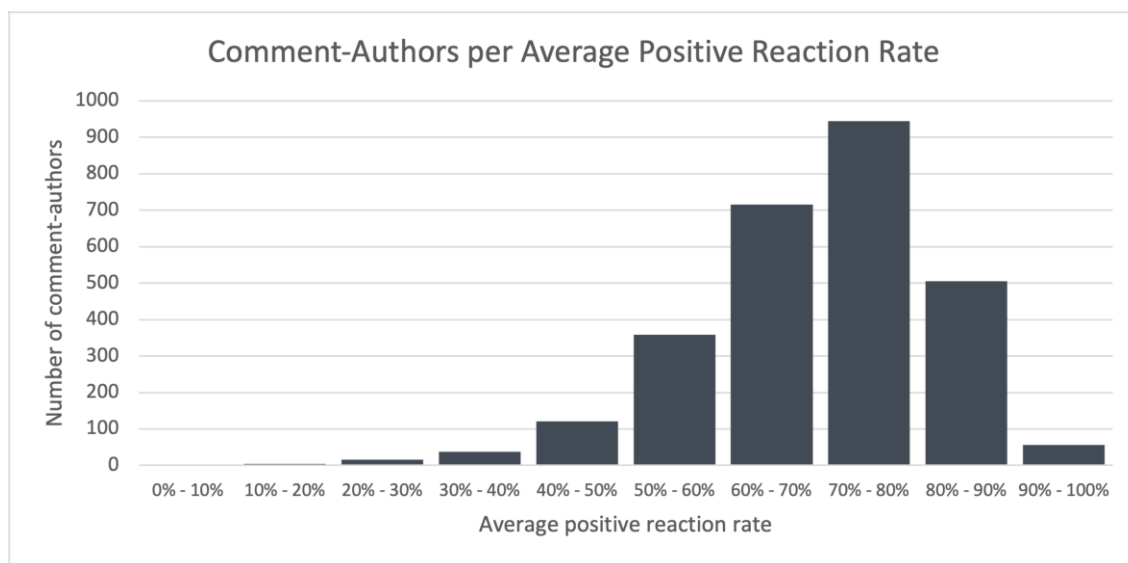


Figure 3.17: Import statistics: average reactions per author

There is a median of 2 comments per author in the dataset. The average value is 7 comments per author. This implies that there are some very active users. The author with the most comments has 921 associated comments in the dataset.

It is interesting to analyze how very active comment authors distribute their activity during a day. Many authors seem to spend a lot of time on the 20 Minuten platform. There are multiple users that are active during 13 or more out of 24 hours in a single day.

The queries that were used to gather this information from the data are included in the code repository.

3.4 Manual Labeling of 20 Minuten Comments

3.4.1 Rationale

In the first weeks of the project, there was no labeled training data for 20 Minuten comments available. Therefore it was decided to manually label some comments to create a first set of training data. This first set of training data allowed to start working on the classifiers while a data delivery of 20 Minuten was still under legal clarification.

In order not to lose too much time for this repetitive task, friends and family members were asked to help with the labeling of the data.

The self-labeled data has the disadvantage that only published (non-rejected) comments were labeled. Still, labeling the data gave some interesting insights about the comments in general. The labeled data helped to create the off-topic classifier described in Section 4.5.

3.4.2 Choice of a Data Labeling Tool

Four tools for manual data labeling were compared. Most of them not only support text-classification but all kinds of labeling problems (audio labeling, video labeling, image labeling, etc.):

- **Label Studio:** Easy setup, easy collaboration, allows to create rich individual user interfaces, limited role and access management in the free version
- **Labelbox:** Free version for most of the functionality, free for education, many features, more complicated setup
- **lighttag:** Free for academic use, not as many annotation types as the other tools
- **ML-Annotate:** Easy user interface, manual setup, limited functionality, project repository no longer maintained

In the end, Label Studio was chosen because of its ease of installation and adequate functionality.

3.4.3 Technical Setup

A Label Studio instance was started on the server for this project. Label Studio is a python application that starts a web server, which can then be publicly accessed².

The comments to be labeled were imported to Label Studio using a `.csv` file. In Label Studio, a custom set of labels and a custom user interface was defined. Invitation links were sent to multiple people, which then labeled each comment and assigned the labels to the comments.

²The setup of the application is explained in detail in the README-File in the GitLab repository: <https://gitlab.ost.ch/ba-troll-detection/troll-candidates-data-labeling>

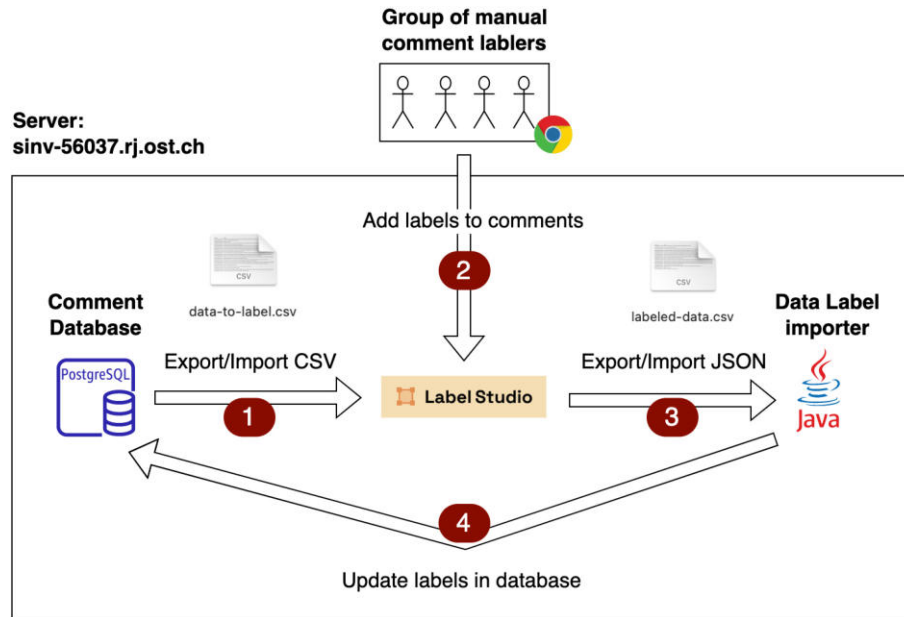


Figure 3.18: Data labeling setup

3.4.4 Choice of comments to label

Comments published between 30/03/2022 and 04/04/2022 were labeled (3'900 in total). Therefore, a query was written to export the comments from the database into the format required as input for Label Studio³.

3.4.5 Choice of labels

In the discussion about which labels to use, attention was paid to two things: On the one hand, the labels should be clear and unambiguous. It should be prevented that different people assess the same comment differently. On the other hand, the labels should be defined as broadly as possible and relate to the three different categories of trolls that are analyzed in this project. This was a balancing act.

The following labels were made available for the comments:

1. **Hate:** Divides society. Stirs up anger / hatred. Heats up conflicts.
2. **Hate:** Insulting / Discriminating
3. **Off-Topic:** Meaningless, Spam
4. **Propaganda:** Possible fake news, unverifiable information, conspiracy theories
5. **Propaganda:** Spreads mistrust towards Western/Swiss authorities, institutions, media... *

* Especially for the last point (mistrust), it should be mentioned that this label also applies to many legitimate comments (not only to state-linked propaganda comments). This label was included because the defamation of institutes is a common strategy of professional trolls.

The people who labeled the data were advised to skip comments when they were in doubt about their decision. They were also given the possibility to mark comments with "second review wanted" if they felt unsure about their choice of labels. The people who labeled the data were also given a document with guidelines and examples that contain more details about when a label should be applied⁴.

³<https://gitlab.ost.ch/ba-troll-detection/troll-candidates-data-labeling>

⁴see <https://gitlab.ost.ch/ba-troll-detection/troll-candidates-data-labeling>

3.4.6 User Interface

This is how the user interface for the labeling of comments looks like:

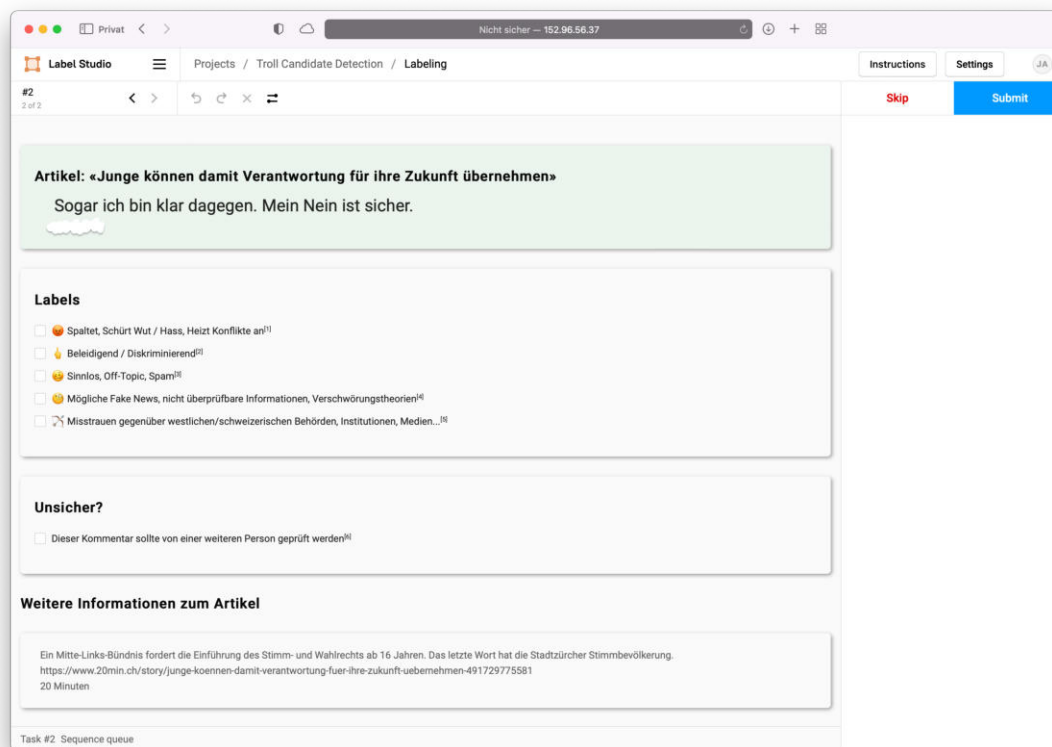


Figure 3.19: User interface of the labeling tool

On the top, the user sees the headline of the current article where the comment occurs. Below the headline, there is the comment itself and the parent-comment of a comment if it exists. At the bottom of the page, additional information about the current article is shown. For each comment, users can select the matching labels and click "submit" to go to the next comment.

The user interface was created using a html-like language.

3.4.7 Labeling Statistics

The manual labeling of the data was done by 6 people. 3'900 comments were labeled in total.

Assigning the comments to the defined categories was often subjective. Some comments are in a gray area. Such controversial comments were marked and double-checked by at least one more person to make sure that more or less the same criteria was applied to all comments. If the classification was not agreed on, the comment was skipped.

During the manual labeling, it was noticeable that there were many more problematic comments on some articles than on others.

Here are the overall statistics per each label:

1. Hate: Divides society. Stirs up anger / hatred. Heats up conflicts: **107 comments (2.7% of the comments)**
2. Hate: Insulting / Discriminating: **69 comments (1.8% of the comments)**
3. Off-Topic: Meaningless, Spam: **172 comments (4.4% of the comments)**
4. Propaganda: Possible fake news, unverifiable information, conspiracy theories: **91 comments (2.3% of the comments)**

5. Propaganda: Spreads mistrust towards Western/Swiss authorities, institutions, media: **182 comments (4.7% of the comments)**

3.4.8 Writing the Labels back into the Database

The `.csv` file exported from Label Studio contains all the metadata about the comment and a list of labels for each comment. This `.csv`-file was then parsed using a custom Java application that writes the labels back into the database⁵.

In the first weeks of the project with no other training data available, those labeled comments were used to set up the classifiers. Later, most of the classifiers were trained again with the data dump that was provided by 20 Minuten. This data had the advantage that also rejected comments are included.

⁵see <https://gitlab.ost.ch/ba-troll-detection/troll-candidates-data-labeling/-/tree/main/labeled-data-to-database-importer-tool>

3.5 Training Data

3.5.1 Data for Machine Learning Training

There are multiple labeled datasets used as training data for different purposes. Each description of the training data contains a reference to the classifier(s) where the training data is used. More specific training data for individual classifiers are described for each classifier in Chapter 4.

3.5.2 Dump of 20 Minuten Comments from 2021

Additionally to the comments that are automatically collected with the 20 Minuten *API*, 20 Minuten provided a large dump of comments:

The dump includes all comments written between January and September 2021, these are more than two million comments in total. The data also includes comments that were not published because they were rejected by the moderation. Each comment has an assigned label, either "ACCEPTED" or "REJECTED".

```
▼ 11: Object
  id: "a434f3d5-68a8-468f-9174-3571a0d88dc9"
  parentId: null
  contentUrl: "https://20min.ch/story/641681468202"
  createdAt: "2021-03-04T09:42:34.274Z"
  tenantId: 6
  contentId: 641681468202
  authorNickname: ██████████
  body: "Niemmert intressierts."
  status: "REJECTED"
  ▼ statusHistory: Array[2]
    ▼ 0: Object
      status: "PREMOD"
      moderatorId: null
      createdAt: "2021-03-04T09:42:34.272Z"
    ▼ 1: Object
      status: "REJECTED"
      moderatorId: "a631941a-956d-4f14-8bce-0912ea2df837"
      createdAt: "2021-03-04T09:42:34.325Z"
```

Figure 3.20: Example of a rejected comment from the 20minuten.ch 2021 dump

Take away for the project: This dataset is used for the Rejected Words Classifier described in Section 4.2, for the Rejection *SBERT* Classifier described in Section A.1.1 and for the Metadata Classifier described in Section 4.6.4. For this purpose, the data was imported into the database as described in Section 3.6.

3.5.3 Russian Troll Tweets by FiveThirtyEight

FiveThirtyEight is an American News Website focussing on data journalism. They obtained and released a dataset containing almost 3 million tweets associated with professional Russian trolls [36]. The dataset was initially created by researchers from the Clemson University. The tweets were posted between 2012 and 2015. The tweets are an example of a coordinated trolling campaign to spread conflict, fear, and disinformation. Each tweet is labeled with one of the following categories: "Right Troll", "Left Troll", "News Feed", "Hashtag Gamer" and "Fear-monger". This dataset also includes metadata about the tweet itself as well as the Twitter-user who posted the tweet. Most of the tweets are written in English, but there are also many tweets in German language. Those tweets, however, were created by a relatively small group of Twitter users.

	author	content	language	publish_date	following	followers	post_type	account_category	tweet_id	article_url
26588	BERKHOFF85	Ein AfD-Politiker im Kreis	German	9/30/2016 9:42	2341	2171		NonEnglish	7.81791250273988608	
26589	BERKHOFF85	Guten Morgen #Berlin. Guten	German	9/30/2016 6:12	2341	2170		NonEnglish	7.81739E+17	http://twitter.com/berkhoff85/statuses/781738521455460352
26590	BERKHOFF85	#Union's-Politiker wollen mel	German	9/30/2016 13:23	2340	2171		NonEnglish	7.81847E+17	http://twitter.com/berkhoff85/statuses/781847044747042817
26591	BERKHOFF85	#Athen will #VfVchtlinge auf	German	9/28/2016 9:04	2342	2166		NonEnglish	7.81057E+17	http://twitter.com/berkhoff85/statuses/781057089821171712
26592	BERKHOFF85	#Dresden: Bekennerschreiben	German	9/28/2016 7:42	2342	2168		NonEnglish	7.81036E+17	http://twitter.com/berkhoff85/statuses/781036408731467776
26593	BERKHOFF85	Guten dunstiger Mystik-Morg	German	9/28/2016 6:35	2342	2166		NonEnglish	7.81019E+17	http://twitter.com/berkhoff85/statuses/781019360827703299
26594	BERKHOFF85	Finanzministerium weist Beri	German	9/28/2016 16:21	2342	2168		NonEnglish	7.81167E+17	http://twitter.com/berkhoff85/statuses/781167054170759168
26597	BERKHOFF85	Der #EU-#Präsident Deal ist sow	German	9/28/2016 14:20	2342	2168		NonEnglish	7.81136E+17	http://twitter.com/berkhoff85/statuses/781136386535460864
26700	BERKHOFF85	Was #Frankreich und #Deutsch	German	9/28/2016 13:19	2342	2168		NonEnglish	7.81121E+17	http://twitter.com/berkhoff85/statuses/781121031943495680
26701	BERKHOFF85	Was ist hier falsch? Wie mei	German	9/28/2016 13:00	2342	2168	RETWEET	NonEnglish	7.81116E+17	http://twitter.com/berkhoff85/statuses/781116469413343232
26702	BERKHOFF85	der vBer der #Ukraine abges	German	9/28/2016 12:20	2342	2168		NonEnglish	7.81106E+17	http://twitter.com/berkhoff85/statuses/781106234816655362
26703	BERKHOFF85	ÖaÖ Die Polizei hat fVfnt mut	German	9/28/2016 11:09	2342	2168		NonEnglish	7.81089E+17	http://twitter.com/berkhoff85/statuses/781088556643213312
26704	BERKHOFF85	Ich weivü, du lebst in deiner	German	9/27/2016 9:16	2342	2167		NonEnglish	7.80698E+17	http://twitter.com/berkhoff85/statuses/780697589737988096
26705	BERKHOFF85	Nur #Bayern und #BW schreibt	German	9/27/2016 8:38	2342	2167		NonEnglish	7.80688E+17	http://twitter.com/berkhoff85/statuses/780688151160209408
26706	BERKHOFF85	Morgen #Deutschland... 2 Spr	German	9/27/2016 7:16	2342	2166		NonEnglish	7.80667E+17	http://twitter.com/berkhoff85/statuses/780667354783121408
26707	BERKHOFF85	Schönen Abend liebe Freund	German	9/27/2016 17:12	2344	2170		NonEnglish	7.80817E+17	http://twitter.com/berkhoff85/statuses/780817278426316801
26708	BERKHOFF85	ein Junge meint, daß er geh	German	9/27/2016 15:55	2344	2170		NonEnglish	7.80798E+17	http://twitter.com/berkhoff85/statuses/780797902855602176
26709	BERKHOFF85	Schweizer Parlament hat fVr	German	9/27/2016 14:57	2344	2168		NonEnglish	7.80783E+17	http://twitter.com/berkhoff85/statuses/780783336780361730
26710	BERKHOFF85	#Aleppo erlebt nach dem End	German	9/27/2016 14:05	2344	2168		NonEnglish	7.80777E+17	http://twitter.com/berkhoff85/statuses/780777020306632706
26711	BERKHOFF85	Die #Mittagspause war toll!!!	German	9/27/2016 12:57	2344	2168		NonEnglish	7.80753E+17	http://twitter.com/berkhoff85/statuses/780753262169427968
26712	BERKHOFF85	Der #Vreze #Islam-Witz de	German	9/27/2016 12:28	2344	2169	RETWEET	NonEnglish	7.80746E+17	http://twitter.com/berkhoff85/statuses/78074645947961823232
26713	BERKHOFF85	#Assad's Armee startet Bod	German	9/27/2016 11:48	2344	2168		NonEnglish	7.80736E+17	http://twitter.com/berkhoff85/statuses/780736386365757632
26717	BERKHOFF85	Warum wirks #FVchtlingskri	German	9/27/2016 10:23	2344	2169		NonEnglish	7.80715E+17	http://twitter.com/berkhoff85/statuses/780715459004909664
26718	BECKRALFBECK265	Japaner versuchte Auto voll	German	9/26/2017 9:06	91	82		NonEnglish	9.12604E+17	http://twitter.com/75285762085037312/statuses/91260419868824780
26719	BECKRALFBECK265	Was ist eigentlich Btsch? Ist	German	9/26/2017 9:06	91	83		NonEnglish	9.12604E+17	http://twitter.com/75285762085037312/statuses/91260419868824780

Figure 3.21: Excerpt from FiveThirtyEight's troll dataset

Take away for the project: This dataset contains only troll-tweets and not non-troll tweets. It has to be considered that the language used in the tweets might be quite different from the language used in the newspaper comment sections, which might lead to weaker performance of the algorithms in practical use. This training data set was used for the State-Linked Classifier, described in Section 4.4.

3.5.4 One Million Posts Corpus

The project "One Million Posts Corpus" published an SQLite database with German comments from the Austrian newspaper "der Standard" [37].

The dataset contains about 10'000 labeled and 1'000'000 unlabeled posts. Each entry contains the comment-body, a user identifier, a time stamp, an excerpt from the article, the status of the comment (online or deleted) as well as the reactions from other users.

The following labels are interesting for this project:

- Sentiment (negative/neutral/positive)
- Off-Topic (yes/no)
- Inappropriate (yes/no)
- Discriminating (yes/no)
- Arguments Used (yes/no)

Take away for the project: The data provided almost has the same data structure as the structure defined in Section 3.2.2. This dataset is used for the Metadata Classifier described in

Section 4.6.3, the Off-Topic Classifier described in Section 4.5 and the Hate Classifier described in Section 4.3.5. For this purpose, the data was imported into the database as described in Section 3.7.

3.5.5 Rheinische Post: Moderator- and Crowd-Annotated German News Comment Datasets

This dataset contains comments from German online discussions [38]. A crowd of people was used to label the comments on whether they contain abusive language or not. Abusive language was categorized as: Sexism, Racism, Threat, Insult, and Profanity. In this thesis, such comments are described as "Hate comments". The dataset not only contains public comments, but also comments that were refused by moderators.

Unnamed: 0 id	Text	Reject Newspaper	Reject Crowd	Rejection Count Crowd	Sexism Count Crowd	Racism Count Crowd	Threat Count Crowd	Insult Count Crowd	Profanity Count Crowd	Meta Count Crowd	Advertisement Count Crowd
0	1911223 Niemand braucht Läschet den Merkel GvPh	0	0	0	0	0	0	0	0	0	0
1	1911225 das war apokalypse now. nicht einmal zu e	0	0	0	0	0	0	0	0	0	0
2	1911229 Katastrophal - Katastrophal - anders kan	0	0	0	1	0	0	0	0	1	0
3	1911239 Dams sollten wir unsere RVPlustungsexporte	0	0	0	0	0	0	0	0	0	0
4	1911243 na ja im notfall sind wir amis ja noch da u	0	0	0	0	0	0	0	0	0	0
5	1911245 ihr schafft das schon	0	0	0	0	0	0	0	0	0	0
6	1911251 Es ist gut, dass BRD-Politiker nichts wirklic	0	1	4	0	0	0	2	1	2	0
7	1911253 Und wieder mal ist die sprechende FvöHnf	0	0	2	1	0	0	1	0	0	0
8	1911255 Es gibt ein Sprichwort was sagt: Glaube ke	0	0	0	0	0	0	0	0	0	0
9	1911261 Da kann man mal sehen. Gedanklich ist die	0	0	0	0	0	0	0	0	0	0
10	1911273 was nun, ihr HvöHnenflieger?? Jetzt geht di	0	0	0	0	0	0	0	0	0	0
11	1911275 Er wird frVhsterns einen Tag nach dem C	0	0	0	0	0	0	0	0	1	0
12	1911277 Felix Austria.	0	0	1	0	0	0	0	0	0	0
13	1911293 FreVher mussten Politiker NVHr Kleinste Kle	0	0	1	0	0	0	1	0	0	0
14	1911295 Da sind sie wieder, die Geister der Bayern	0	0	0	0	0	0	0	0	0	0
15	1911297 Hat nicht schon mal eine Partei einem Kan	0	0	0	0	0	0	0	0	0	0
16	1911299 Lt Hecking (Zitat MONNA) hat die Borussia	0	0	0	0	0	0	0	0	0	0
17	1911309 Also wieder eine Mvögllichkeit Vettermeirt	0	0	0	0	0	0	0	0	0	0

Figure 3.22: Excerpt from crowd-voted comments

Take away for the project: Not every troll uses abusive language. However, abusive language is used by hater trolls. This training data set is used for the hate classifier described in Section 4.3.5

3.6 Import of '20 Minuten' Comments

3.6.1 Included Information

The dataset from the year 2021 was provided by 20 Minuten as a folder containing over 500 individual **JSON**-files. Each **JSON**-file contained 5'000 comments with additional metadata (article *URL*, creation date, author nickname, status, parent comment...). To make the data easier to handle and query, it was decided to store those comments into the database as well.

3.6.2 Import Procedure

To import the comments into the database, a Java-Application was written⁶. The application parses each **JSON**-file, converts the comments into the expected structure of the entities (see Section 3.2.2) and stores all comments into the database.

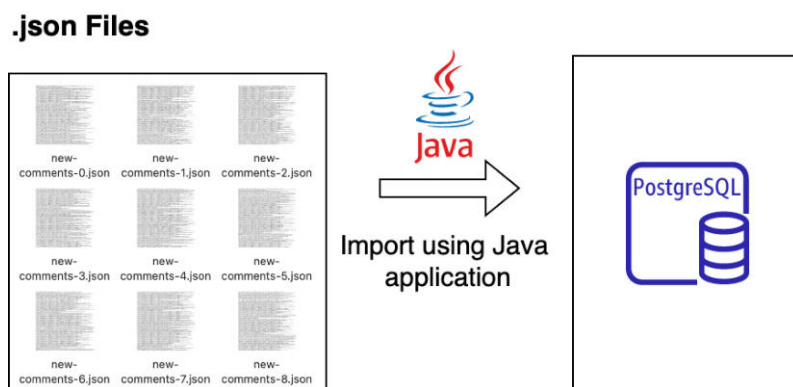


Figure 3.23: Transformation and import of the 20minuten.ch 2021 dump

3.6.3 Changes made to the data

Before the import, a few changes were made to the data. First, it was checked if each reply comment references a valid parent comment. This was not the case for approximately 400 comments. Those comments were excluded from the import because they would violate the database constraints.

The dataset also includes French comments from the French version "20 Minutes". Unfortunately, the data did not contain a label about the language of a comment. Comments that explicitly referenced articles for the French version were excluded from the import. This step removed most of the comments in the French language, but not all of them.

3.6.4 Result

In total, about 560'000 rejected comments and about 1'265'000 accepted comments were imported into the database.

⁶<https://gitlab.ost.ch/ba-troll-detection/20-minuten-json-to-database>

3.7 Import of 'Der Standard' Comments

3.7.1 Included Information

This trainingdata set contains one million comments from the Austrian Newspaper "der Standard" (see Section 3.5.4). About 10'000 comments are labeled with different categories [37].

3.7.2 Relevance of the Dataset

The reason that not only 20 Minuten comments were imported into the database but also comments from "der Standard" was that "der Standard" provided additional metadata for each comment (additional information about the comment authors, article, and reactions). Therefore, those comments were better suited to build a metadata classifier (see Section 4.6.3). The dataset is available as an SQLite file.

3.7.3 Import Procedure

Pleasingly, the SQLite file almost has the same structure as the database structure which is used for the import of the 20 Minuten comments (see Section 3.2.2). To import this dataset to the existing database, the data was first transformed using *SQL* queries. The output of those queries was exported as a separate *.csv*-file for each table. The *.csv*-files were then imported into the existing database⁷.

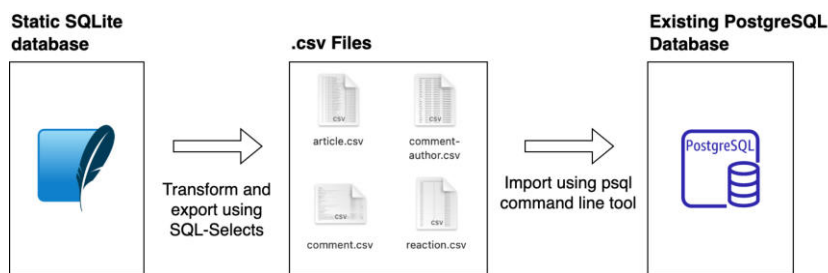


Figure 3.24: Transformation and import of the one million posts corpus project database

3.7.4 Changes made to the data

Some comments referenced non-existing (deleted) comments. Those comments violate the foreign-key constraint from the database. Those comments (1'100 in total) were therefore removed from the dataset.

The original labeled data contains a total of 9 labels. Those labels were conflated to only two labels: "troll" and "non-troll". All comments labeled as "Off-Topic", "Inappropriate" and "Discriminating" were mapped to the "troll" label. Labeled comments with non of those categories were mapped to the "non-troll" label.

3.7.5 Result

In total, 1000 troll comments, 8000 non-troll comments and the corresponding comment-authors, articles, and reactions were imported into the database.

The fact that the training data now has the same structure as the existing data makes the upcoming work easier. *SQL*-queries can be reused to work with both datasets – data from 20 Minuten and data from the One million posts corpus project.

⁷see <https://gitlab.ost.ch/ba-troll-detection/one-million-posts-corpus-to-database>

CHAPTER 4

Data Modeling and Evaluation

This chapter focuses on the classification strategies to identify trolls. After a brief overview, the individual classifiers are explained in more detail.

4.1 Overall Classification Strategy

4.1.1 Big Picture

Inappropriate comments can be detected using different characteristics. Some features are based on the metadata of a comment (author, date and time of publication, likes and dislikes...), while other features are based on the comment's content.

In the following sections, multiple classifiers are presented. The classifiers are trained to detect different categories of inappropriate comments:

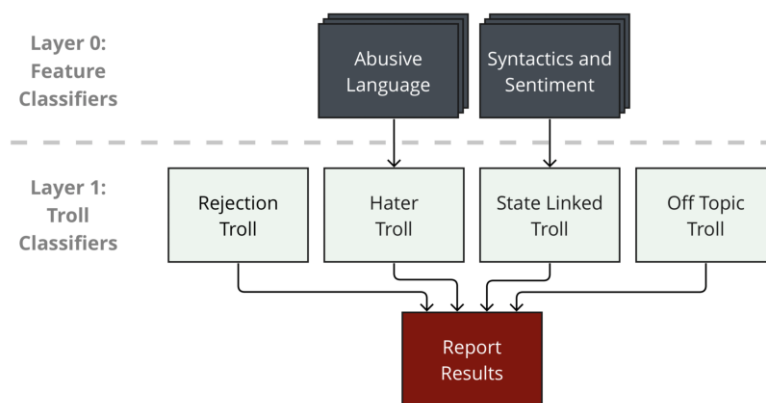


Figure 4.1: Classifiers big picture

4.1.2 Choice of the Troll Categories

The project differentiates between "off-topic trolls", "hater trolls" and "state-linked trolls". To select those categories, the commenting-guidelines for 20 Minuten, Tagesanzeiger, Watson, Blick and the Neue Zürcher Zeitung were analyzed. The goal was to define as few categories as possible, while at the same time defining categories that can describe most of the unwanted comments.

- The category "hater troll" includes comments that are aggressive, anti-social, disrespectful, discriminatory, harassing, insulting, offending, offensive, etc.
- The category "off-topic troll" includes comments that are not relevant to the topic, comments that are spam and comments that do not contribute constructively to the discussion.
- State-linked propaganda trolls are more difficult to detect. Still, this category was included because state-linked trolls often do not fall into the categories "hater troll" or "off-topic troll".
- The category "rejection troll" includes all comments that were rejected by the (manual) moderation. This category usually overlaps with the other categories.

4.2 Rejection Words Classifier

4.2.1 Overview

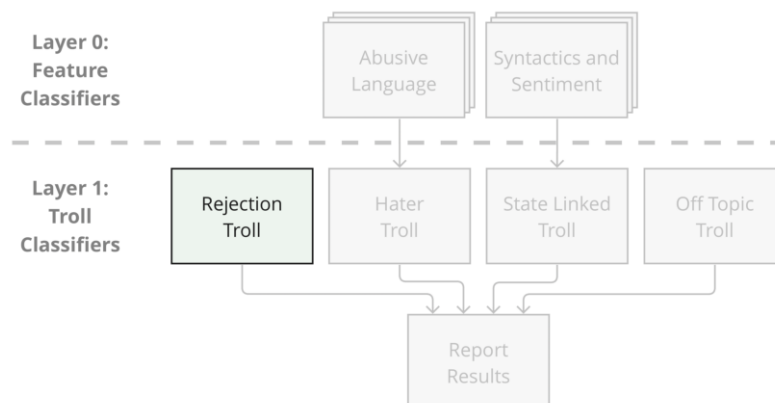


Figure 4.2: Rejection troll overview

4.2.2 Idea

The classifier described in this section predicts the probability for a 20 Minuten comment to be rejected by the moderation.

There are many reasons why a comment might be rejected. The reasons include hate speech, spam, inclusion of links and more¹. Thanks to the 2 million labeled comments from 20 Minuten, there is a lot of training data available for rejected and non-rejected comments (see Section 3.5.2).

Before using more advanced methods like *BERT* or lemmatization, a rather simple approach was chosen for this classifier: Simply put, "bad" and "good" words were extracted from the dataset from 20 Minuten. In this context, a word is called "bad" if comments containing this word are often rejected. The idea of this approach to automatically detect comments that are rejected because they contain offensive terms or curse words.

4.2.3 Other Rejection Classifiers

More advanced methods were also tested to predict a possible rejection but this did not improve the performance of the classifier. Those discarded classifiers are described in Section A.1.

4.2.4 Similar Approaches from the Literature

In similar work, "bad words" have often been used as a feature to detect troll comments [1]:

- The Metadata Troll Detector from Stephan Dollberg (described in Chapter 4.6.2) used a "Bad Word Count" as one of several features. He used a predefined "list of bad words", which is not described in detail in his thesis. In the project, which worked with Reddit comments, the feature however did not play a significant role. A classifier trained with only this single feature achieved an accuracy of 52% [29].
- The study "Hunting for Troll Comments in News Community Forums" (described in Chapter 4.6.2) also used the "Number of bad words" as a feature for their classifier. A list of 458 bad words was translated from English to Bulgarian using Google Translate. Similar words were grouped using a word2vec model. When they used this feature as the only feature for the classifier, it achieved an accuracy of 64% or 60% (depending on the data source) to detect troll comments [39].
- Instead of the number of bad words, some similar studies like "Content based approach to find the credibility of user in social network" also work with the density of bad words [40].

¹see commenting guidelines from 20 Minuten: <https://www.20min.ch/story/die-kommentarrichtlinien-von-20-minuten-119025471145>

To summarize the literature studied, the performance of these approaches was rather mixed. However, a massively larger amount of training data is available for this project. Furthermore, the data does not have to be translated first which is an advantage as well. The goal is therefore to achieve accuracy, recall, and precision of at least 65% each.

It should be mentioned that not every troll comment necessarily contains "bad words / curse words". Especially professional trolls often try to remain unrecognized and do not violate the guidelines. Therefore, they avoid using curse words in their comments.

4.2.5 Data Understanding and Data Preparation

Instead of using a predefined list of offensive or harmful German words, the comments provided by 20 Minuten were used to extract words that were previously used in comments. This dataset is described in Section 3.5.2. 75% of the nearly 2 million comments available were used to extract a list of words. The training- and test-data was taken from the remaining 25% of the comments that were not used to create the list of words.

In total, 824'163 different words were extracted from the existing comments using SQL. Those words are normalized: all words were converted to lower case and all characters that do not originate from the Swiss-German alphabet were removed.

Subsequently, the following two values were calculated for each word:

- **Rejection Rate:** The percentage of rejected comments containing this words in relation to all comments containing this word.
- **Occurrences:** The number of comments in the dataset that contain this word.

The result is a 25 MB file with all used words and the two calculated values. Here is an example excerpt:

```

1 word,rejectionrate,occurrences
2
3 "corona",0.372260254271415,52149
4 "coronaaktivisten",0.454545454545455,11
5 "coronaangsthasen",0.428571428571429,7
6 "coronaansteckung",0.454545454545455,22
7 "coronaapp",0.266666666666667,30
8 "coronaauflagen",0.181818181818182,11
9 "coronabedingt",0.361702127659574,47
10 "coronadiktatur",0.690265486725664,113
11 "coronahotspot",0.277777777777778,18
12 "coronahysteriker",0.666666666666667,33
13 "coronainfektion",0.276018099547511,221
14 "coronamassnahmengegner",0.375,16
15 "coronapolitik",0.395939086294416,197
16 "coronavorschriften",0.222222222222222,18

```

Listing 4.1: Excerpt of the list of words

The classifier can analyze the individual words in a comment using this list. This approach is different compared to the existing classifiers from other projects described above. They use a list of bad words with no additional information for each word. In contrast, the approach described here assigns each word a "score / rejection rate". By specifying the number of occurrences, it is prevented that rarely used words get too much influence on the model.

Since the list is compiled from existing comments, it was made sure that the classifier built upon this list was trained and tested with unseen comments that were not used to compile this list of words.

4.2.6 Feature Engineering

The following features are extracted for a single comment:

- Maximum rejection rate of a word in the comment with occurrence of at least X (10 different features in total)

- Number of words in the comment with a rejection rate greater than X and occurrence greater than Y (160 features in total)
- Proportion of words in the comment with rejection rate between X and Y and occurrence of at least Z (100 features in total)
- Percentage of words that are not present in the list of words (never-seen-words).

The features chosen for this classifier are more extensive than the features used in similar studies mentioned above. While some analyzed projects use "amount of bad words in a comment" as the only feature, additional experimental / new features are used for the classifier described in this section.

The calculated features were scaled using Python's `StandardScaler`.

4.2.7 Modeling

Based on these features, different models were trained. It was found that a logistic regression classifier achieved the highest overall performance:

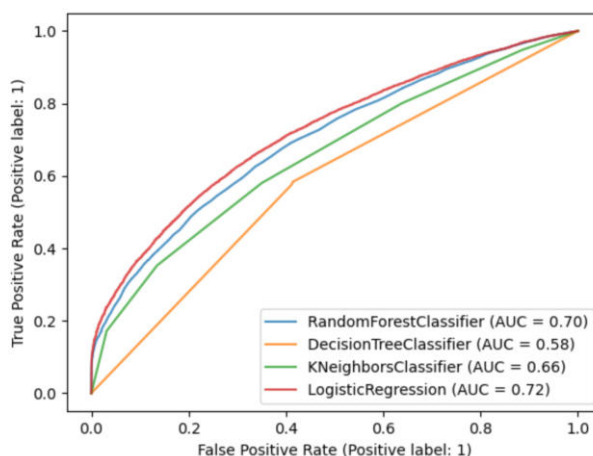


Figure 4.3: ROC curve for different rejection classifiers

Thus, a logistic regression classifier was selected for hyperparameter optimization. It is not surprising that a simple logistic regression model achieved the best results. The relationship between the features and the label "rejected" or "accepted" is simple: A comment that contains words with a high rejection-score has a higher probability of being rejected. This correlation is expressed well with a logistic regression classifier.

A balanced dataset with 5'000 rejected comments and 5'000 non-rejected comments was used to train and test the classifier. Those comments were randomly selected. Only comments that were not used to compile the list of words were used as training data and test data. Out of those 10'000 comments, 7'000 were used as training data (70%) and 3'000 as test data (30%)

4.2.8 Optimization

The first step was to investigate which features had a particularly strong influence on the model. The following features were the most important for the model:

- Maximum rejection rate of a word in the comment with occurrence of at least 100
- Number of words in the comment with a rejection rate greater than 0.3 and occurrence greater than 1
- Maximum rejection rate of a word in the comment with occurrence of at least 8
- Maximum rejection rate of a word in the comment with occurrence of at least 1

The features with the least impact were:

- Percentage of words that are not present in the list of words
- Proportion of words in the comment with rejection rate between 0 and 0.1 and occurrence of at least 100 (note: such words are almost inexistent)
- Number of words in the comment with a rejection rate greater than 0.8 and occurrence greater than 512

A manual feature selection was performed in an attempt to further improve the model. However, it became apparent that the removal of less relevant features slightly worsens the model. For this reason, all features were used in the final model.

A grid search was used to test various hyperparameters for the model. The model is regularized with the L2 norm, and with a regularization term of 1e-8.

4.2.9 Evaluation

The classifier achieved a recall of 62% and a precision of 76% on a validation set which was neither used to build the list of words nor to train the classifier.

Table 4.1: Scores for the rejection words classifier

Metric	Score
precision	76%
recall	62%
f1-score	68%
accuracy	71%

The classifier can correctly predict non-rejected comments as such in 80% of the cases and troll comments as such in 62% of the cases:

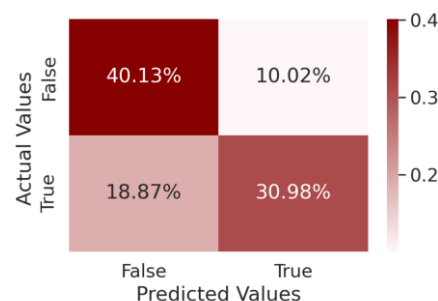


Figure 4.4: Confusion matrix for the rejection classifier (20 Minuten)

The target of 65% accuracy and precision was exceeded. The recall is slightly below 65% for the "troll" label but 80% for the "non-troll" label. This means that the classifier is better in detecting non-trolls than trolls. It is noticeable that the *ROC* curve rises sharply at the beginning: With this model, it is possible to detect about 20% of the troll comments with almost no false positives. 20 Minuten could therefore use such a classifier to make a first automated pre-selection of the comments.

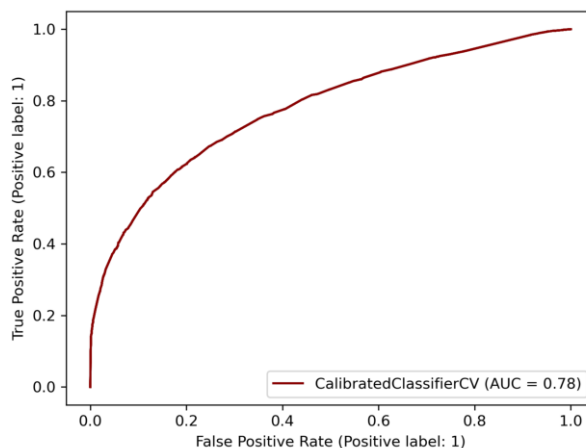


Figure 4.5: ROC curve for the rejection classifier

4.2.10 More meaningful probabilities

The probabilities returned by the classifier were not really meaningful. For a logistic regression classifier, Sklearn does not return a true probability, but rather a "score" [41]. In this example, the score was usually in a range between 0.499 and 0.501. To have scores that are easier to interpret in the user interface, the scores of the model were recalibrated to make them look like real probabilities (platt scaling).

4.2.11 Examples

Here are some example comments that were analyzed with the rejection classifier:

Table 4.2: Rejection words classifier: test comments

Comment	Pred. Prob.	Pred. Class	Actual Class
Armselige Drecks Scheiss Bullen. Wo sie abzocken können sind sie da.	99.7%	Rejected	Rejected
Die Leute von [...] haben Null Ahnung... arbeiten bei euch eigentlich nur Vollposten und Idioten?	99.6%	Rejected	Rejected
Selber schuld. Nach McDonalds gehen und Qualität erwarten?!?! Bitte !!	33.8%	Accepted	Rejected
Die Ignoranz der weltweiten Klimaveränderung ist noch viel schlimmer als das momentan viel zu feuchte Wetter. Solange alle Bäuche satt sind stört das aber leider niemanden.	49.8%	Accepted	Accepted
95% der Kommentare hier sind eine Anklage an unser Bildungssystem!	60.9%	Rejected	Accepted

4.2.12 Limitations

This classifier is strong to detect comments that use abusive language but weaker to detect more subtle inappropriate content.

The approach of this classifier is rather experimental. Since existing comments from 20 Minuten were used to compile a list of words, it is assumed that this classifier might have a weaker performance if it is used in a different context (for example for a different newspaper).

4.2.13 Conclusion and Limitations

Compared to the literature, the model performed surprisingly well. It is assumed that this is due to the large amount of training data and the experimental features used.

In a future project, this classifier could be extended even further. For example, other projects have worked with the Levenshtein distance to detect obfuscated bad words [42]. In addition, other techniques such as stemming could be applied to the words.

The created word-list could be interesting for other projects as well: If the list is sorted by the "Rejection Rate" in descending order, and only often-used words are shown, the result is a list of well-known German curse words. Such a list can be useful since it seems that there are only much smaller "German bad word lists" available online.

4.3 Hate Troll Classifiers

4.3.1 Overview

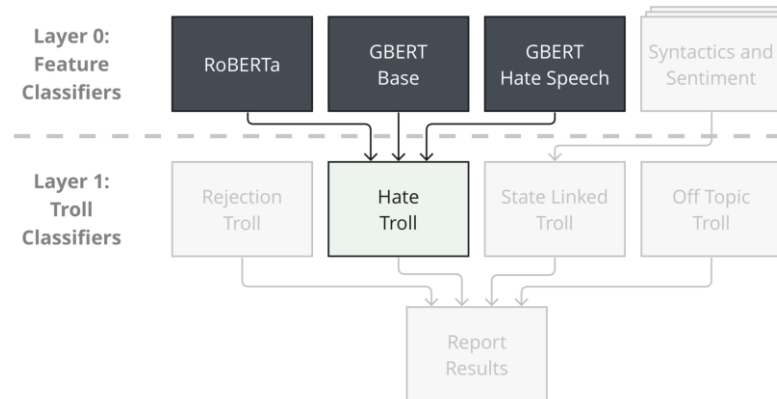


Figure 4.6: Hate troll overview

Hate Speech

Hate speech is an essential part of many troll comments. Hate speech is used to attack an individual directly or to spew hatred in a community. The impact of hate speech can be felt directly or indirectly. In an indirect attack, the initial hate speech can lead to a spiral which encourages others to participate and escalate the situation [43].

Hate speech is defined by the Committee of Ministers of the European Council as follows [44]:

“all forms of expression which spread, incite, promote or justify racial hatred, xenophobia, anti-semitism or other forms of hatred based on intolerance, including intolerance expressed by aggressive nationalism and ethnocentrism, discrimination and hostility against minorities, migrants, and people of immigrant origin“

(Committee of Ministers of the European Council)

Data Acquisition and Understanding

To develop different hate speech classifiers, two different datasets were used as training data:

Rheinische Post This dataset is provided by Dennis Assenmacher et al. In their work, they collected and annotated the largest German (news) comment dataset. The data was acquired through a collaboration with the German newspaper Rheinische Post. The data was subsequently labeled by experts and crowd workers [45]. As described in the original paper, the subset "RP-Crowd-3" contains comments where at least three people agreed on the label. This subset produced the best classifier results in their research. A detailed description of this training data set can be found in Section 3.5.5.

The labels given to each comment are as follows:

Table 4.3: Labels for the dataset (hate classifier)

Label	Explanation
Sexism	Attacks on gender identity
Racism	Attacks on someone's origin, ethnicity, nationality
Threats	Stating violent actions
Insults	Denigrating or disrespectful statements
Profane language	Sexually explicit and inappropriate language
Meta	Comments providing no contribution
Advertisement	Advertising unrelated services

Some of the labels listed fall into the category of hate speech.

One million post corpus from Der Standard This dataset contains comments from the Austrian newspaper Der Standard. The dataset was reduced to 1170 comments containing hate speech. A detailed description of this training dataset can be found in Section 3.5.4.

Combination The datasets from Rheinische Post and Der Standard were combined. The combination of the two sets resulted in the distribution shown in Figure 4.7. Label 1.0 stands for hate speech and 0.0 for non-hate speech.

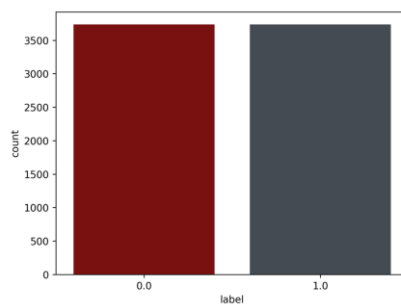


Figure 4.7: Combined dataset label count

Classifier Overview

Assenmacher et al. fine-tuned three pre-trained *BERT* models for 100 epochs each. A description of *BERT* can be found in the Appendix G.6.1.

A similar approach was used to create the following three classifiers described in this thesis. The dataset was split into a train-set, test-set and validation-set. The train-set contains 80% of the dataset. The remaining 20% are split into a test-set and the validation-set. The comments were selected randomly.

Figure 4.8 illustrates the architecture of the classifiers. Each classifier that follows has the same structure.



Figure 4.8: Structure of the hate classifier

Hypothesis

1. The fine-tuned language models separate the data better than the un-tuned ones.

This hypothesis is validated in the following sections.

Unsuccessful Approaches

Several methods were tried with lackluster results. The first one was using *TF-IDF* and different machine learning models such as random forest and *SVM*, the result being an *AUC* of 0.70 for logistic regression.

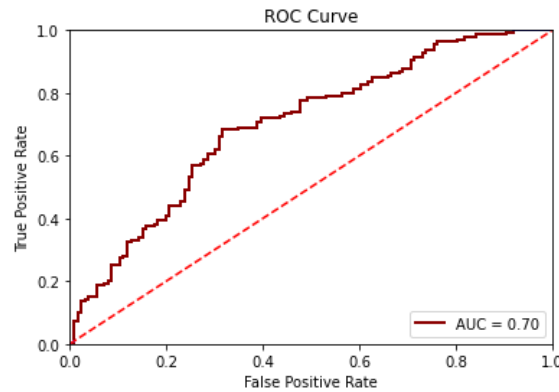


Figure 4.9: Logistic Regression *ROC*

Next, a guide from Google for text classification was followed², which used a convolutional neural network with fast text encoding. The result was a training and validation accuracy around 69%.



Figure 4.10: Validation and training accuracy of unused classifier

²<https://developers.google.com/machine-learning/guides/text-classification>

A method proposed by Saima Sadiq et. al. [46] was tested as well. The classifier was built as closely as possible to the original with the difference that another data set was used. The reported accuracy of 92% and recall of 90% could not be replicated as the model overfitted quite severely. The model contains 3 dense layers with 64 units. Dropout as a regularization method was tried. But the premise of using 30'000 features in a densely connected neural network was prone to overfit.

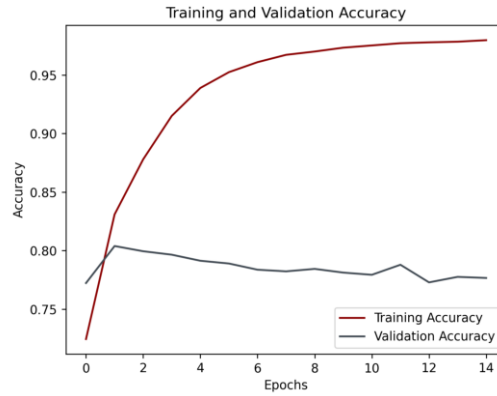


Figure 4.11: Validation and training accuracy of unused classifier (2)

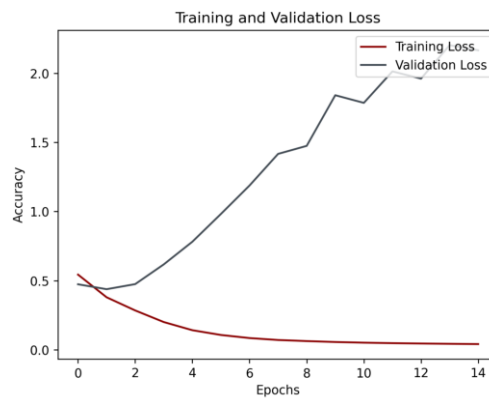


Figure 4.12: Validation and training loss of unused classifier

4.3.2 RoBERTa Classifier

Overview

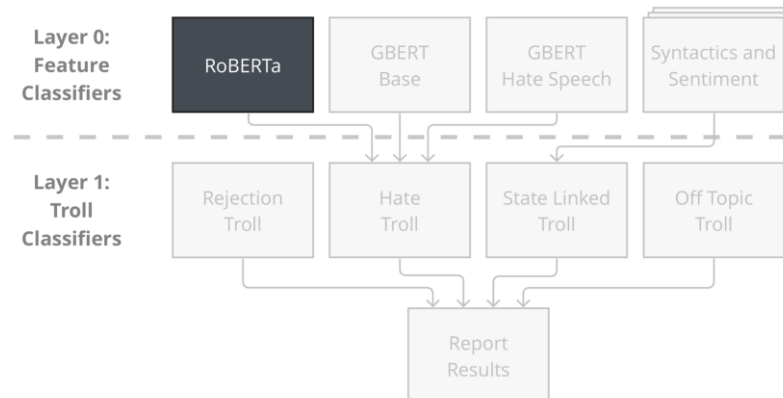


Figure 4.13: Hate RoBERTa classifier overview

Trained Embeddings

To fine-tune the transformer, "Cross English & German RoBERTa for Sentence Embeddings" was chosen since it performs slightly better than the German-only variant [47]. It was then trained for 100 epochs. The encoding of the trained transformer was reduced to 2 dimensions using *PCA* to visualize the results. 1.0 indicates that the data point is hate speech and 0.0 means it is not hate speech.

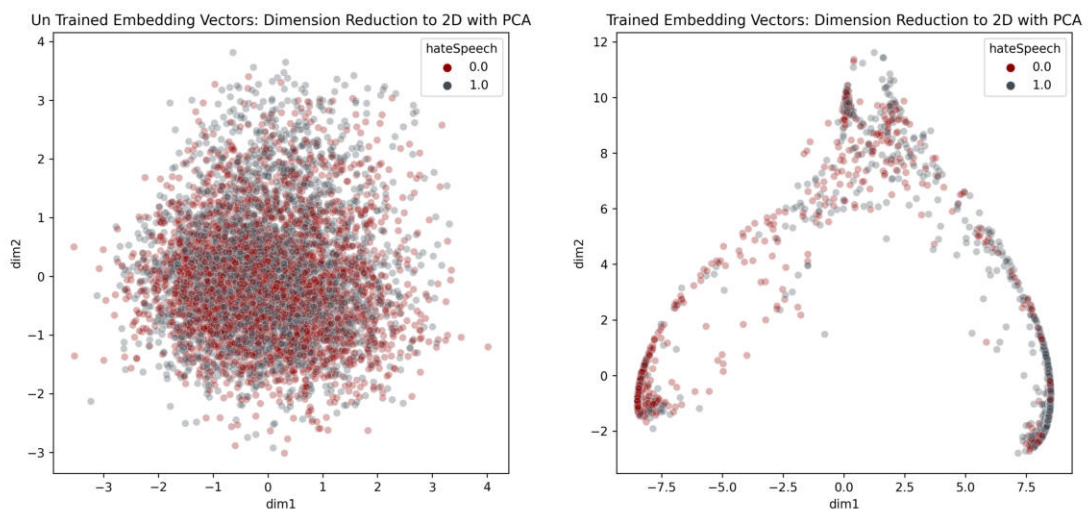


Figure 4.14: Scatterplot of two dimensions reduced by PCA on train-set

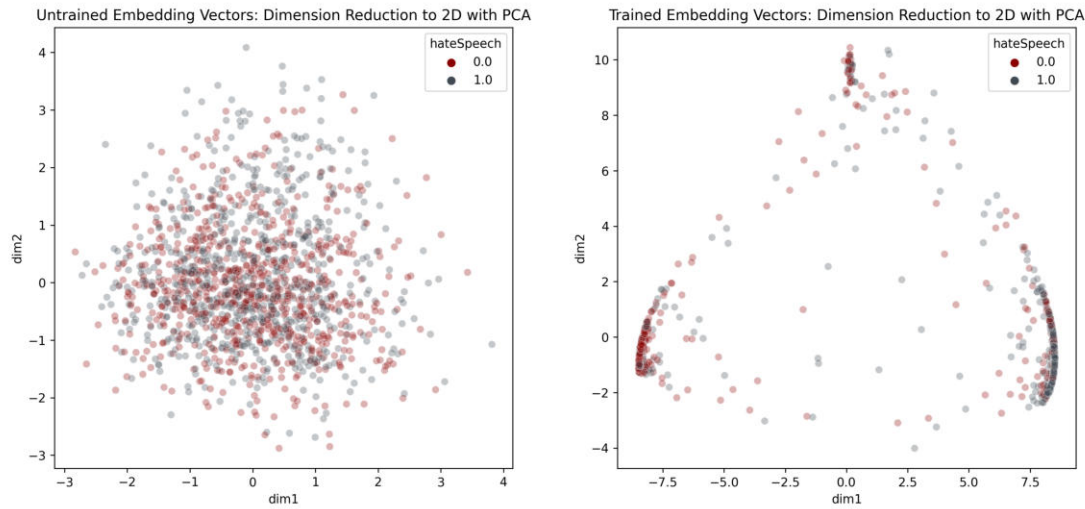


Figure 4.15: Scatterplot of two dimensions reduced by PCA on test-set

Figure 4.14 shows that the trained transformer separates the data well, even on the unseen test-set, as shown in Figure 4.15. The explained variance for the reduction to two dimensions is 6.38% for the untrained and 90.54% for the trained transformer. Even in higher dimensions, the explained variance of the untrained transformer remains low.

Principal Component Selection

To select the optimal number of embedding vectors, a Scree plot was created 4.16. The result was that 4 features are enough, as the remaining features do not increase the explained variance to a substantial margin. The 4 principal components achieve an explained variance of 99.38%.

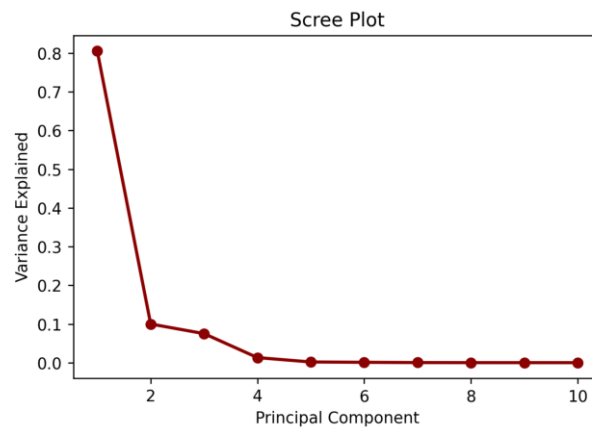


Figure 4.16: Scree Plot of the reduced train-set

Model Selection

Multiple classifiers were tested to decide which to choose. The classifiers were tuned for the best hyperparameters using random search. The tuned hyperparameters were then used to evaluate the different methods.

Table 4.4: RoBERTa: hyperparameter tuning results

Classifiers	Best Hyperparameters	Accuracy	Recall
<i>SVM</i>	kernel: <i>linear</i> , C: 500, probability: <i>True</i>	0.77	0.80
Random Forest	n_estimators: 200, min_samples_split: 10, min_samples_leaf: 2, max_features: <i>log2</i> , max_depth: 50	0.77	0.79
<i>KNN</i>	n_neighbors: 25	0.77	0.78
AdaBoost	learning_rate: 0.0001, n_estimators: 10	0.77	0.78
LDA	solver: <i>svd</i> , store_covariance: <i>True</i>	0.77	0.79
MLP	activation: <i>tanh</i> , alpha: 0.0, learning_rate: <i>constant</i> , hidden_layer_sizes: (50, 50, 50), max_iter: 10000, solver: <i>sgd</i>	0.77	0.79
<i>QDA</i>	store_covariance: <i>True</i> , reg_param: 0.4	0.77	0.79
Logistic Regression Linear	penalty: <i>l2</i> , C: 0.006700187503509591	0.77	0.79

The *SVM* has a slightly higher recall score than the others and was therefore chosen for this classifier.

Evaluation

The trained classifier achieved a recall of 82% on the validation set, which was locked away and not used during the creation of the model. The metrics are listed here:

Table 4.5: Hate classifier: *SVM* scores

Metric	Score
precision	77%
recall	82%
f1-score	79%
accuracy	78%

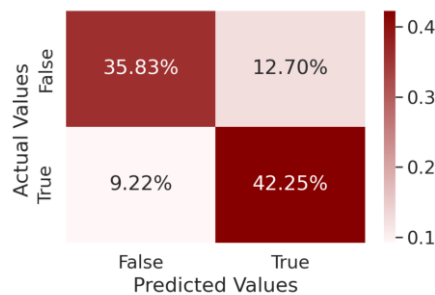


Figure 4.17: Confusion matrix for the *SVM*

4.3.3 German BERT Hate Speech Classifier

Overview

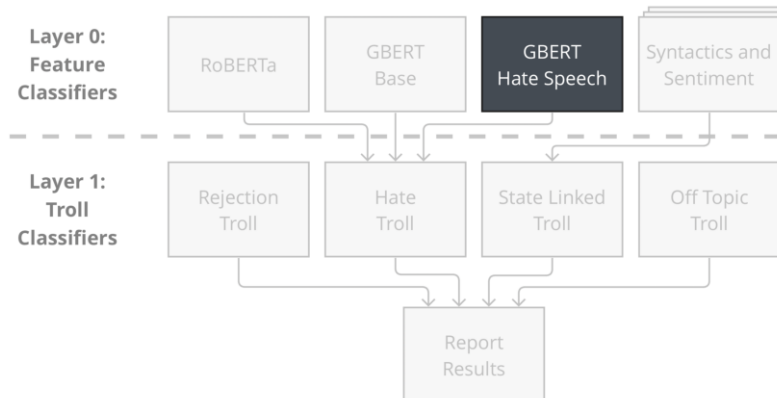


Figure 4.18: German BERT hate speech classifier overview

Trained Embeddings

For this classifier, the pre-trained $BERT_{Hate}$ [48] transformer was chosen and fine-tuned with the train-set for 100 epochs. The $BERT_{Hate}$ transformer was trained on the GermEval18Coarse dataset [48]. 1.0 indicates that the data point is hate speech and 0.0 means it is not hate speech. Reducing the dimensions using *PCA* resulted in the following visualizations:

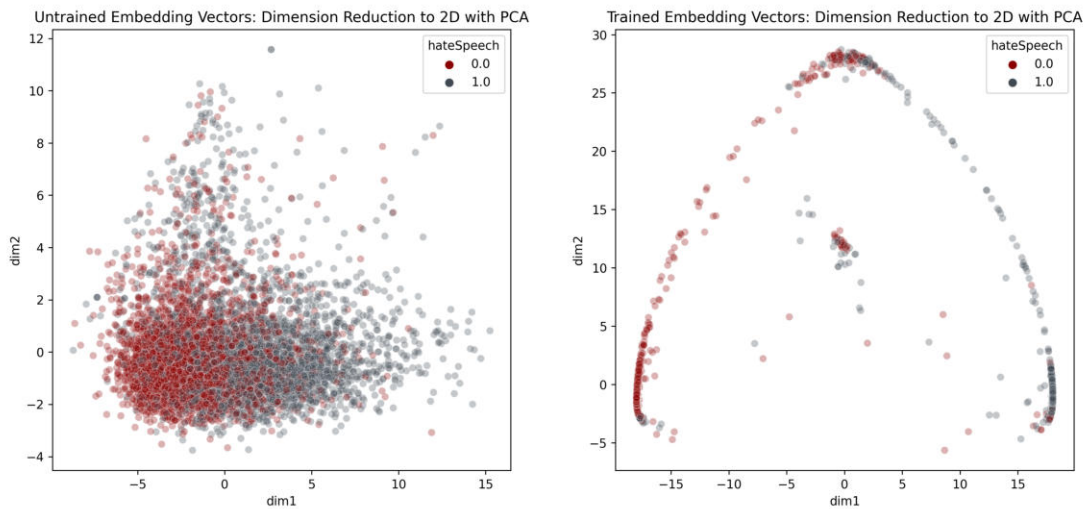


Figure 4.19: Scatterplot of two dimensions reduced by PCA on train-set

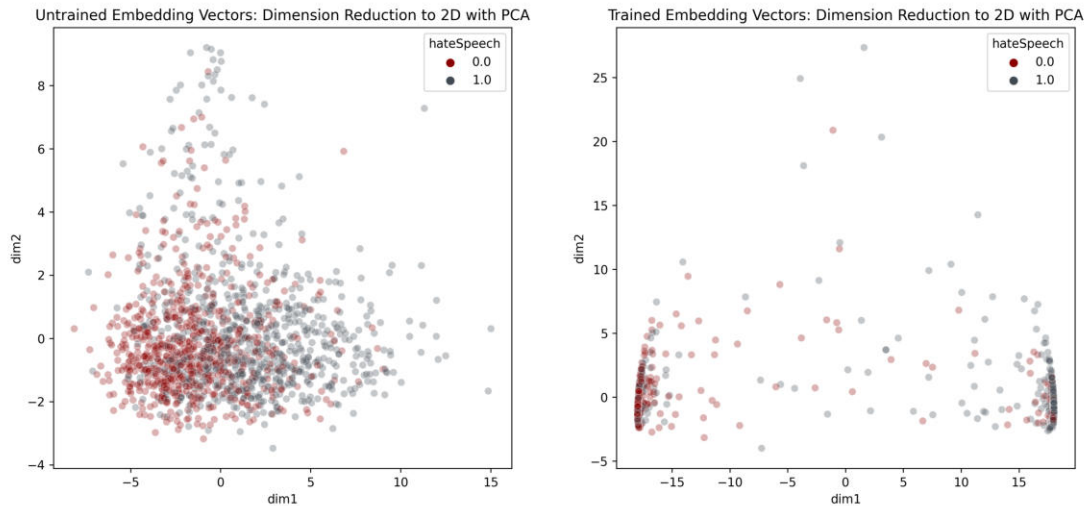


Figure 4.20: Scatterplot of two dimensions reduced by PCA on test-set

Figure 4.19 shows that even for the untrained model, the two labels are somewhat separable. Still, the trained model shows a clear improvement. The explained variance is 25.78% for the untrained and 97.66% for the trained model. The better separability for the untrained model in comparison to the *RoBERTa* model can be explained by the fact that the *BERT_{Hate}* was pre-trained on hate speech.

Principal Component Selection

For this selection, a Scree plot was plotted as well 4.21. The plot shows that 3 components would be enough, but 4 were ultimately chosen as the explained variance increases from 99.19% to 99.62%, further components did only increase it by 0.1% and were ignored.

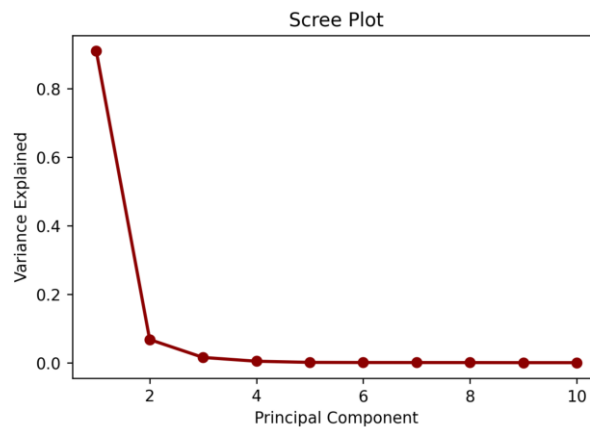


Figure 4.21: Scree Plot of the reduced train-set

Model Selection

To choose a classifier, various ones were tested. The results of the hyperparameter tuned classifiers are shown in the table 4.6. The tuning was done using a randomized search on hyperparameters.

Table 4.6: German BERT Hate: hyperparameter tuning results

Classifiers	Best Hyperparameters	Accuracy	Recall
<i>SVM</i>	kernel: <i>poly</i> , degree: 4, C: 1000, probability: <i>True</i>	0.77	0.79
Random Forest	n_estimators: 1200, min_samples_split: 5, min_samples_leaf: 1, max_features: <i>sqrt</i> , max_depth: 70, bootstrap: <i>True</i>	0.79	0.78
<i>KNN</i>	n_neighbors: 13	0.80	0.79
AdaBoost	learning_rate: 1.0, n_estimators: 100	0.79	0.78
LDA	solver: <i>svd</i> , store_covariance: <i>True</i>	0.80	0.78
MLP	activation: <i>relu</i> , alpha: 0.05, learning_rate: <i>adaptive</i> , hidden_layer_sizes: (50, 100, 50), max_iter: 10000, solver: <i>adam</i>	0.79	0.79
<i>QDA</i>	store_covariance: <i>True</i> , reg_param: 0.001	0.80	0.78
Logistic Regression Linear	penalty: <i>l2</i> , C: 16.496480740980207	0.80	0.78

The result shows that the *KNN* has the highest accuracy and recall scores. Therefore, it was chosen for this classifier.

Evaluation

The classifier achieved a recall of 79% on the validation set, which was neither used to tune the transformer nor to train the classifier.

Table 4.7: German Bert hate classifier scores

Metric	Score
precision	79%
recall	79%
f1-score	79%
accuracy	78%

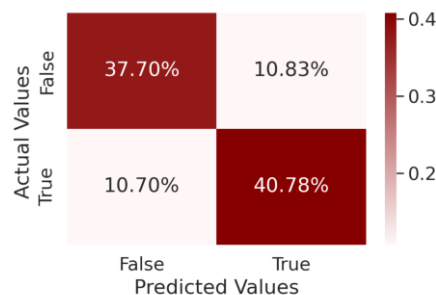


Figure 4.22: Confusion matrix: *KNN* classifier

4.3.4 German BERT Base Classifier

Overview

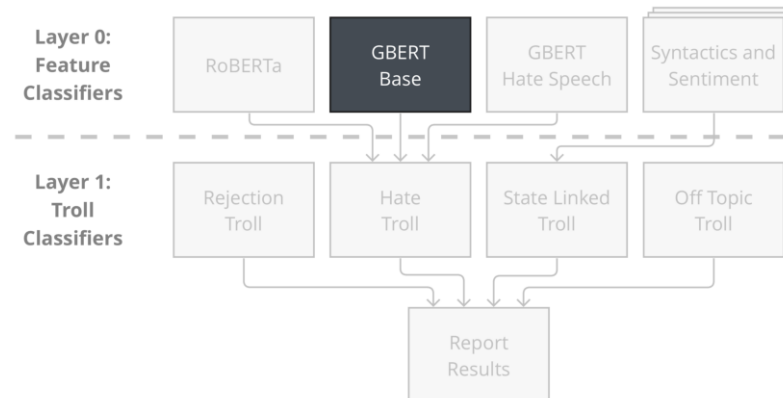


Figure 4.23: German BERT Base hate classifier overview

Trained Embeddings

The last transformer chosen for the abusive classifier is the $GBERT_{Base}$ [49], which was trained with the train-set for 100 epochs as well. Reducing the dimensions using *PCA* resulted in the following visualizations:

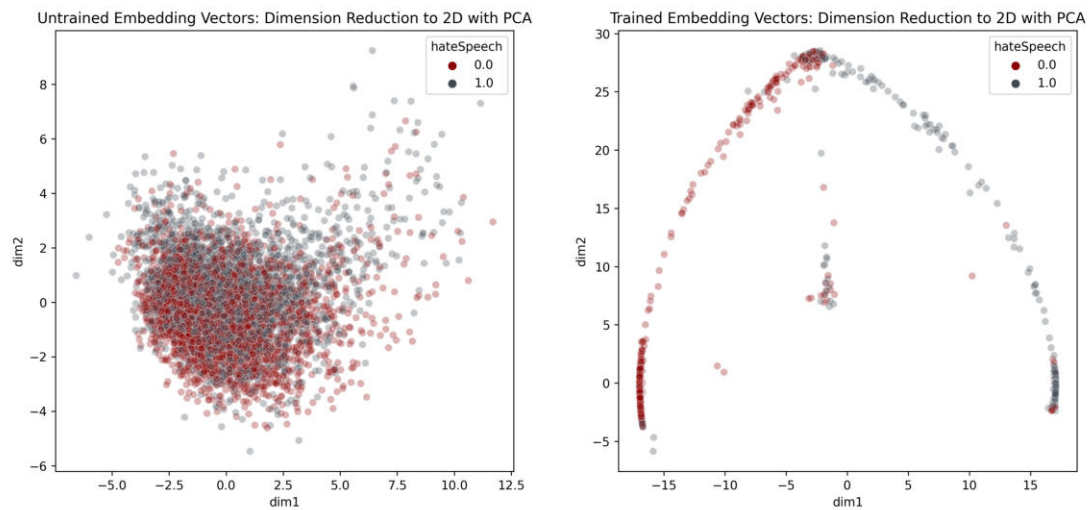


Figure 4.24: Scatterplot of two dimensions reduced by PCA on train-set

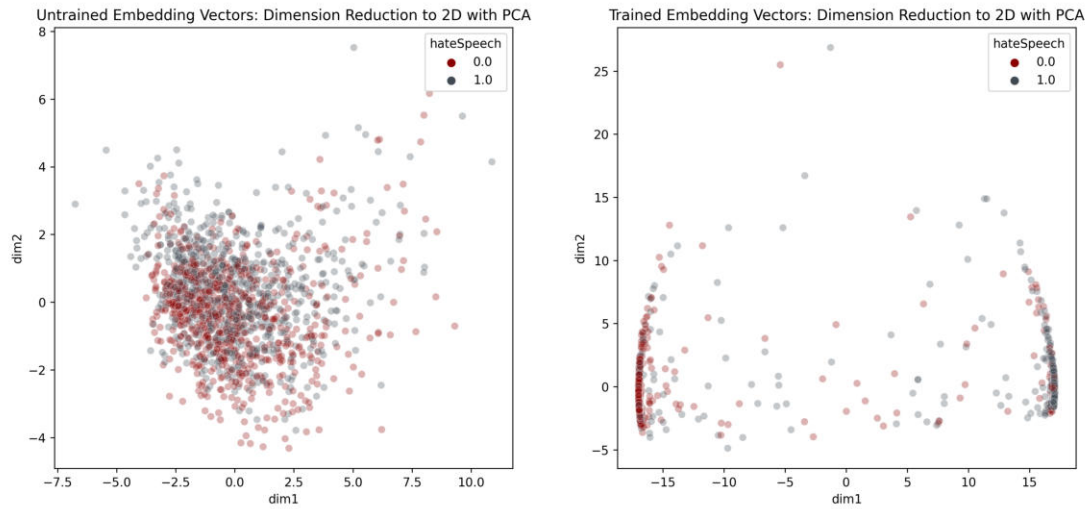


Figure 4.25: Scatterplot of two dimensions reduced by PCA on test-set

The un-tuned $GBERT_{Base}$ transformer shows similar separability as the untrained $RoBERTa$ transformer. The fine-tuned transformer clearly shows the separability of the two labels. The explained variance for two dimensions is 12.01% for the untrained and 98.47% for the trained transformer.

Principal Component Selection

To select the optimal number of components, a Scree plot was made. 3 components together get an explained variance of 98.96%, which is more than enough. The 4th component was added to boost the explained variance to 99.19%. Adding the last component does not increase the complexity of the model unnecessarily because of the low numbers needed to achieve a high explained variance.

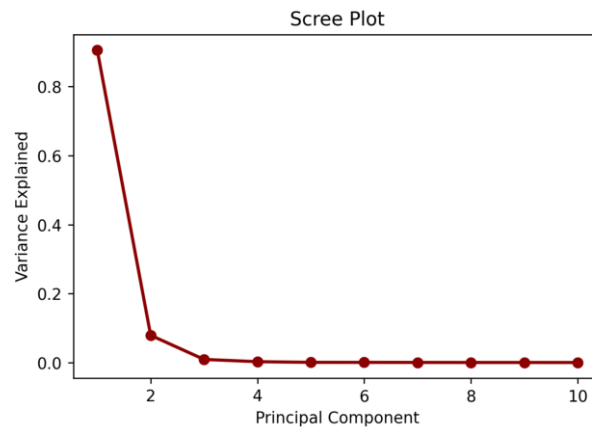


Figure 4.26: Scree Plot of the reduced train-set

Model Selection

The hyperparameters were tuned for multiple classifiers and the results are shown in Table 4.8. The tuning was done in the same manner as the *RoBERTa* and German BERT hate speech classifier described earlier.

Table 4.8: German BERT Base: hyperparameter tuning results

Classifiers	Best Hyperparameters	Accuracy	Recall
<i>SVM</i>	kernel: <i>poly</i> , degree: 4, C: 100, probability: <i>True</i>	0.80	0.75
Random Forest	n_estimators: 1200, min_samples_split: 5, min_samples_leaf: 1, max_features: <i>sqrt</i> , max_depth: 70, bootstrap: <i>True</i>	0.81	0.80
<i>KNN</i>	n_neighbors: 13	0.81	0.79
AdaBoost	learning_rate: 1.0, n_estimators: 100	0.82	0.79
LDA	solver: <i>svd</i> , store_covariance: <i>True</i>	0.81	0.78
MLP	activation: <i>relu</i> , alpha: 0.05, learning_rate: <i>constant</i> , hidden_layer_sizes: (50, 50, 50), max_iter: 10000, solver: <i>adam</i>	0.80	0.77
<i>QDA</i>	store_covariance: <i>False</i> , reg_param: 0.1	0.80	0.78
Logistic Regression Linear	penalty: <i>l2</i> , C: 0.006700187503509591	0.81	0.76

The random forest classifier was chosen because the recall was the highest and the accuracy is only 1% worse than the top performer.

Evaluation

The classifier achieved a recall of 79% on a validation-set which was neither used to tune the transformer nor to train the classifier.

Table 4.9: German Bert Base classifier scores

Metric	Score
precision	85%
recall	79%
f1-score	82%
accuracy	82%

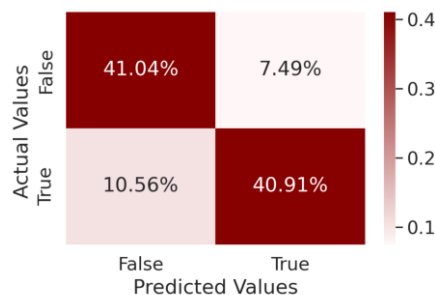


Figure 4.27: Confusion matrix: random forest classifier

4.3.5 Hate Ensemble Classifier

Overview

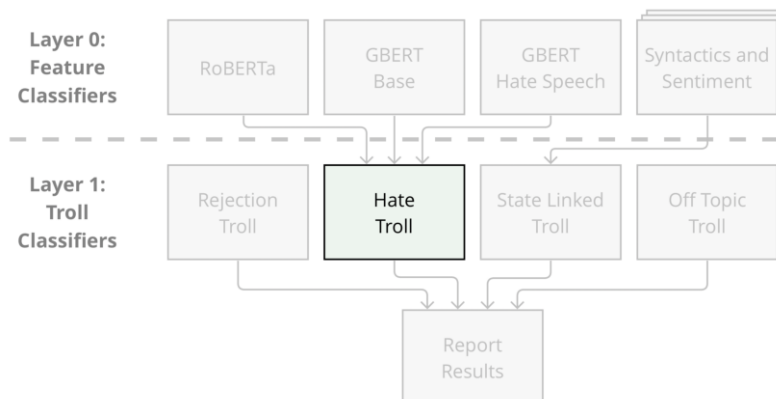


Figure 4.28: Hate classifier ensemble overview

The combination of the three previously presented classifiers is described in this section. The three fine-tuned language models perform better than their untrained counterparts. This explains the steps taken by Assenmacher et. al. [45], where fine-tuned *BERT* models were used as well.

Hypothesis

The hypothesis for the hate troll classifier is as follows:

1. The proposed ensemble classifier performs better in comparison to the individual classifiers.

The hypothesis was accepted, as shown in the following results.

Classifier

For the ensemble classifier, two variants were tested: a stacking classifier and a voting classifier. The final estimator for the stacking classifier was left to the default setting, which is logistic regression. The *ROC* curve was plotted and are visualized in Figure 4.29. The *ROC* curve was made using the test-set:

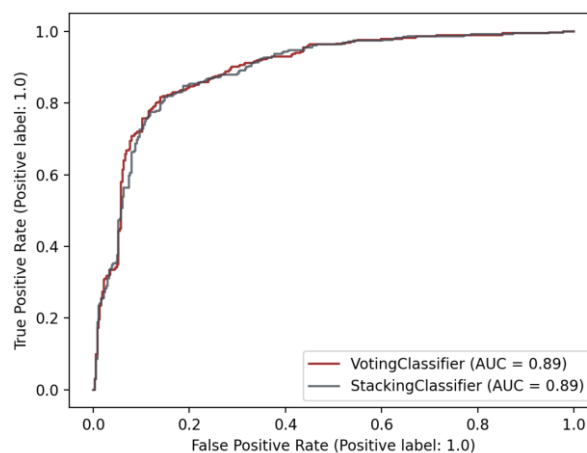


Figure 4.29: ROC curve of the ensemble classifiers

Both classifiers have the same area under the curve, but the voting classifier had a recall of 82%, whereas the stacking classifier only achieved 81%. The voting classifier was therefore chosen.

Evaluation

The voting classifier achieved a recall of 84% on the validation-set. The validation-set was used to get the metrics for the evaluation. It contains data the model never saw before. The area under the curve is 0.90. This is similar to area under the curve achieved by Assenmacher et. al, which was 0.914 [45]. For their classifier, Assenmacher et. al trained solely on RP-Crowd-3, while the data-set for the model used in this thesis combines data from RP-Crowd-3 and data from "Der Standard". The hypothesis that an ensemble method improves the classifier is accepted because the precision, recall, f1-score and accuracy are all over 80% which was not the case for an individual classifier.

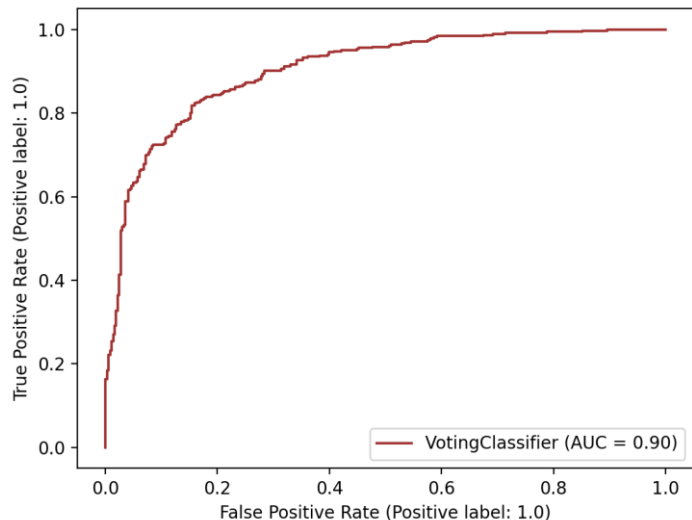


Figure 4.30: *ROC*: voting classifier

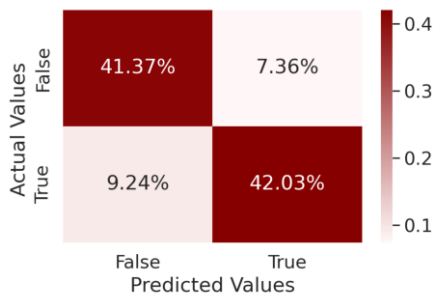


Figure 4.31: Confusion matrix: voting classifier

Table 4.10: Hate ensemble classifier scores

Metric	Score
precision	83%
recall	84%
f1-score	84%
accuracy	83%

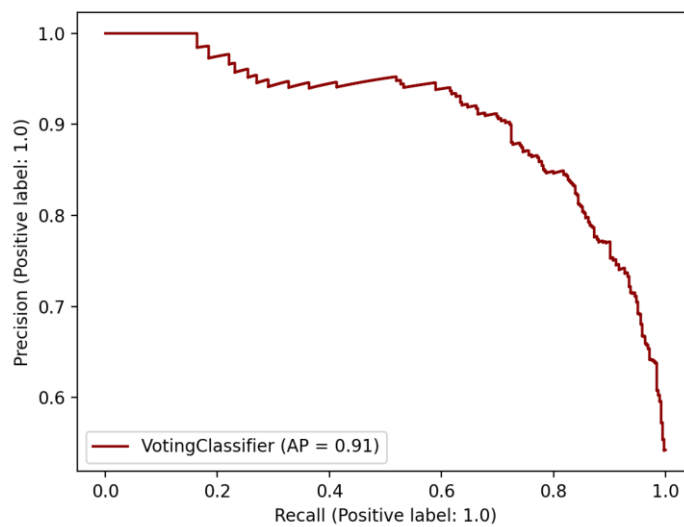


Figure 4.32: Precision recall curve: voting classifier

Examples from the Validation-set

Here are some example comments from the test dataset and the result from the final hate troll classifier:

Table 4.11: Hate troll classifier: test comments

Comment	Pred. Prob.	Pred. Class	Label
Stoppt das Asylgeschäft. Stoppt die Aufnahme von Asylanten. Solange Asylanten aufgenommen werden, wird das Geschäft weiter laufen. Italien macht es richtig.	33%	0.0	1.0
Eigentlich müsste die katholische Kirche - allein schon wegen Verstoßes gegen das AGG im Thema Gleichberechtigung - die Gemeinnützigkeit verlieren und sofort steuerpflichtig werden.	5%	0.0	0.0
Es gibt auch dumme Kühe mit zwei Beinen.	95%	1.0	1.0
Ich hätte da Platz für jemanden, der nix geschenkt will und für Kost und Logis (nicht im Zelt) den ganzen Tag Steine schneidet. Verspreche auch freundlich zu sein und kein Gas in die Augen zu sprühen. Was hält das Gutmenschentum hier von diesem Vorschlag?	69%	1.0	1.0
Für Größenwahnsinnige gibt es das Fletcher's memorial. Aber da Sie viele "Gläubige" (Fanatiker, Patrioten, Unbedarfte/Deppen, Gefolge, Opportunisten) haben - schlage ich mal die gesamte Sahara als Naturschutzgebiet für diese "menschenähnliche" (welche aber nicht so handelt) Existenzen vor. Trump, Putin, Maduro, Orban u.v.a. passen da auch rein! Donald kann die Aufsicht für den Bau eines Doms übernehmen!	96%	1.0	1.0
Deine unterstützung Für die terroristen reicht auch	4%	0.0	1.0
Wen ein Ösi in Deutschland diskriminiert wird bekommt er dann Asyl in den USA?	61%	1.0	0.0

Further analysis of the classifier found that the model learned an inherent bias from the pre-trained

models. To illustrate the problem, the sentence "_____ sind amüsant" was chosen and evaluated by the ensemble classifier and the raw $BERT_{Hate}$ [48].

Table 4.12: Hate troll classifier: bias

Comment	Ensemble classifier	$BERT_{Hate}$
Schweizer sind amüsant	36%	46%
Schwarze sind amüsant	94%	93%
Schwule sind amüsant	95%	97%
Moslems sind amüsant	95%	95%

This inherent bias in the classifier could be explained by the fact that if terms like "Schwarze" or "Schwule" are used in a sentence, the sentences are more often derogatory and containing hate speech in the training data. The question of where the bias came from and the impact of such biases was not explored as part of this thesis.

4.4 State-Linked Troll Classifier

4.4.1 Overview

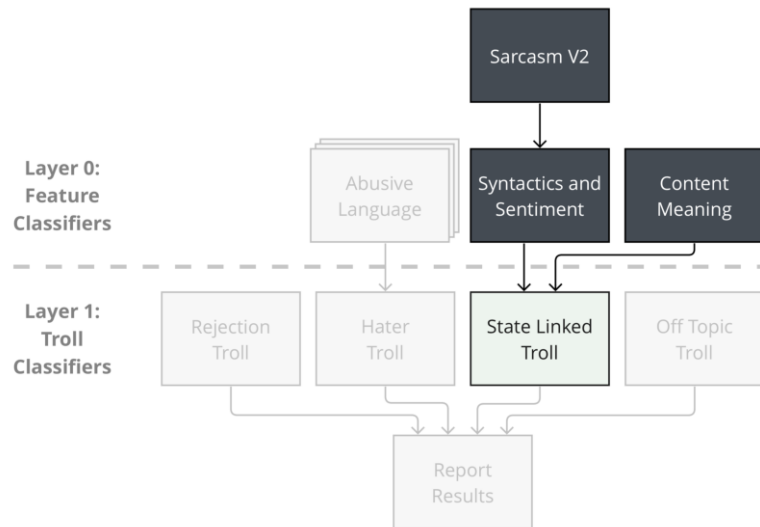


Figure 4.33: State-linked troll overview

The idea for this classifier is to detect state-linked trolls based on syntactical features combined with semantic features using an ensemble method. The individual classifiers and their motivations will be explained in more detail in the subsequent subsections.

The huge number of posts published for state-linked trolling campaigns indicate that the trolls use templates to write their comments [50]. The patterns of those templates could be recognized with the help of machine learning. Alizadeh et al. pointed out that machine learning algorithms perform well with historical training data, but in terms of applying these approaches in the wild to identify currently active operations, the prediction performance drops drastically because trolls change behavior over time [50]. According to Alizadeh et al., the algorithms' performance also depends on the country behind the campaign. They found it easier to spot Chinese activity than Russian activity [51].

Despite their political motivations, state-linked social media trolls do not always talk about politics. In fact, they often just share links to download music or links to some news articles [50]. One explanation for this behavior might be that those professional trolls are first trying to build an audience. This fits with Facebook's statement that sometimes Russian troll networks are removed, which are still in the "early stages" of building an audience [52].

4.4.2 Syntactics and Sentiment Classifier

Overview

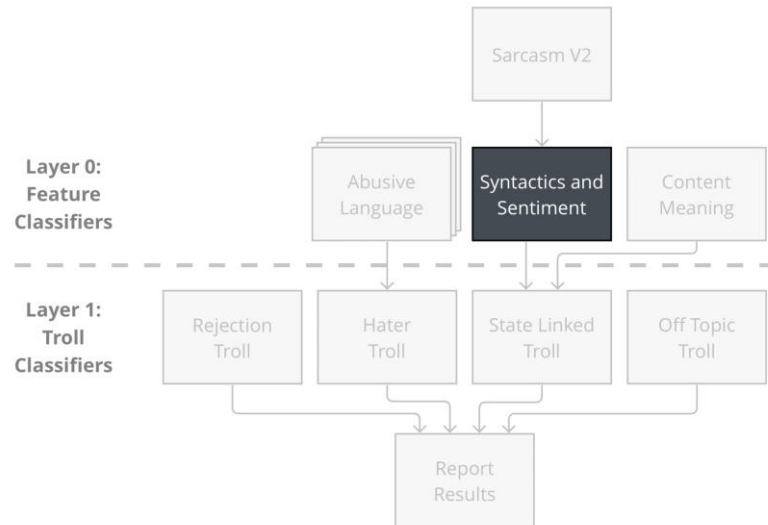


Figure 4.34: Syntactics and sentiment classifier overview

Idea

Ghanem et al. have shown that state-linked trolls can be identified effectively from a purely textual basis using syntactic and stylistic features [53]. Their classifier reaches a precision of 96% and a recall of 93%. These scores show that trolls can be effectively identified using an English dataset. The goal of this classifier is to show that state-linked trolls can also be effectively identified in German language.

Hypotheses

The hypotheses for the syntactics and sentiment classifier are as follows:

1. State-linked trolls employ the parts of speech in German on average differently than ordinary users.
2. Syntactical dependencies between phrases in a sentence differ on average from state-linked trolls to ordinary users.
3. State-linked trolls utilize on average more named entities in their comments, e.g., to refer to politicians, than ordinary users.
4. Frequencies of e.g., emojis or links inside a comment on average differ from state-linked trolls to ordinary users.
5. State-linked trolls score on average a different readability index in their comments than ordinary users.
6. On average, state-linked trolls express a different sentiment in their comments than ordinary users.
7. On average, state-linked trolls have a different misspelling rate in their comments than ordinary users.
8. A classifier to detect state-linked trolls which is trained on a set of tweets produces meaningful results if applied to newspaper comments as well. Due to the lack of training data, this hypothesis was not investigated within the scope of this project. It is assumed that the results are meaningful, although the performance will likely be worse.

A discussion about this hypotheses can be found in Section 4.20. Statistical tests about the hypotheses can be found in the appendix in Chapter D.

Data Acquisition and Understanding

IRA Dataset The IRA dataset³ consists of 3 million Russian troll tweets connected to the Internet Research Agency, a Russian troll factory. Out of these 3 million posts, 100'000 tweets are written in German. A description about this training data set can be found in Section 3.5.3.

By investigating the German tweets, it stands out that only about half of the tweets are about politics. These political tweets often support Donald Trump and express themselves negatively about Hillary Clinton. IRA users also talk positively about Angela Merkel. Examples of such posts are as follows:

- Trump ist wenigstens nicht langweilig #WirLiebenTrump
- Hillary ist an die Probleme der Bürger nicht interessiert #WirLiebenTrump
- #WirLiebenTrump #Trump konzentriert auf die Aufrechterhaltung Stabilität in der Welt
- #Merkel nimmt die Fragen und Anliegen der Menschen ernst #Merkelmussbleiben
- #Merkel wird enge Zusammenarbeit mit #Grossbritannien weiterhin sichern.
- Bei Merkel hat Deutschland keine Feinde! #Merkelmussbleiben

Twitter Dataset Representing Regular Users The IRA dataset only contains posts from troll users. Regular users represent the counterpart. The assumption is that the majority of Twitter users are regular users. Thus, random tweets were sampled in a Twitter archive. After looking through these randomly sampled tweets, it was noticed that people talk about all kinds of topics, while in the IRA dataset about half of the tweets deal with politics.

Data Preparation

The dataset representing the regular users has been selected to increase the number of tweets that include political comments. This was done to ensure that apples are compared with apples. This was especially important for the content meaning classifier described in Section 4.4.4, which uses the same dataset. The political terms used are listed in Table 4.13. The tweets in the IRA dataset mostly refer to German politics and partly to foreign policy. This is why the terms in Table 4.13 have been limited to fit the topics of the IRA trolls. A similar filtering method was performed by Ghanem et al. [53]. Considering that the objective is to classify comments in German-language / Swiss newspapers, this circumstance of primarily focusing on German politics is not optimal.

³<https://github.com/fivethirtyeight/russian-troll-tweets>

Table 4.13: Political terms used for the state-linked dataset

Political Terms			
Abkommen	die Linke	Nato	SPD
AfD	Diktatur	Opposition	Staat
Aussenpolitik	Erdogan	Parlament	Trump
Brexit	EU	Partei	UNO
Bund	FDP	Politik	Verteidigungsministerium
Bundesinnenminister	Flüchtling	politisch	Vize
Bundesland	G20	Propaganda	Wahlen
Bundesregierung	Innenpolitik	Putin	Wahlkampf
Bundesstaat	Kanzler	Rechtsstaat	wählen
CDU	Kanzlerin	Regierung	Wähler
Clinton	Koalition	Sanktionen	
CSU	Merkel	Scholz	
Demokratie	Minister	Schröder	
die Grünen	Ministerium	Seehofer	

Furthermore, it was noticed that the average tweet length for ordinary users is about 146 characters while IRA trolls have an average length of about 106 characters. To ensure that short texts are not automatically classified as a troll text, the dataset representing regular users has been adjusted by randomly dropping longer tweets, targeting the 106 characters on average.

The following table explains what additional text cleaning steps were applied to the entire dataset before feature engineering, and what elements were purposely kept in the data.

Table 4.14: State-linked classifier: text cleaning

Element	Removed?	Reason
Hashtags	Yes	Hashtags are rarely used inside 20 Minuten comments. In the end, the goal is to identify troll comments for newspaper articles and not troll tweets . Additionally, the IRA dataset and the dataset for the regular tweets are both a few years old and contain mostly outdated hashtags. Especially hashtags from the IRA datasets like #Merkelmussbleiben or #WirLiebenTrump are less prominent these days. This was confirmed by the advanced search on Twitter ⁴ .
Links	No	The average tweet in the dataset contains many more links than a regular comment on 20 Minuten. Nonetheless, they are kept in the dataset to extract the number of links features later. However, for the other feature engineering steps they are removed to reduce noise.
Mentions	No	Reply comments on 20 Minuten often refer to one other comment and thus only contain one mention to another user. It is possible, but rarely seen, that a comment contains multiple mentions. The mentions are kept inside the dataset, so that the number of mentions can be included as a feature later.

⁴The following query was performed: [https://twitter.com/search?q=\(%23Merkelmussbleiben\)&src=typed_query&f=live](https://twitter.com/search?q=(%23Merkelmussbleiben)&src=typed_query&f=live)

In a last step, the troll and non-troll datasets were balanced, so that they both contain 80'000 tweets each, resulting in a total of 160'000 samples.

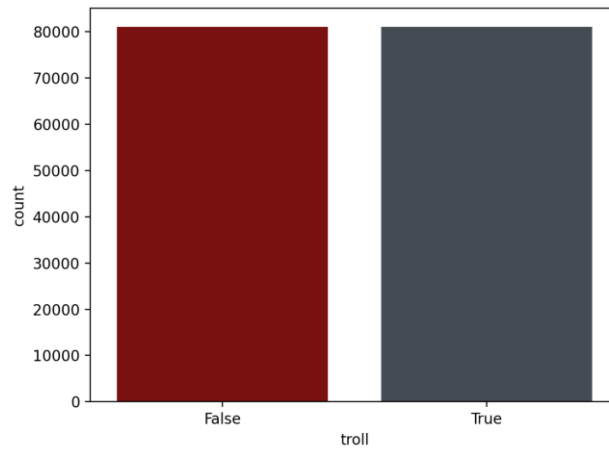


Figure 4.35: State-linked troll classifier label counts

Feature Engineering

Table 4.15: Features overview for the state-linked classifier

Category	Features	Amount
Part of Speech	Percentages of adjectives, adpositions, adverbials, auxiliary verbs, comparative conjunctions, determiners, interjections, nouns, numbers, particles, pronouns, proper nouns, punctuations, subordinate conjunctions, spaces, verbs	16
Syntactic Dependency Parsing	Max dependency childs, Mean dependency childs, Numbers of ac, adc, ag, ams, app, avc, cc, cd, cj, cm, cp, cvc, da, dep, dm, ep, ju, mnr, mo, ng, nk, nmc, oa, oc, og, op, par, pd, pg, ph, pm, pnc, punct, rc, re, rs, sb, sbp, svp, uc, vo (see below for a description)m	43
Named Entity Recognition	Numbers of locations, organizations, persons, miscellaneous	4
Frequency	Number of words, Number of exclamation marks, Number of question marks, Number of uppercase words, Number of emojis, Number of smilies :), Number of big smilies :D, Number of wink smilies ;), Number of sad smilies :(, Number of mentions, Number of links, Number of characters per word, Number of syllables per word, Number of long words, Number of complex words	15
Readability	FleschReadingEase, Kincaid, ARI, SMOGIndex, RIX, LIX, Coleman-Liau, DaleChallIndex, Gunning-FogIndex	9
Sentiment	Polarity, Subjectivity	2
Sarcasm	Sarcasm (see Section 4.4.3)	1
Misspelling	Number of misspelled words	1
Total		91

Part of Speech Ghanem et al. showed that native language identification features are among the most important features to identify state-linked trolls [53]. These features include part of speech features. These features are extracted using Spacy⁵ and the categories are counted using sklearn's `CountVectorizer`. To reduce noise, two preprocessing steps were applied: removal of mentions and removal of links.

Syntactic Dependency Parsing This feature is also considered a native language identification feature described by Ghanem et al. [53]. The goal is to examine the dependencies between phrases of a sentence to determine its grammatical structure [54]. This feature is extracted using Spacy⁶. The German dependency labels are inspired by the TIGER Treebank annotation scheme⁷. The TIGER corpus was built with about 50'000 sentences taken from the Frankfurter Rundschau and consists of words with their respective annotations [55]. To make things more clear, an example is provided. For the German sentence "Sie will wieder Berge sehen", the following graph is generated by displacy⁸:

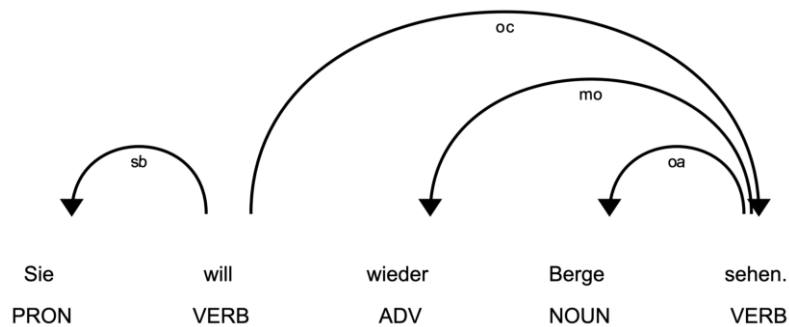


Figure 4.36: Dependency parsing sample graph

With the corresponding table:

Table 4.16: Syntactic dependency parsing labels (example)

Word	Label	Children
Sie	sb	[]
will	ROOT	[Sie, sehen]
wieder	mo	[]
Berge	oa	[]
sehen.	oc	[wieder, Berge]

The feature "max dependency childs" corresponds to the to the maximum number of child dependencies for a word in a comment and "mean dependency childs" corresponds to the mean. This feature is not taken from literature, the idea is to capture some degree of linkage.

A selection of dependency labels are described in Table 4.17. For the rest, please refer to the TIGER annotation scheme paper by Stefanie Albert et al. [55].

⁵<https://spacy.io>

⁶<https://spacy.io>

⁷<https://v2.spacy.io/api/annotation>

⁸<https://spacy.io/usage/visualizers>

Table 4.17: Syntactic dependency parsing labels

Label	Description	Explanation
PNC	Proper Noun Component	Corresponds to the individual components of first names, surnames and name affixes [55]. Examples are: <ul style="list-style-type: none"> • Der unglückliche Jose_{PNC} Maria_{PNC} Sanchez_{PNC} • Die Pfadfindergruppe "Edelweiss"_{PNC}
NK	Noun Kernel	There are pronominal, substantival and adjectival noun kernel elements [55]. Several noun kernels together can form one nominal phrase (NP). Examples are: <ul style="list-style-type: none"> • [Die_{NK} grosse_{NK} schwarze_{NK} Katzen_{NK}]_{NP} • [Die_{NK} [auf ihren_{NK} Sohn_{NK} stolze]_{NK} Frau]_{NP} • [10'000_{NK} Menschen_{NK}]_{NP} • [Bei der_{NK} Bundestagswahl_{NK} 1998_{NK}]_{NP}
RS	Reported Speech	Normally reported speech is a grammatical mechanism for reporting the content of someone else's statement without directly quoting it [56]. However Stefanie Albert et al. used it in their examples as quoted speech [55]. Examples are: <ul style="list-style-type: none"> • Helmut Kohl: "Der Mantel der Geschichte ..." _{RS} • "Ich hoffe, dass es klappt" _{RS} verabschiedete sich der Arzt
CJ	Conjunct	Two or more conjuncts (CJ) are joined by a coordinating conjunction (CD) to form one coordination [55]. Examples are: <ul style="list-style-type: none"> • Peter_{CJ} und_{CD} [sein Schwager]_{CJ} • Weder_{CD} Peter_{CJ} noch_{CD} Paul_{CJ}
CD	Coordinating Conjunction	Stefanie Albert et al. lists the following words as coordinating conjunctions [55]: und _{CD} , aber _{CD} , denn _{CD} , doch _{CD} , wie _{CD} , sowie _{CD} , bis _{CD} , beziehungsweise _{CD} /bzw. _{CD} , respektive _{CD} /resp. _{CD} . Binary coordinating conjunctions receive the same label [55]: entweder oder _{CD} , weder noch _{CD} , sowohl als _{CD} .
CP	Complementizer	All sentence-initiating conjunctions that trigger a the verb to be at the last position (German: Verbletzstellung) are annotated as CP [55]. Examples are: <ul style="list-style-type: none"> • [Obwohl oder gerade weil]_{CP} er kommt
RC	Relative Clause	A relative clause is a clause that modifies a noun or noun phrase [57]. Examples are: <ul style="list-style-type: none"> • Der Mann, [dessen Tochter ich kenne]_{RC} • Dort, [wo er wohnt]_{RC}
SB	Subject	Corresponds to the subject of a sentence [55].
MO	Modifier	A modifier is a word or phrase that modifies another word in the same sentence. [55]. An examples is: <ul style="list-style-type: none"> • Fast_{MO} eine Million Studenten
OA	Accusative Object	Corresponds to an accusative object in a sentence [55].

Continued on next page

Table 4.17 – continued from previous page

Label	Description	Explanation
OC	Object Clausal	Parts that are complements of nominal phrases are annotated as OC. [55]. An examples is: <ul style="list-style-type: none"> • Der Beschluss, [ein Haus zu bauen]_{OC}

To reduce noise, two preprocessing steps for the syntactic dependency parsing were finally applied: removal of mentions and removal of links.

Named Entity Recognition This feature is not part of the analyzed papers dealing with state-linked trolls. The hypothesis for this thesis states that state-linked trolls might mention people such as politicians or places by name more often than ordinary users. This feature was also extracted using Spacy and the named entities are counted using sklearn’s `CountVectorizer`. No preprocessing steps are applied to extract these features.

Frequency For this category, the number of links might be of most interest because manual inspection has shown that a lot of state-linked trolls use links in their posts. The number of characters and syllables per word and the number of long and complex words are extracted using the `Readability`⁹ package. The rest is counted using regular expressions. Words are considered long if they exceed 6 characters [58]. Words are considered complex if they consist of three or more syllables [59]. This, however, does not include compound words, familiar jargon or proper nouns [59].

Readability As the assumption is that as most trolls are not native speakers, their writing style should be different. This could manifest itself in the readability indexes. Michele Tomaiuolo et al. also suggested this feature in their paper [1]. The readability indexes are extracted using `Readability`¹⁰, a Python library which also supports German. The indexes are described in more detail in Table 4.18. It should be noted that these readability scores are primarily developed for english text. Thus, the most meaningful results would be achieved using English text. To get meaningful results in German, some original formulas had to be adapted, like adding German word lists. It is left to the feature selection algorithm to decide, whether these scores are meaningful enough for classification.

Table 4.18: Readability scores

Index	Description
FleschReadingEase	Developed by Rudolf Flesch to improve the readability of newspapers [60]: $score = 206.835 - 1.015\left(\frac{words}{sentences}\right) - 84.6\left(\frac{syllables}{words}\right)$
Kincaid	Developed by the US Navy to improve their technical manuals [60]: $score = 0.39\left(\frac{words}{sentences}\right) + 11.8\left(\frac{syllables}{words}\right) - 15.59$
ARI	ARI relies on a factor of characters per word, instead of the usual syllables per word [61]: $score = 4.71\left(\frac{characters}{words}\right) + 0.5\left(\frac{words}{sentences}\right) - 21.43$

Continued on next page

⁹<https://pypi.org/project/readability/>

¹⁰<https://pypi.org/project/readability/>

Table 4.18 – continued from previous page

Index	Description
SMOGIndex	It measures how many years of education the average person needs to have to understand a text [62]. It is best for texts of 30 sentences or more: $score = 1.043\sqrt{polysyllables * \frac{30}{sentences}} + 3.1291$
LIX	This measure indicates the difficulty of reading a text [58]: $score = \frac{words}{sentences} + \frac{long\ words * 100}{words}$
RIX	The RIX index is a simplified version of LIX [63]: $score = \frac{long\ words}{sentences}$
Coleman-Liau	This index approximates the US grade level required to understand the text [64]: $score = 0.0588L - 0.296S - 15.8$ Where L is the average number of letters per 100 words and S is the average number of sentences per 100 words.
DaleChallIndex	A list of easily understandable words is used. Any word not in that list is considered to be difficult [65]: $score = 0.1579\left(\frac{difficult\ words}{words}\right) + 0.0496\left(\frac{words}{sentences}\right)$
GunningFogIndex	The index estimates the years of formal education a person needs to understand the text on the first reading [59]: $score = 0.4\left(\frac{words}{sentences}\right) + 100\left(\frac{complex\ words}{words}\right)$

One preprocessing step is applied: Inserting a newline character after every sentence so that the library recognizes each sentence. Links are intentionally not removed because many links make a text less readable.

Sentiment Ghanem et al. included sentiment features for their IRA classifier [53]. However, their sentiment feature only focused on polarity, which turned out to be semi-useful [53]. For this classifier, subjectivity and objectivity of the comment is also included. No preprocessing steps are applied, since this is handled by the library `textblob-de`¹¹.

Sarcasm As Ghanem et al. pointed out in their work that IRA trolls seem to be frequently employing sarcasm [53]. As no Python library was found to extract this feature for German texts, a custom sarcasm classifier was created for this purpose, which is described in Section 4.4.3. To remove noise, the following preprocessing steps are applied: removal of mentions, removal of links. This feature is not likely to be significant because the performance of the custom sarcasm classifier is low.

Misspelling This feature could not be found in other papers related to classification of state-linked trolls. The assumption is that the majority of state-linked trolls are not native speakers and are thus more likely to make spelling mistakes. However, with the increasingly better translation tools this might not be an issue anymore. This feature is extracted using `pyspellchecker`, a python library¹² which also supports German. Unfortunately, this library is far from perfect. Country

¹¹<https://github.com/markuskiller/textblob-de>

¹²<https://github.com/barrust/pyspellchecker>

names like "Schweiz" and less known vocabulary are also counted as misspelling. Links also cause the misspelling count to increase. Hence, the following steps are applied in preprocessing: removal of mentions, removal of links and removal of punctuation.

Feature Selection Before the feature selection algorithm, it was necessary to find out which characteristics correlate. If there is collinearity between only two features, then inspecting the correlation matrix will show a high correlation between them. But if there is collinearity between more than two variables, a more elaborate approach is needed [66]. This was achieved using the variance inflation factor. Figure 4.37 shows the features where the collinear features are already removed.

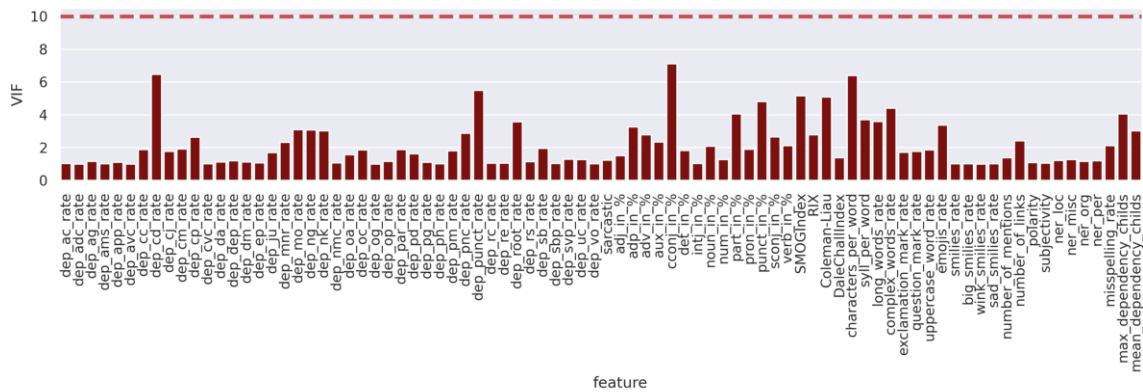


Figure 4.37: VIF plot

As a rule of thumb, values above 5 or 10 indicate a problem with collinearity [66]. In Figure 4.37, 10 was chosen as the threshold.

Now that the collinear features are eliminated, the most important features need to be selected. This is done using the Forward Stepwise Selection algorithm described in G.3.3. First, it was tried to optimize the recall score. This did not work out well, since the precision score was consistently low. Hence, the F1 score was optimized.

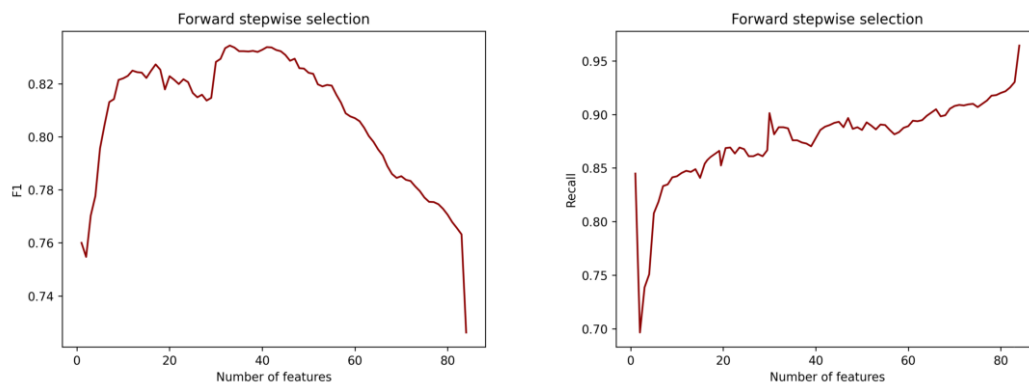


Figure 4.38: Feature selection scorings run

The first 22 features are selected to train the syntactic and sentiment classifier. These characteristics are listed in Table 4.19 in descending order by their importance according to Figure 4.38.

Table 4.19: State-linked classifier: optimal features sorted

1. Features Sorted ↓	2. Features Sorted ↓	3. Features Sorted ↓
number_of_links	misspelling_rate	adj_in_%
number_of_mentions	long_words_rate	ner_per
mean_dependency_childs	emojis_rate	dep_rs_rate
adp_in_%	verb_in_%	
sarcastic	punct_in_%	
dep_pnc_rate	SMOGIndex	
syll_per_word	ner_loc	
DaleChallIndex	conj_in_%	
noun_in_%	polarity	

As the corresponding hypothesis tests have shown (see Appendix D.2), the `conj_in_%`-curves do not appear to be significantly different for both classes. Hence, this feature was dropped.

Modeling

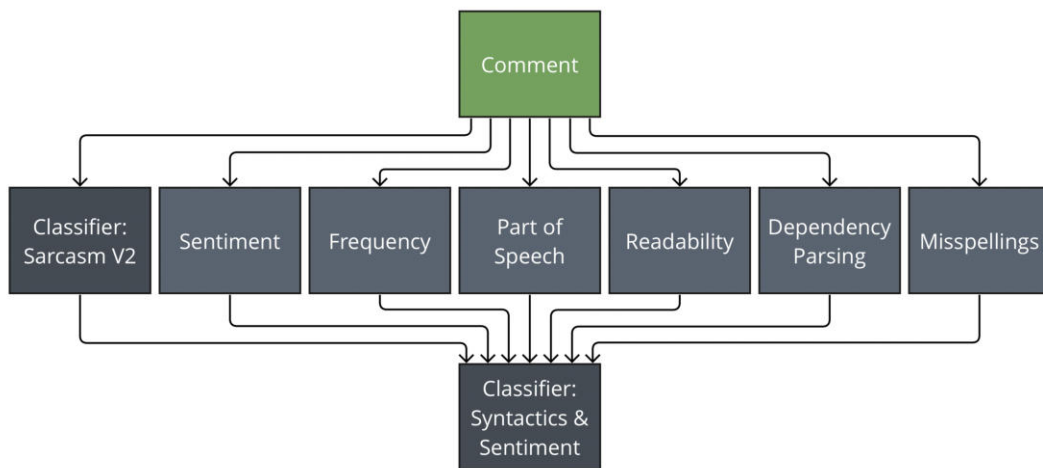


Figure 4.39: State-linked troll classifier structure

The optimal classification scheme with its optimal hyperparameters was evaluated on a set of 40'000 tweets.

Table 4.20: State-linked classifier: hyperparameter tuning results

Classification Scheme	Best Hyperparameters	Accuracy	Recall
<i>SVM</i>	kernel: <i>rbf</i> , gamma: 0.001, C: 1000	0.84	0.86
Random Forest	n_estimators: 800, min_samples_split: 5, min_samples_leaf: 1, max_features: <i>sqrt</i> , max_depth: 50	0.90	0.91
LDA	solver: <i>svd</i>	0.80	0.81
<i>QDA</i>	reg_param: 0.1	0.78	0.95
Logistic Regression Linear	penalty: <i>l2</i> , C: 93	0.82	0.84
Logistic Regression Poly 2	penalty: <i>l2</i> , C: 39	0.84	0.85
Logistic Regression Poly 3	penalty: <i>l2</i> , C: 61	0.84	0.86
<i>KNN</i>	n_neighbors: 17	0.83	0.89

The random forest scheme seemed to be the most promising and was thus picked to train the classifier with all the training data.

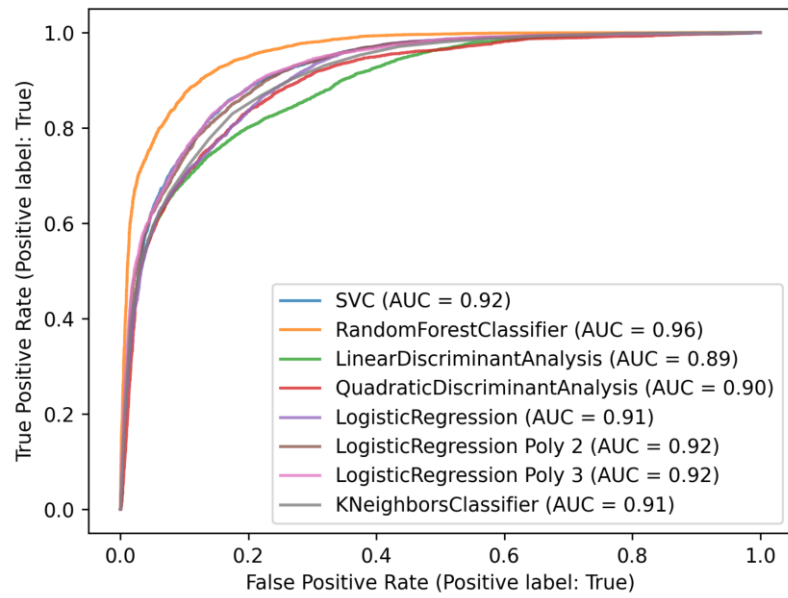


Figure 4.40: State-linked classifier ROC plots

The stratification of the feature space performed by the random forest classifier appears to work optimally for these particular features. This is especially true for features with discrete values such as the number of links.

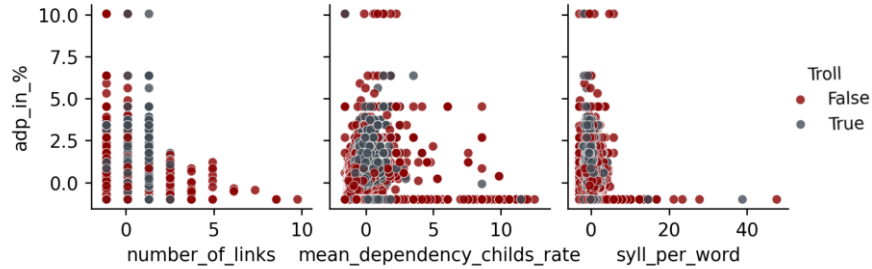


Figure 4.41: Pairplots syntactic features

Evaluation

The syntactics and sentiment subclassifier provides an interesting insight into how state-linked trolls structure their texts. For example, trolls seem to employ nouns and adpositions more often than ordinary users.

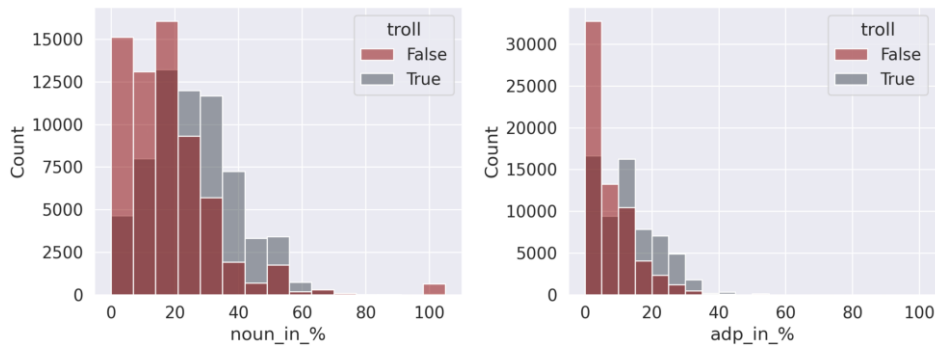


Figure 4.42: State-linked trolls usage of nouns (left) and adpositions (right)

Links are frequently used by state-linked trolls. The average link count of a state-linked troll is above 1. State-linked trolls seem to be slightly more sarcastic as well.

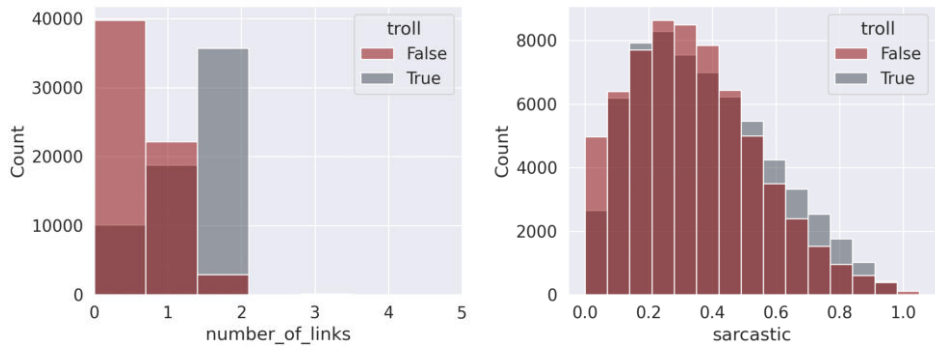


Figure 4.43: State-linked trolls usage of links (left) and sarcasm (right)

State-linked trolls tend to use reported speech more often than regular users and they decorate their text less with emojis.

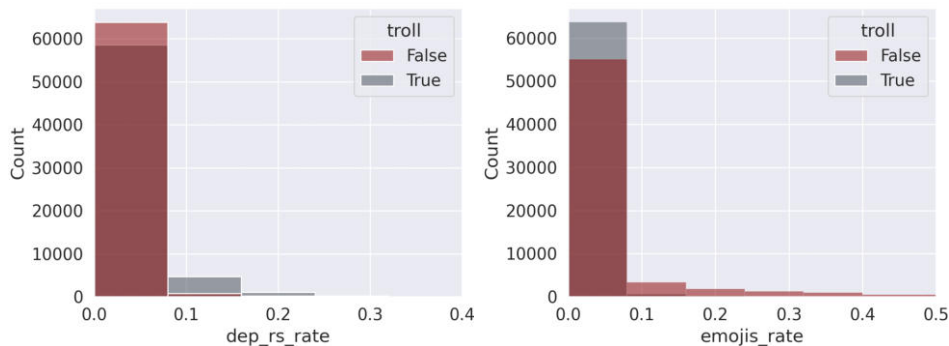


Figure 4.44: State-linked trolls usage of reported speech (left) and emojis (right)

It looks like named location (`ner_loc_rate`) and person (`ner_per_rate`) entities are more popular amongst state-linked trolls.

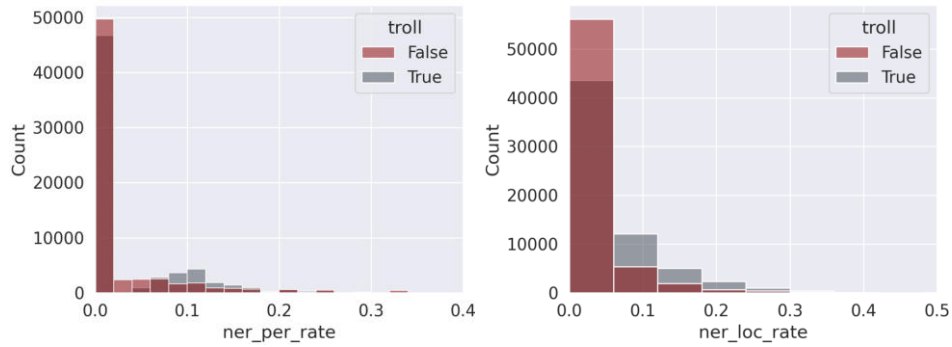


Figure 4.45: State-linked trolls usage of named person (left) and location (right) entities

A more extensive selection of features including their hypothesis tests is listed in Appendix D.

Figure 4.46 shows the features used in the syntactic and sentiment classifier with their respective importance.

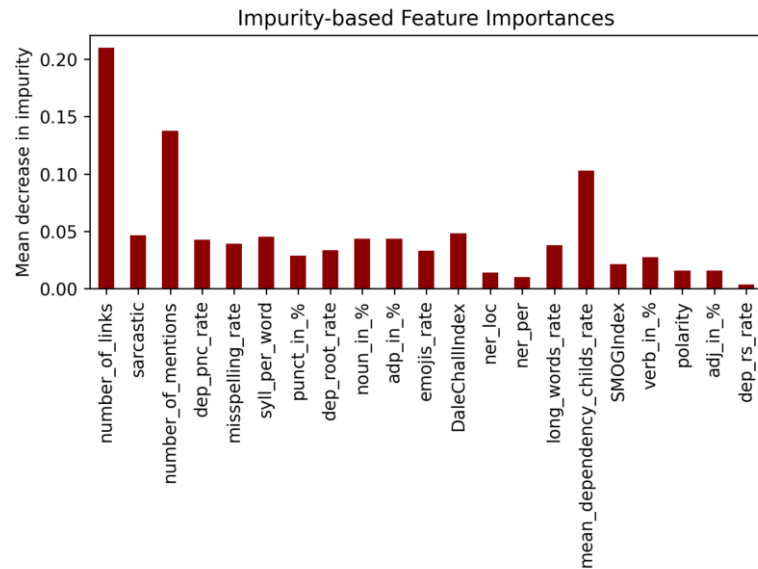


Figure 4.46: Feature importance for the syntactic and sentiment subclassifier

Trained on all the 129'000 training samples, the syntactic and sentiment classifier achieves a good performance. The random forest classifier is tested on a test-set of 16'000 samples that have never been used during training or validation. On this test-set, the classifier achieves an accuracy of 91% and a recall of 92% for the label "true" (troll).

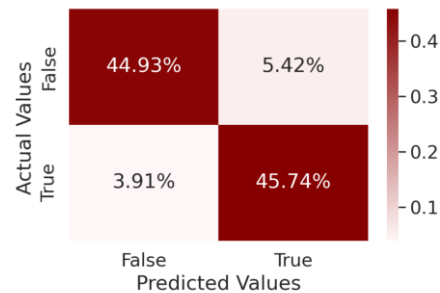


Figure 4.47: State-linked classifier syntactic and sentiment confusion matrix

Table 4.33 shows the final scores of the state-linked syntactics and sentiment classifier.

Table 4.21: State-linked ensemble classifier scores

Metric	Score
precision	89%
recall	92%
f1-score	91%
accuracy	91%

Limitations

An important limitation is that the Twitter dataset does not represent typical comments from an online newspaper. In the end, the goal is to identify trolls in the comment sections of online newspapers and not on Twitter. Manual inspection of the Twitter dataset has revealed a quite different writing style with a wider use of links, hashtags, emojis and multiple mentions in one post. Also, the character limit is different. While Twitter, at the time of this writing, allows for only 280 characters in one post¹³, 20 Minuten comments can contain up to 1000 characters. There is no training data available for state-linked trolling campaigns in German speaking online newspapers. Therefore it was not evaluated as part of this project how well the classifier performs in such a context.

Another limitation is the fact that the publication time of the regular posts and the state-linked trolls differs by over one year. Different topics are of interest in these different time periods and user behavior might change as well. This can cause the classifier to perform better on the evaluation data compared to real use.

¹³<https://developer.twitter.com/en/docs/counting-characters>

4.4.3 Sarcasm Classifier Version 2

Overview

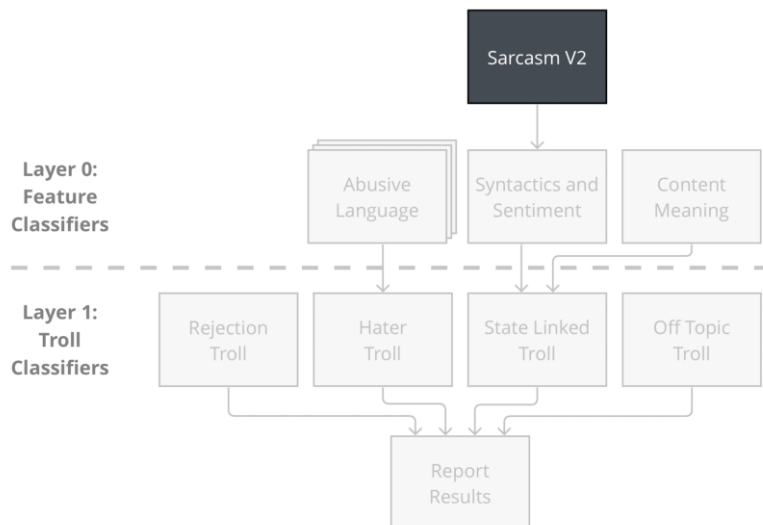


Figure 4.48: Sarcasm classifier version 2 overview

The sarcasm classifier version 2 described in this subsection is part of the syntactics and sentiment classifier discussed in Section 4.4.2 and an update to the initial sarcasm classifier version 1 illustrated in Appendix A.2.

Idea

After a manual inspection of the IRA dataset, Ghanem et al. pointed out that irony is frequently employed in troll tweets [53]. However, they did not include this feature in their classifier. Following up on this existing research, it is interesting for this thesis to validate whether irony helps to detect trolls. As no existing German classifier could be found, a custom classifier was to be implemented for this purpose.

The reason for updating the first sarcasm classifier to a new version was that experiments have shown that a simpler approach is already sufficient, whereas the previous classifier was much more complex. This updated classifier works with the same data as the previous classifier.

Literature

Various papers have shown that the classification of irony is challenging [67], but Francesco Barbieri et al. showed that it is to some degree possible to distinguish ironic from non-ironic tweets [68]. Their irony classifier achieved a F1 score of 60%. They struggled to differentiate sarcastic from ironic comments. The sarcasm classifier version 1 follows this approach more closely than version 2.

Sambit Mahapatra has shown that a more simplistic approach can already suffice [69]. Using FastText, his classifier achieves 86% precision and 86% recall, respectively. This was the inspiration for the sarcasm classifier version 2.

Definition of Sarcasm and Irony

According to the British encyclopedia as quoted from Wikipedia, sarcasm is defined as follows:

“Sarcasm is the caustic use of words, often in a humorous way, to mock someone or something. Sarcasm may employ ambivalence [...].”

(Wikipedia - 10.04.2022, British encyclopedia)

Irony is defined as follows in the Greek-English Lexicon, as quoted from Wikipedia:

“Irony [...], in its broadest sense, is a rhetorical device [...] in which what on the surface appears to be the case or to be expected differs radically from what is actually the case.”

(Wikipedia - 10.04.2022, Greek-English Lexicon)

In both cases, different things are said than what is actually meant. The main difference is that sarcasm focuses more on negativity. During a conversation, sarcasm, and irony can be recognized by the context, facial expressions, gestures, and tone of voice. For text, people switch to other forms of communication, such as emojis, to convey these aspects. Although sarcasm largely depends on the context, it should to some degree be possible to automatically detect sarcasm.

Hypothesis

The hypothesis for the sarcasm classifier version 2 is as follows:

1. Sarcasm in German can be detected with similar performance to Sambit Mahapatra’s classifier using English data.

This hypothesis is discussed in the following sections.

Data Acquisition

Unfortunately, no existing German training data was found. Therefore, it was decided to work with an originally English training dataset which is translated to German. The danger of this approach is that meaning is lost during the translation. The following two datasets were used for training:

The Headlines Dataset The headlines dataset¹⁴ consists of 30’000 headlines from sources like the HuffPost¹⁵ or the satirical platform The Onion¹⁶.

The Reddit Dataset A labeled reddit dataset¹⁷ was found on Kaggle. This dataset was generated by scraping comments from Reddit containing the `#sarcasm` tag. Out of the 1 million comments, 100’000 comments were randomly selected for further analysis to not exceed the available processing power.

¹⁴<https://www.kaggle.com/datasets/rmisra/news-headlines-dataset-for-sarcasm-detection?resource=download>

¹⁵<https://www.huffpost.com>

¹⁶<https://www.theonion.com>

¹⁷<https://www.kaggle.com/datasets/danofer/sarcasm>

Data Preparation

The two datasets were merged into one big English dataset. These two datasets are focused on sarcasm rather than irony. The final dataset consists of about 73'000 non-sarcastic and 55'000 sarcastic texts.

The English dataset was translated to German using DeepL¹⁸. A glance at a sample of the translated dataset compared to the original dataset showed that the majority of translations seemed to be reasonable.

Drawing the word-clouds of the sarcastic and the non-sarcastic dataset with stop words removed indicates that sarcastic and non-sarcastic texts cannot only be detected by the words used in the text. Other structural information has to be considered as well.

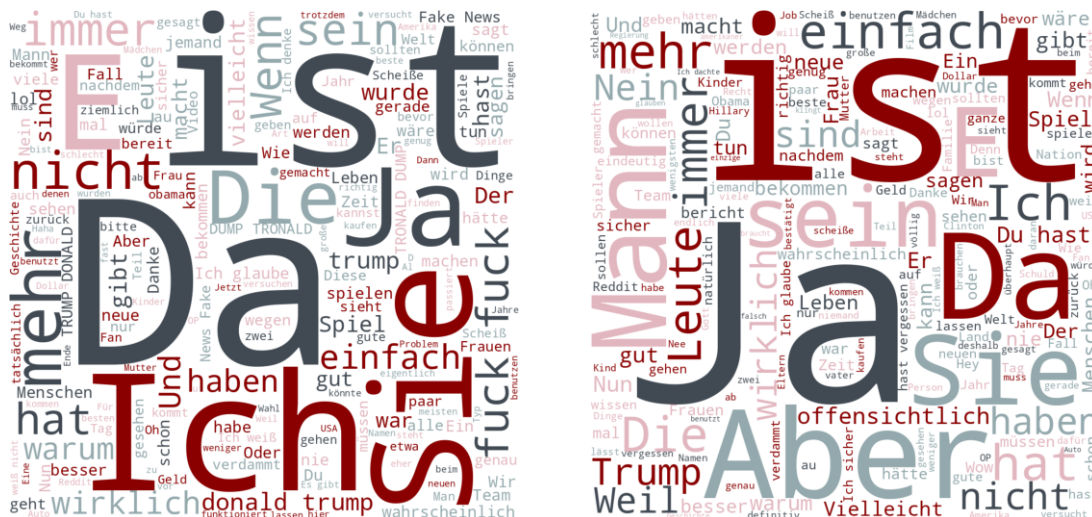


Figure 4.49: Non-sarcastic wordcloud (left) and sarcastic wordcloud (right)

Feature Engineering

This classifier is built on FastText. To also include the part of speech structure, a *POS*-tagging has been applied. After every word and punctuation character, a /CATEGORY has been appended. For example, "Ich koche gerne" is transformed to "ich/PRON koche/VERB gerne/ADV ./PUNCT". This follows the approach described by Sowmya Vajjala et al. [70] and should help capture the structure of the sentence. The following preprocessing steps were applied: lowercasing and *POS*-tagging.

The hyperparameters were tuned automatically. The FastText library¹⁹ implements this functionality²⁰. The following hyperparameters were found:

Table 4.22: Sarcasm classifier version 2: optimal hyperparameters

Hyperparameter	Value
FastText vector dimensions	353
Word n-grams	1
Min length of char n-gram	3
Max length of char n-gram	6
Loss function	softmax

¹⁸<https://www.deepl.com/translator>

¹⁹<https://fasttext.cc>

²⁰<https://fasttext.cc/docs/en/autotune.html>

Evaluation

The final classifier achieves an accuracy of 70% and a recall of 60% for the sarcasm label. Figure 4.50 represents the corresponding confusion matrix.

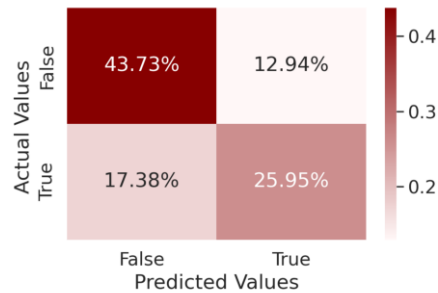


Figure 4.50: Sarcasm classifier version 2: confusion matrix

Table 4.23: Sarcasm classifier V2 scores

Metric	Score
precision	67%
recall	60%
f1-score	63%
accuracy	70%

Finally, the recall score can be improved by adjusting the decision threshold. Figure 4.51 shows how much precision would suffer by doing that:

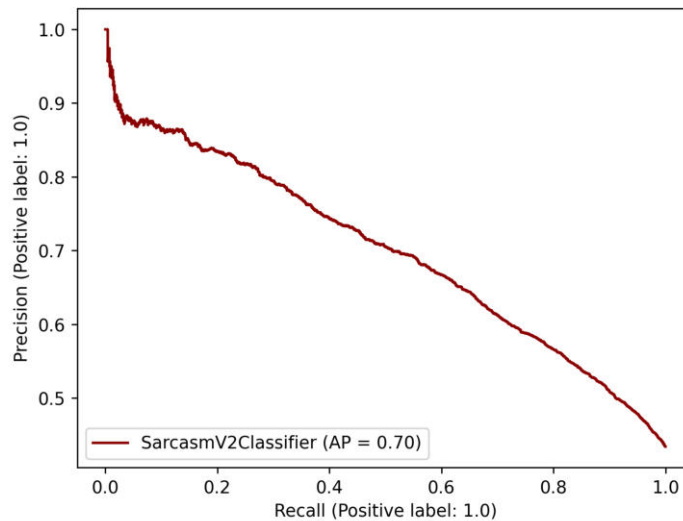


Figure 4.51: Sarcasm classifier version 2 precision vs recall

Using a decision threshold of 50%, some made-up comments were tested:

Table 4.24: Sarcasm classifier: test comments

Comment	Personal Oppinion	Pred. Prob.	Pred. Class
Freude, Kapitalismus!	Sarcastic	81%	True
Die gute alte britische Küche.	Grey Area	25%	False
Schlaf ist für die Schwachen.	Sarcastic	23%	False
Das Preis-Leistungsverhältnis stimmt bei Netflix nicht in der Schweiz :(Not Sarcastic	38%	False
Ich arbeite 40 Stunden pro Woche, um so arm zu sein.	Sarcastic	50%	True
Mir geht es großartig! Ich hoffe, dass ich mein ganzes Leben lang in der Käsekuchen Fabrik Kellnerin sein werde!	Sarcastic	48%	False
Ich liebe dieses Kleid. Das Design betont wirklich Ihr Doppelkinn.	Sarcastic	11%	False
Im Norden geht es mit viel Sonnenschein weiter, im Süden dominieren die Wolken.	Not Sarcastic	34%	False
Sie ist von Beruf her Lehrerin.	Not Sarcastic	39%	False
Was für eine Überraschung.	Sarcastic	83%	True

Conclusion

In contrast to the previous classifier, this classifier seems to be more certain in its predictions. However, there are still comments that are misjudged.

The hypothesis stated at the beginning of this subsection cannot be formally accepted. The scores of 67% precision and 60% recall from the classifier do not even come close to the 87% precision and recall from Sambit Mahapatra. Possible reasons might be that the use of an additional Reddit dataset with possibly lower quality and the translation to German.

Despite still not being a great classifier, it is noteworthy how this classifier with its simple architecture outperforms the previous classifier. A further advantage is that since there are no other features apart from FastText, predictions can be made much quicker. It looks like the saying "less is more" applies once more.

Limitations

Detection of sarcasm is hard, sometimes even for humans. Not surprisingly, the classifier does not reach high scores. It appears that the context is too important to only focus on the isolated comment. Additionally, sarcasm tends to be partly subjective. There are cases in which it is debatable whether a comment is sarcastic or not.

Furthermore, this model was trained on a translated dataset. Meaning can be lost through translation. Even if there was a perfect translator that does not lose any meaning during translation, it might still not be ideal. A sentence that has the same meaning in English and German might be taken as sarcastic to English speakers, while it seems legitimate to German speakers due to cultural differences.

Future Research

In a future work, the performance of an ensemble method using both developed sarcasm classifiers could be tested.

As Ghanem et al. pointed out, irony seems to be frequently employed in troll tweets. Because more data sets were found for sarcasm than for irony, a sarcasm classifier was created rather than an irony classifier. A future classifier could focus on irony as well.

4.4.4 State-Linked Content Meaning Classifier

Overview

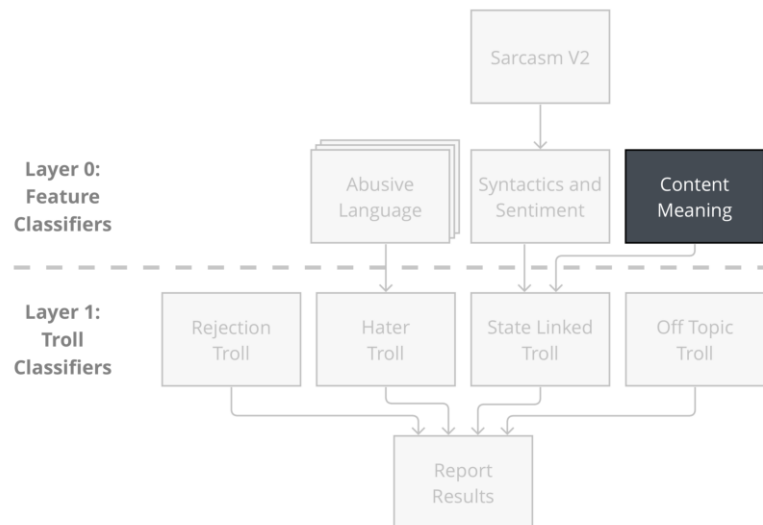


Figure 4.52: Content meaning classifier overview

Idea

Various papers have shown that state-linked trolls can also be detected on a semantic basis using *BERT* embeddings [71] [72]. These papers have used English data sets. The goal of this classifier is to show that state-linked trolls can also be effectively identified for the German language.

Hypothesis

The hypothesis for the state-linked content meaning classifier is as follows:

1. Posts of state-linked trolls can be distinguished from posts of ordinary users by their semantic meaning.

An evaluation of this hypothesis can be found in Section 4.26

Data Acquisition

The same dataset as for the syntactics and sentiment classifier was used. A detailed description of this dataset can be found in Section 4.4.2.

Feature Engineering

Table 4.25: Features overview for the state-linked content meaning classifier

Category	Features	Amount
Embeddings	SBERT, FastText	2
Total		2

First, a custom FastText²¹ model has been trained for the dataset since it worked reasonably well for the sarcasm classifier version 2 (cf. Subsection 4.4.3). After manually trying some test comments, it has become apparent that it is difficult to tell how the classifier really works. For example, this seemingly legitimate sentence "Im Norden geht es mit viel Sonnenschein weiter, im Süden dominieren die Wolken" was classified as a state-linked troll with 90% certainty. The word

²¹<https://fasttext.cc>

"dominieren" could be decisive for this score. Nevertheless, the meaning of the sentence seems to be alright and should not be associated with a state-linked troll that strongly. After manual inspection of concrete example-texts, FastText was dropped and work continued with SBERT.

The "Cross English & German RoBERTa for Sentence Embeddings" model²² was chosen as a pre-trained baseline model. This model was fine-tuned for the state-linked dataset. Figure 4.53 shows the non-fine-tuned model and the fine-tuned model.

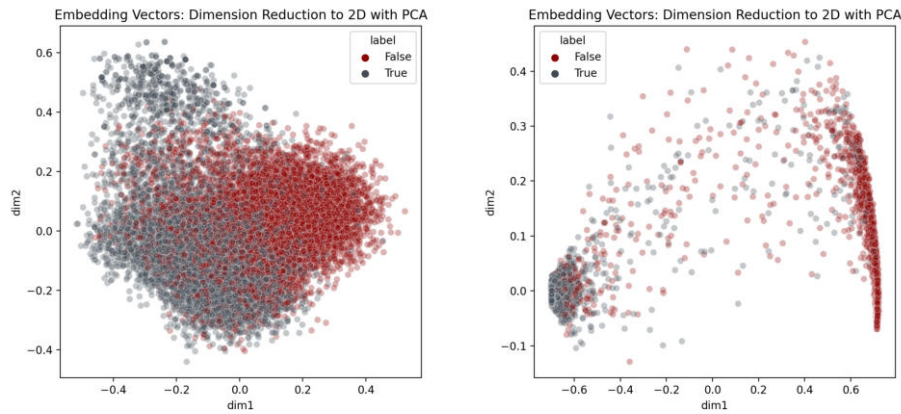


Figure 4.53: State-linked trolls non-fine-tuned model (left) and fine-tuned model (right)

For both plots, the dimensions were reduced from 768 to 2 with *PCA*. The non-fine-tuned plot explains about 7% of the total variance, while the fine-tuned plot explains 96%. It appears that fine-tuning the models helps to distinguish the two classes better.

Although the plots in Figure 4.53 appear promising, experiments with this fine-tuned model have shown that the results are rather arbitrary. For example, sentences like "Ich habe hunger", or "In den Ferien reite ich gerne Kamele" are each rated as "troll" with about 70% probability, whereas the non-fine-tuned model rated the same sentences below 10%.

To get the optimal number of principal components, a 10-fold cross validation has been performed on a subset of 20'000 training samples. *QDA* has been chosen as the classification scheme because the fitting time was more performant than other algorithms. From Figure 4.54 it appears that the more principal components are used, the better the classifier performed. In order not to use too many dimensions, the first 200 principal components were selected.

²²<https://huggingface.co/T-Systems-onsite/cross-en-de-roberta-sentence-transformer>

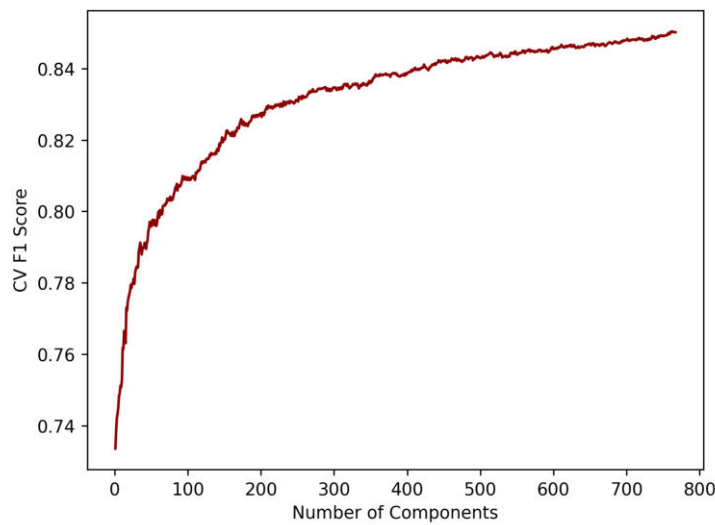


Figure 4.54: Content meaning classifier: optimal number of principal components

These 200 dimensions explain about 92% of the total variance. To find out the number of dimensions, a Scree plot was intentionally not used, since no "elbow" was visible in the curve.

Figure 4.55 shows that the principal components are not collinear. For the sake of clarity, only the first 50 dimensions have been plotted, but the remaining bars roughly have the same heights.

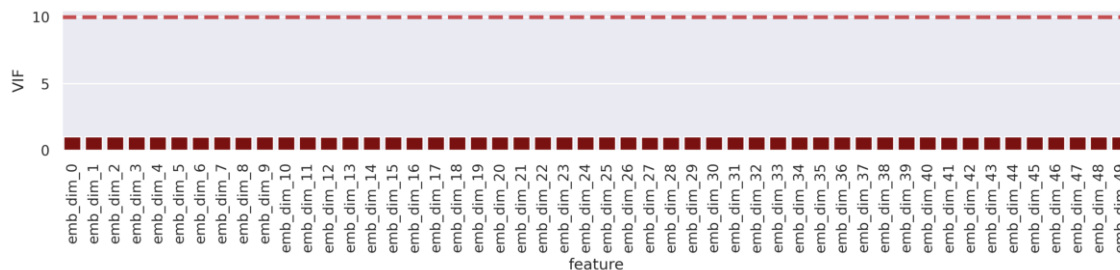


Figure 4.55: VIF plot

Modeling

Random forest appears to perform the best and is picked to train the final model with all training data:

Table 4.26: State-linked classifier: hyperparameter tuning results

Classification Scheme	Best Hyperparameters	Accuracy	Recall
<i>SVM</i>	kernel: <i>rbf</i> , γ : $1 * 10^{-3}$, C: 10	0.82	0.83
Random Forest	n_estimators: 1200, min_samples_split: 2, min_samples_leaf: 1, max_features: <i>sqrt</i> , max_depth: 90	0.84	0.86
LDA	solver: <i>svd</i>	0.79	0.83
<i>QDA</i>	reg_param: 0.01	0.80	0.82
Logistic Regression Linear	penalty: <i>l2</i> , C: 62'676	0.79	0.85
<i>KNN</i>	n_neighbors: 25	0.81	0.92

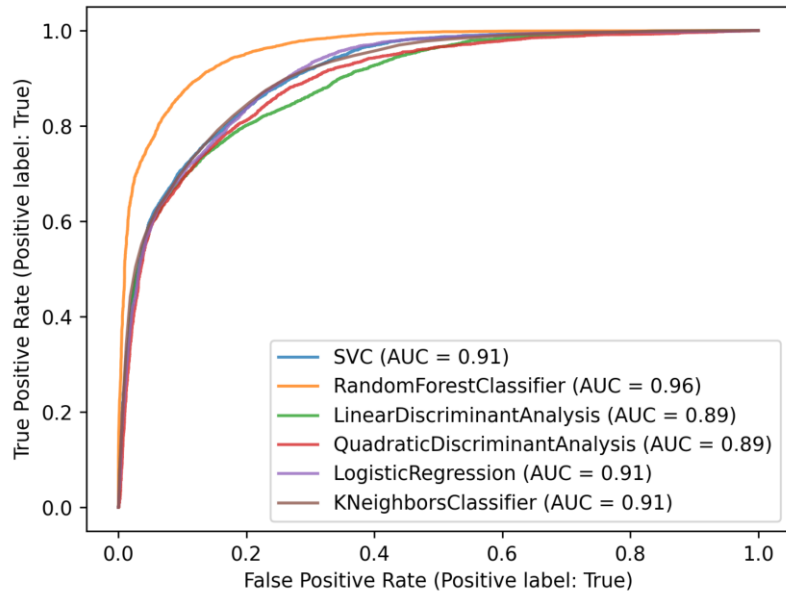


Figure 4.56: State-linked classifier content meaning ROC plots

Evaluation

The content meaning classifier achieves a lower test performance than the syntactics and sentiment classifier but it still seems to be effective. The classifier is tested on a test-set of 16'000 samples which were never used during training or hyperparameter tuning. The final model achieves an accuracy of 86% and a recall of 90%.

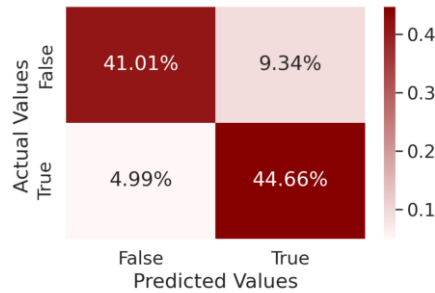


Figure 4.57: State-linked classifier contents meaning confusion matrix

Table 4.27 shows the final scores of the state-linked content meaning classifier:

Table 4.27: State-linked content meaning classifier scores

Metric	Score
precision	83%
recall	90%
f1-score	86%
accuracy	86%

Limitations

The IRA-trolls published their tweets from 2015 to 2017. The dataset of the regular tweets only contains samples from 2019 and 2020. As a result, the political discussions of the two classes do not relate that well, as there are at least two years in between. For this reason, the classifier might perform better than it would with other data sets. Unfortunately, an older dataset of regular tweet samples could not be found. As for the state-linked syntactics and sentiment classifier, it was not tested in this project how well the classifier generalizes, for example if newspaper comments are used for classification instead of tweets. This is due to the lack of similar training data for other domains.

4.4.5 State-Linked Ensemble Classifier

Overview

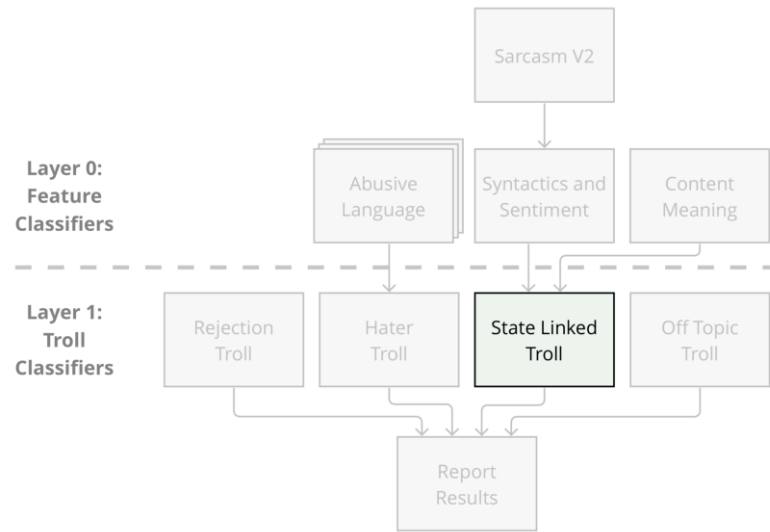


Figure 4.58: Ensemble classifier overview

Hypothesis

The hypothesis for the state-linked ensemble classifier is as follows:

1. An ensemble method will improve the classifier relative to the individual components.

This hypothesis is discussed in the following subsections.

Modeling

The two subclassifiers are now combined using a stacking and a voting strategy. The results are shown in Figure 4.59.

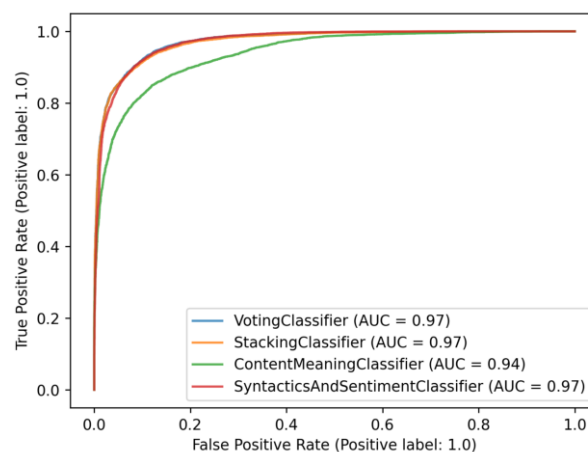


Figure 4.59: State-linked ensemble classifier ROC plots

Unfortunately, the ensemble classifier does not improve much over the syntactics and sentiment classifier. Since stacking and voting have the same performance, the simpler ensemble scheme has been chosen where the probabilistic results are averaged using Equation 4.1.

$$\hat{P} = \frac{1}{N} \sum_{i=1}^N P_i(Y = Troll|X = Text) \quad (4.1)$$

Evaluation

Putting these classifiers together by averaging the probabilities as described above, the final classifier achieves an accuracy of 91% and a recall of 92%.

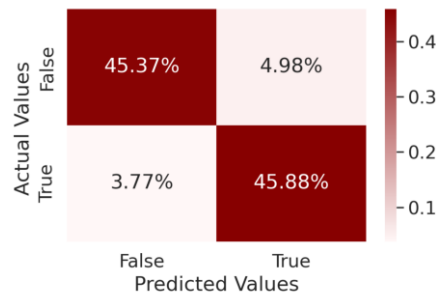


Figure 4.60: State-linked classifier final confusion matrix

Table 4.28 shows the final scores of the ensemble classifier.

Table 4.28: State-linked ensemble classifier scores

Metric	Score
precision	90%
recall	92%
f1-score	91%
accuracy	91%

While having a marginal improvement in performance over the syntactic and sentiment classifier, the ensemble classifier is likely to be more robust in the wild, since it combines syntactical features with the semantic meaning of the comments. This, however, could not be validated formally.

The following figure shows the relationship between precision and recall in case further adjustment is required.

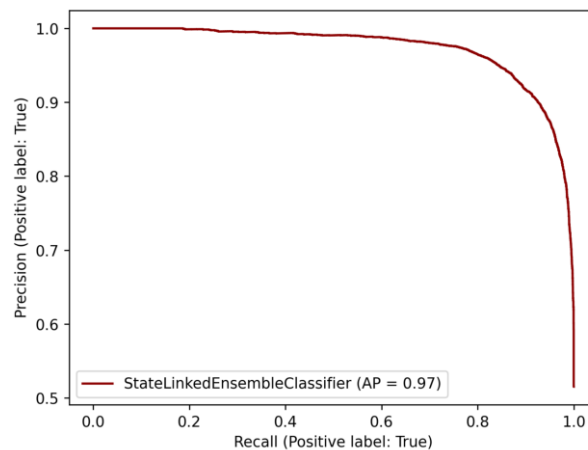


Figure 4.61: State-linked classifier precision vs recall

A few exemplary comments are shown in the following table.

Table 4.29: State-linked ensemble classifier: test comments

Comment	Syntactics and Senti- ment	Content Meaning	Ensemble
Erdogan ist es gelungen, eine stabile und kompetente Regierung zu schaffen! Siehe http://www.example.com/	64%	52%	58%
Ich mag Hunde	3%	17%	10%
@reisender9 Du bist doch auch für den Brexit oder?	4%	50%	27%
SPD geht in die Opposition: Rekordverlust für GroKo-Parteien - AfD wird Dritter	28%	46%	37%
Der Westen sollte seine Speerspitze nicht noch tiefer in das Fleisch Russlands hineinstossen. Als der momentan Stärkere sollte er jetzt auf Putin zugehen. Sonst könnte es am 9. Mai heissen: Wollt ihr den totalen Krieg? Ein entfesselter und verletzter Bär wäre für die ganze Welt eine Gefahr, wenn er nur noch wild um sich schlägt.	1%	32%	17%
Trump hat gesagt, alle Flüchtlinge seien selbst schuld.	41%	40%	41%

It should be noted that despite the high performance scores, this classifier might not be as effective as it first appears for classifying newspaper comments. The fact that state-linked trolls first try to build an audience cannot be transferred to online newspapers, since newspapers are not social media platforms. Hence state-linked social media trolls usually contain a lot of irrelevant post, while on newspapers, they have to get to the point quickly. Also, the writing style used in newspaper comments differs from the writing style used in tweets. There is no similar training data for newspaper comments available. This project therefore did not investigate how the performance of the classifier changes in a different domain.

A second issue is that the IRA dataset is not up-to-date. While this classifier might work well to identify IRA trolls from 2017, it might no longer be effective for the current trolls. Five Years is a long time in the context of social media. State-linked trolls regularly switch their behavior, which makes it crucial to train the classifier with up-to-date data. Unfortunately, more current training data was not available for this project.

4.5 Off-Topic Troll Classifier

Overview

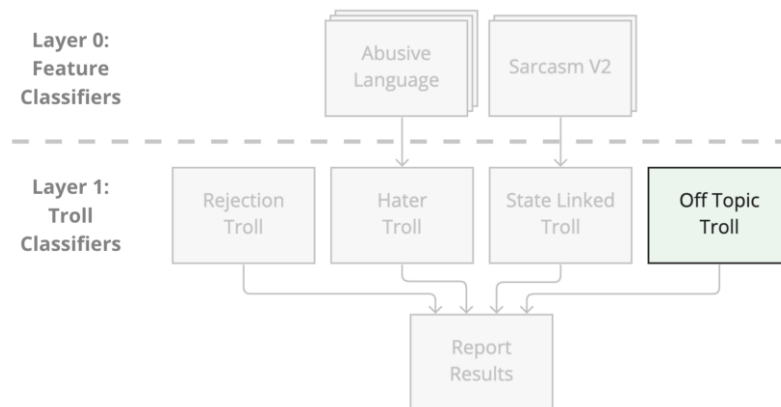


Figure 4.62: Off-Topic troll overview

Idea

Understanding the article’s comments helps to understand the public opinion. Many organizations analyze comments to improve their decision-making [73]. A study has shown that about 20% to 50% of the comments in online news are irrelevant to the article [74]. The goal of this classifier is to identify the subset of comments that have no relation to the article’s journalistic content.

Literature

Alshehri et al. created a classifier that achieves up to 92% accuracy by only comparing comments with articles [73]. Unfortunately, they used a custom created dataset, which was not publicly available and written in English.

Hypotheses

The Hypotheses for the off-topic troll classifier are as follows:

1. Off-topic comments have, on average, a lower cosine similarity to the article’s content than on-topic comments.
2. Off-topic comments have, on average, a lower cosine similarity to the other comments in the same article as on-topic comments.

The assumption for hypothesis 2 is that the majority of the commentators stay on-topic. An evaluation of those hypotheses can be found in Section 4.5.

Data Acquisition

Der Standard Dataset The only German dataset found containing the labels Off- and On-Topic was Der Standard’s One Million Posts Corpus. Despite its name, this dataset consists of only 3599 labeled samples, of which 580 are off-topic, and 3019 are on-topic. A description of this training data set can be found in Section 3.5.4.

Self-made 20 Minuten Dataset Because appropriate labeled German data is hard to find, a custom labeled dataset has been created. It consists of 3901 labeled comments, with 172 marked as off-topic (see Section 3.4).

Feature Engineering

The following table lists the initially tried features during feature engineering:

Table 4.30: Feature overview for the off-topic classifier

Category	Features	Amount
Similarities	Cosine Similarity to the Article, Maximum Cosine Similarity to the other Comments, Mean Cosine Similarity to the other Comments, Rate of Article Words in the Comment	4
Total		4

The following table shows the embedding variants considered during the feature engineering. To calculate the *TF-IDF* embeddings, the following preprocessing steps were applied: lowercasing, removal of links, lemmatization. The intention of these steps was to remove noise and use lemmatization to put the words in the same form for better word comparison.

Table 4.31: Embedding variants

Category	Dimensions
SBERT Embeddings	768
<i>TF-IDF</i> Embeddings	2'000

Alshehri et al. leveraged a similar approach of encoding the comments using the *BERT* architecture [73]. They used a pre-trained model and fine-tuned it on their manually labeled comments and articles. For the classifier created in this thesis, the "Cross English & German *RoBERTa* for Sentence Embeddings" model²³ was chosen as an alternative pre-trained model. Sentence-BERT (*SBERT*) is a modification of *BERT* that use siamese network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity [75] [47].

Different measures exist to calculate the similarity between two documents. Most methods require to first calculate an embedding vector of the documents and then compare these embedding vectors [76]. The similarity between two vectors can be calculated using the cosine similarity (see Equation 4.2). The benefit of these embedding vectors is that with clever representations, the similarity can be captured even if the two documents have no overlapping words. In this case, this applies to the *SBERT* embeddings [75] and less to *TF-IDF*²⁴.

$$\text{cosine similarity} = \cos(\theta) = \frac{\mathbf{a} * \mathbf{b}}{\|\mathbf{a}\| * \|\mathbf{b}\|} \quad (4.2)$$

TF-IDF vectors are suitable to be compared with the cosine similarity [77]. This is also true for the *SBERT* encodings [75] [78] [79].

²³<https://huggingface.co/T-Systems-onsite/cross-en-de-roberta-sentence-transformer>

²⁴*TF-IDF* appears to be more similar to a transformed one-hot encoding: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Extracting the Features from Der Standard The histograms plot of the *SBERT* embedding similarities for the on and off-topic comments shows that the data cannot be separated at all. Visualizing the similarity vs. the length of the comment does not help either. What can be observed from the second graph is that the longer comments are, the more similar they appear to be to the article. This makes sense because short comments such as "Die werden sich noch wundern" or "Also doch?" do not contain much information and therefore receive a lower similarity score.

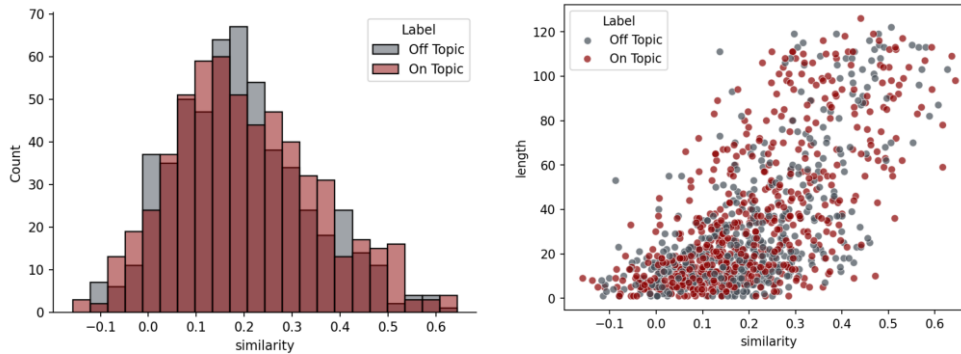


Figure 4.63: Off-Topic comments from Der Standard with *SBERT* embedding similarities compared to the article

Figure 4.64 shows that Tfidf-Vectorization pretty much yields the same result. There is a bigger accumulation on the left side of the graphs than before. This is because the Tfidf-Vectors are sparse [80]. If no words between two documents match, then the dot product in Equation 4.2 between the two vectors is zero, resulting in a cosine similarity of zero.

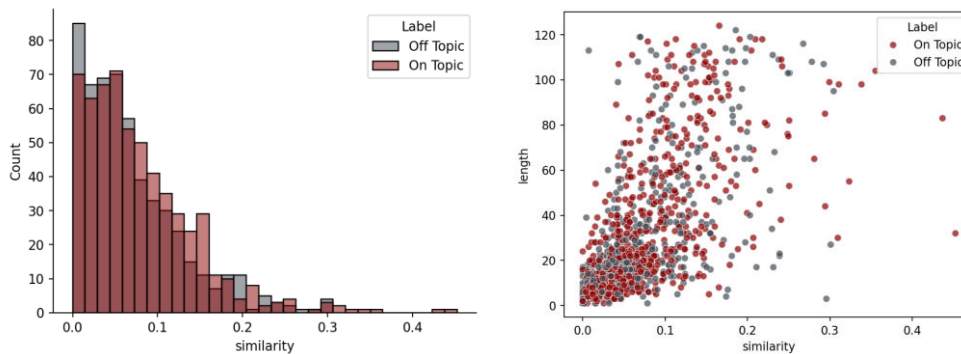


Figure 4.64: Off-Topic comments from Der Standard with Tfidf embedding similarities compared to the article

Seeing these plots puts the quality of the dataset into question. After reviewing some comments with their labels, it appears that this data set is more likely to label a comment as off-topic than we would. This aligns with the statement of Alshehri et al. where labeling off-topic comments is a difficult problem even for humans [73].

Extracting the Features from the Self-made 20 Minuten Dataset In Figure 4.65, the same procedure for the self-made 20 Minuten dataset is repeated. Although the dataset is too small to draw a conclusion, a somewhat clearer distinction between the two classes can be identified.

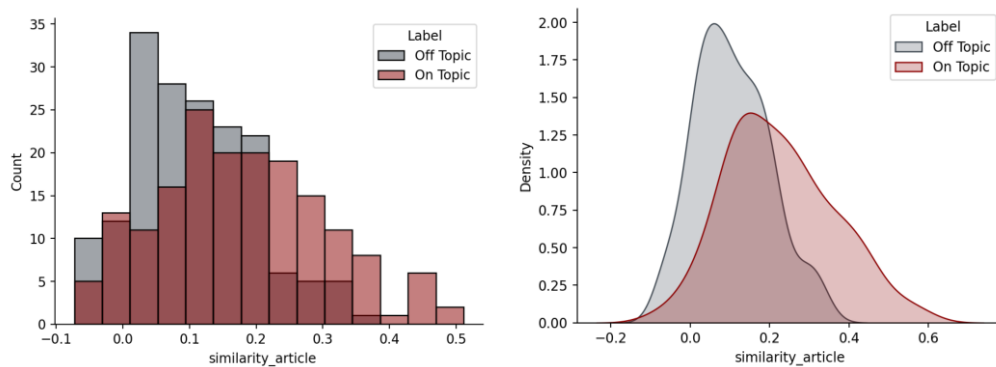


Figure 4.65: Off-Topic comments from 20 Minuten with *SBERT* embedding similarities compared to the article

A hypothesis test was performed to check whether these two distributions are significantly different.

Hypothesis Test *SBERT* Similarities between Comment and Article

Let

$$X = \text{Off-Topic Similarities}$$

$$Y = \text{On Topic Similarities}$$

be the two sample sets, then the Shapiro scores

$$\text{ShapiroScore}_X \approx 0.984, \text{ with } p \approx 0.081$$

$$\text{ShapiroScore}_Y \approx 0.988, \text{ with } p \approx 0.211$$

show, that the samples appear to be normal distributed because both p-values are > 0.05 [10], while the

$$\text{LeveneScore} \approx 15.097, \text{ with } p \approx 0.0001$$

shows, that the two variances are not similar because the p-value is < 0.05 [13]. The *ShapiroScore* and *LeveneScore* are calculated using SciPy²⁵. The sample sets appear to fit the assumptions of the *t*-test for unequal variances, according to Chapter ???. The statistical test is thus performed using Equation 1.3. The hypotheses are formulated like:

$$H_0 : \bar{X} = \bar{Y}$$

$$H_1 : \bar{X} \neq \bar{Y}$$

The following sample statistics are calculated in Python:

$$n_X = 151, n_Y = 151$$

$$\bar{X} \approx 0.103, \bar{Y} \approx 0.184$$

$$S_X^2 \approx 0.008, S_Y^2 \approx 0.017$$

Inserting the arguments into Equation 1.3 results in a *t*-statistic of

$$t = \frac{0.103 - 0.184}{\sqrt{\frac{0.008}{151} + \frac{0.017}{151}}} \approx \mathbf{-6.3}$$

²⁵<https://docs.scipy.org/doc/scipy/index.html>

At a significance level of 5%, the critical value is ± 1.969 for a degree of freedom of about 265 according to Equation 1.4. The calculated t -statistic is $-6.3 < -1.969$, thus the null hypothesis can be rejected, which implies that the two means of the distributions are significantly different and the feature is **meaningful**.

The similarities of the comment in question compared to other comments belonging to the same article were also calculated. With articles having up to 500 comments, the number of comparison comments had to be reduced due to performance reasons during training. Hence, only non-reply comments are used for comparison. This is because the dataset has shown that non-reply comments are usually longer and thus contain more meaning to extract. The total number of comparison comments was also limited to the 30 longest comments from the same article, excluding the comment in question. These 30 similarity scores are aggregated into two features: mean similarity and max similarity. Figure 4.66 visualizes these two features.

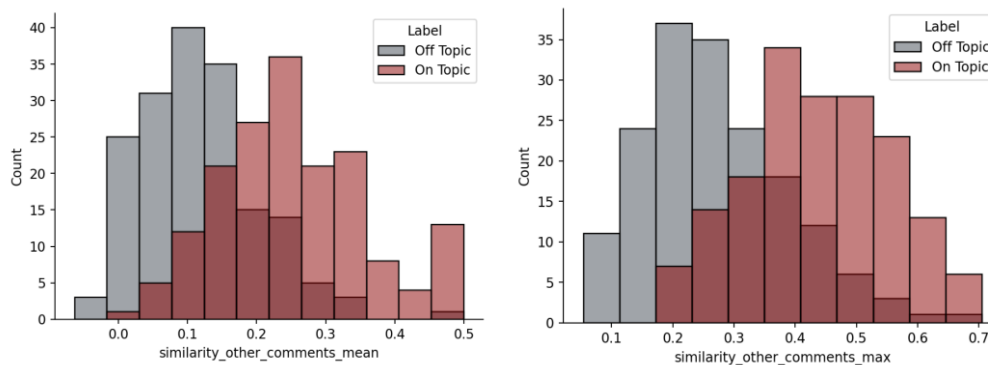


Figure 4.66: Off-Topic comments from 20 Minuten with SBERT embedding similarities compared to other comments mean (left) and max (right)

Clearly the mean and max features are highly correlated with a correlation coefficient of 0.88. Experiments with first classifiers using Logistic Regression showed that mean performed slightly better than max. Hence, the mean feature is further analyzed. No evidence could be found in literature that the arithmetic average of multiple cosine similarities is an appropriate measure. However, neither could anything be found that taking the mean should not be done. A hypothesis test was performed to evaluate the meaningfulness of this feature:

Hypothesis Test Mean *SBERT* Similarities between Comments

Let

X = Off-Topic Similarities

Y = On-Topic Similarities

be the two sample sets. The Shapiro scores

$$ShapiroScore_X \approx 0.965, \text{ with } p \approx 0.0002$$

$$ShapiroScore_Y \approx 0.123, \text{ with } p \approx 1.238 * 10^{-27}$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence, the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The hypotheses are formulated as follows:

H_0 : X and Y come from the same distribution

H_1 : X and Y come from different distributions

The Kolmogorov–Smirnov statistic calculated with SciPy²⁶

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.543}, \text{ with } p \approx \mathbf{8.979 * 10^{-21}}$$

shows that H_0 can be rejected because $p < 0.05$ [7]. Thus the two distributions appear to be significantly different and the feature is **meaningful**. The approximate value of $D_{X,Y} \approx 0.543$ is drawn into the *CDF* plot in Figure 4.67.

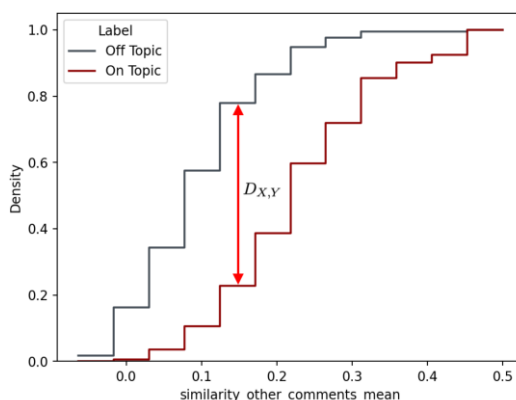


Figure 4.67: Off-Topic comments from 20 Minuten with rate of article words in comment

The features so far focus on the meaning of the comment. Experiments have shown that this method alone is not enough. Consider the following example: "Hakan Yakin überschätzt sich gewaltig." This comment is very short and does not contain much meaning that could help the *SBERT* features. However, this comment is still on-topic which can be recognized by the name which was also mentioned in the article. This is why it might be helpful to combine the previous features with a feature that compares comment and article on a per-word basis. This is done by measuring the rate of words inside a comment which also occur in the article.

$$\text{Article Words Rate} = \frac{\text{Size of Word Intersections}}{\text{Size of Comment Word Set}} \quad (4.3)$$

Size of Word Intersections indicates the intersection of words in the comment with the words in the article. *Size of Comment Word Set* represents the number of words in the comment without duplicates. This equation is similar to the Jaccard Similarity where the denominator is the size of the union of article and comment [81]. Since the article usually contains numerous words, the denominator would in most cases be a large number. For this reason, the denominator is chosen such that the lowest result value is zero and the highest is 1. The following preprocessing steps are performed to ensure that no irrelevant words are counted: removal of stopwords, removal of mentions, removal of links, lemmatization.

²⁶<https://docs.scipy.org/doc/scipy/index.html>

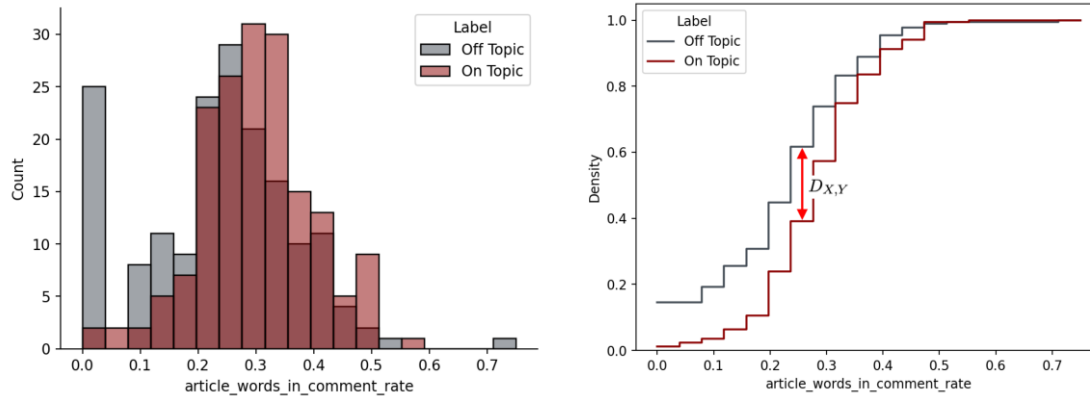


Figure 4.68: Off-Topic comments from 20 Minuten with rate of article words in comment histogram (left) and *CDF* (right)

Since this feature is calculated with Equation 4.3 which was not taken from literature, its significance needed to be validated with a hypothesis test:

Hypothesis Test Rate of Article Words in Comment

Let

X = Off-Topic Similarities

Y = On-Topic Similarities

be the two sample sets. The Shapiro scores

$$\text{ShapiroScore}_X \approx 0.952, \text{ with } p \approx 0.0004$$

$$\text{ShapiroScore}_Y \approx 0.988, \text{ with } p \approx 0.207$$

show that the X is not normal distributed because $p < 0.05$ [10]. Hence, the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The hypotheses are formulated like

H_0 : X and Y come from the same distribution

H_1 : X and Y come from different distributions

The Kolmogorov–Smirnov statistic calculated with SciPy²⁷

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.232}, \text{ with } p \approx \mathbf{0.0006}$$

shows that H_0 can be rejected because $p < 0.05$ [7]. Thus, the two distributions appear to be significantly different, and the feature is **meaningful**. The approximate value of $D_{X,Y} \approx 0.232$ is drawn into the *CDF* plot in Figure 4.68.

²⁷<https://docs.scipy.org/doc/scipy/index.html>

Modeling

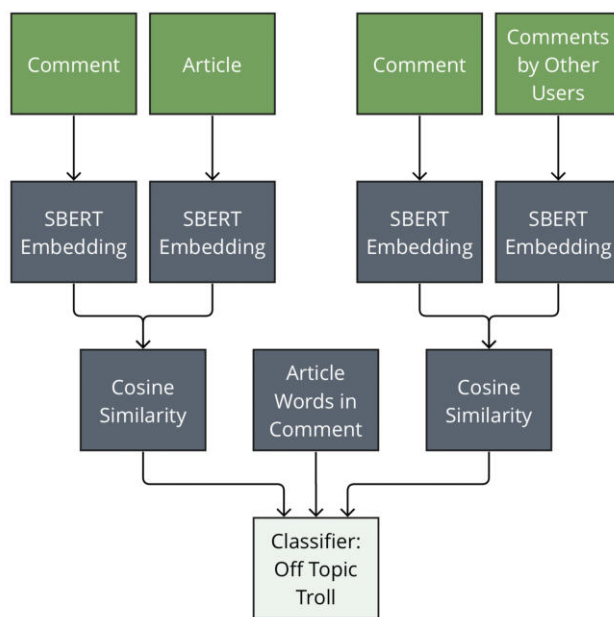


Figure 4.69: Off-Topic classifier final structure

The structure of this classifier is as follows: The comment in question is compared to the article text and its accompanying comments. For this purpose, the texts are converted into representative embedding vectors, so that the cosine similarity can be calculated between these vectors. The strategy for comment vs. article comparison is taken from Alshehri et al. [73], but the comparison between comments uses a custom approach not seen in literature by us. Additionally, the comment is compared to the article based on a word comparison.

Different classification schemes were tested. Figure 4.70 only shows three models to keep the graph readable.

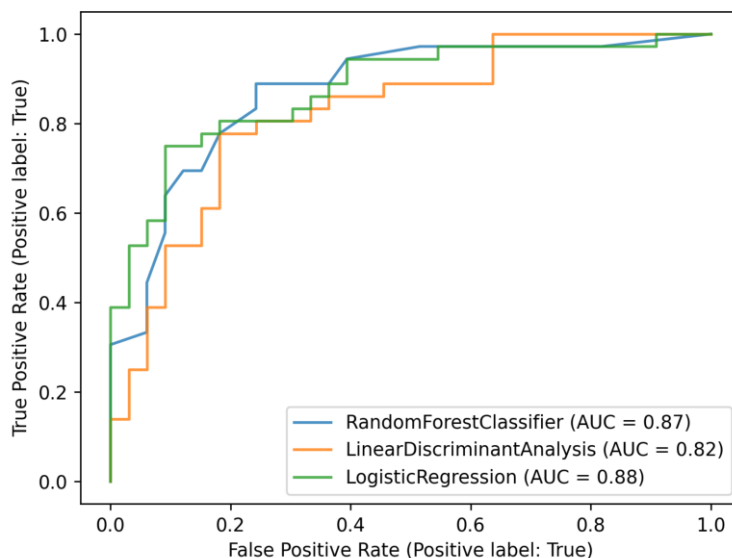


Figure 4.70: Off-topic classifier ROC curves

Logistic Regression was chosen to train the final model. In case there are no other comments in the

same article, a comparison between comments cannot be performed. Hence, an additional classifier has been trained with the same features, but without the feature "Mean *SBERT* Similarities between Comments". Consequently, this additional classifier makes its predictions based only on the similarity to the article and the rate of article words in the comment. This classifier employs Logistic Regression as well.

Evaluation

The final classifier achieves an accuracy of 80% and a recall of 78%. Figure 4.71 shows the confusion matrix and the *ROC* curve.

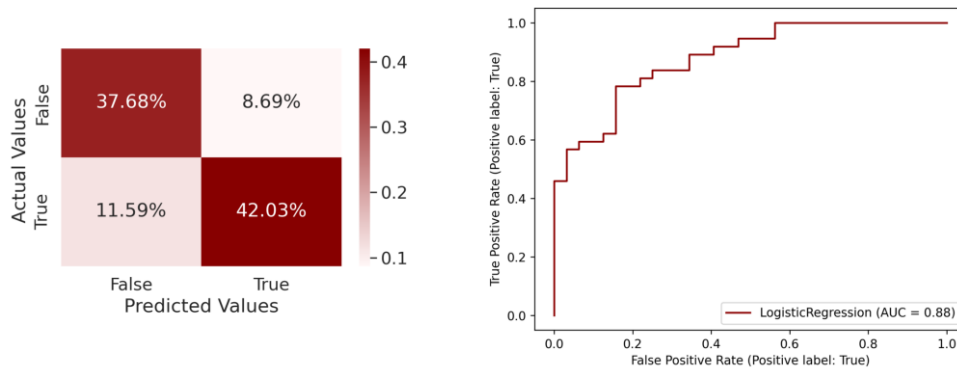


Figure 4.71: Confusion matrix (left) and ROC curve (right) for the off-topic classifier

The following table shows the scores of the final classifier.

Table 4.32: Off-topic classifier scores

Metric	Score
precision	83%
recall	78%
f1-score	81%
accuracy	80%

The second classifier, which does not compare the comment in question to other comments in the article, achieves an accuracy of 77% and a recall of 73%.

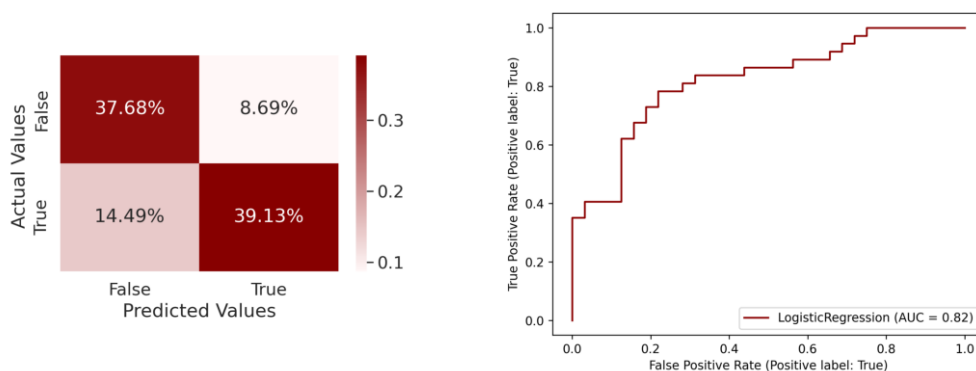


Figure 4.72: Confusion matrix (left) and ROC curve (right) for the off-topic classifier without comparison to other comments

The following table shows the scores of the simpler version of the off-topic classifier.

Table 4.33: Simple off-topic classifier scores

Metric	Score
precision	82%
recall	73%
f1-score	77%
accuracy	77%

The hypotheses for the off-topic classifier, stated at the beginning of this section, can both be accepted. It has been shown that the comments' similarity score to the article and the comments mean similarity to the other comments from the same article have significantly different distributions when trolls and non-trolls are separated.

The classifier does not reach Alshehri et al.'s high accuracy of 92%. One possible explanation might be that Alshehri et al. performed a fine-tuning on their *BERT* transformer [73]. Another reason could be the different sizes and quality of the datasets.

Limitations

If the commentary relates to the same overarching topic, such as foreign policy, then the similarity score is most likely high. In this case, the comment may still be off-topic, but the distinction is more subtle. For example, the article could deal with political relations with France, while the comment focuses on the elections in England. The classifier does not work well in these situations.

The assumption is that the majority of users in the comment section are ordinary users. If that is not the case, then the idea of measuring the similarity against other user comments becomes counterproductive because the classifier could be tricked by posting a lot of similar off-topic comments.

In this classifier, the closeness of two texts has been measured using embedding vectors. However, there is also another approach using knowledge-based measures that omit vectors entirely and promises good performance [76]. A combination of *SBERT* embeddings and knowledge-based measure would be interesting to explore in a future work.

4.6 Metadata-Only Approach

4.6.1 Introduction

The previously described classifiers analyse the text of a comment to predict whether or not a comment belongs to a troll category. The two classifiers described in this section analyze the metadata of a comment to make a prediction. One metadata classifier works with training data from "Der Standard" while the other metadata classifier works with data from "20 Minuten". While the classifier for "Der Standard" produced mixed results, the classifier for "20 Minuten" performed badly.

4.6.2 Similar approaches in literature

There are multiple existing projects that use metadata for the detection of troll comments. Two of these projects were used as reference and as a benchmark to compare the results of this classifier:

- **Hunting for Troll Comments in News Community Forums** is a study about comments for a Bulgarian newspaper. The authors used a balanced set with over 1000 troll-comments and 1000 non-troll comments as training data. A Logistic-Regression Classifier was trained and achieved an accuracy of over 80%. Metadata features were only one part of the features used for this classifier. Most features were taken from the unstructured content of the comment (like bag of words, bag of stems. . .). The evaluation has shown that the metadata features were less important in general, overall the metadata features had an accuracy of 71% [39].
- **The Metadata Troll Detector** was a semester thesis at ETH Zürich by Stephan Dollberg. As training data, he used a total of 400 comments from Reddit being classified as "troll" or "non-troll". A support vector machine trained on the data achieved an average f1 score, recall and precision of about 70%. The features used in the ETH study are similar to the features used for this project [29].

Based on those existing results, a recall rate and precision rate of 70% is considered a success for this project. For this type of classifier, the precision rate is considered to be more important than the recall rate. This means that the number of comments that are falsely accused of trolling should be minimized.

4.6.3 Comment Metadata Classifier for "Der Standard"

Data Acquisition

To develop this classifier, labeled training data from the "one million corpus project / der Standard" was used to detect hater trolls (see Section 3.7). A random balanced dataset with 1000 comments was created. The dataset is relatively small because only users that have written more than one comment were considered for the analysis. This decision was done because otherwise some of the metrics could not be calculated (for example: "seconds passed until the next comment" - see below for a full list of features). This classifier does not differentiate different categories of trolling. All comments that were labeled as either "Discriminating", "Off-Topic" or "Inappropriate" in the original dataset are seen as "troll comments". Comments that do not have one or more of those labels are seen as "non-troll comments".

Data Understanding

Visualizing the dataset and different features has shown that troll-users indeed are more active than normal users as stated in the studies mentioned above. Trolls tend to write more than one comment per article, and they often write many comments during a short period of time. With these findings and by reading studies about similar classifiers, a list of metadata features was extracted. For each feature that was used in one or more of the analyzed research projects, a source is given below.

According to the paper that was published together with the "der Standard" dataset, the comments were not always selected randomly [37]. At first, comments were selected randomly for annotation.

Later, the comments to annotate were selected in a way to increase the number of problematic comments.

Furthermore, some comments were never published publicly or later removed (because they violate the guidelines). This means that some features are biased:

- The overall number of reactions is smaller for some troll-comments because they were removed after some time by the moderators.
- The overall number of replies is smaller for some troll-comments because they were removed after some time by the moderators.

The biased features were not used for the final classifier.

Feature Engineering

Table 4.34: Features of time behavior

Nr.	Feature	Description and Source	Used?
1	Number of comments from the same user on the same article	Trolls are more likely to comment multiple times on the same article [1].	Yes

Table 4.35: Features for user interaction

Nr.	Feature	Description and Source	Used?
2	Total number of replies	Troll comments usually receive more replies than the average comments because they are provoking reactions. This feature has shown to be significant in similar studies [29] [1].	No (Biased)
3	Average Reply Word Count	A high average word count per reply is an indication of a meaningful discussion. This feature was also used in a similar study [29].	No (Biased)
4	Number of Upvotes	This feature turned out to be very significant in other studies [29] [1].	No (Biased)
5	Number of Downvotes	This feature turned out to be very significant in other studies [29] [1]. Troll accounts usually receive more downvotes from non-trolls. Some literature also states that professional trolls receive less downvotes than unprofessional trolls [82] [83].	No (Biased)
6	Percentage of positive reactions	Whereas the previous two features represent absolute numbers, the up- and downvotes can also be represented as a fraction of both values. This feature has the advantage that it is not influenced by the popularity of an article and other influences [1].	No (Biased)
7	Is reply comment (0 or 1)?	This feature was used in similar projects [83] [82].	Yes

Table 4.36: Features of comment content

Nr.	Feature	Description and Source	Used?
8	Automated Readability Index	Research has shown that trolls often write in a less understandable language [1].	Yes

Continued on next page

Table 4.36 – continued from previous page

Nr.	Feature	Description and Source	Used?
9	Number of characters	Trolls are more likely to write short comments. Larger comments are usually more insightful. Trolls, however, want to provoke, which is done more efficiently using short and precise comments [1] [84].	Yes
10	Number of Words	Similar to the previous feature, troll comments tend to have fewer words. Troll comments often only contain a statement but no further facts, references, or explanations [29] [85].	Yes
11	Average length of words	Longer words are often more meaningful. Thus, troll comments are suspected to have a lower average word length [85].	Yes
12	Percentage of all capital words with more than three letters	Especially spam messages are often written in capital letters. For internet comments, writing words in capital letters is sometimes used to express anger [39] [29] [84] [85]. Capital words with less than four letters are not counted in this project because this would count many abbreviations as well.	Yes
13	Cosine similarity between comment and article headline / article description	A low similarity might be an indication for an off-topic troll [1] [82] [83].	Yes
14	Contains emojis (0 or 1)	The use of emojis can be an indication that a comment was written on a smartphone. Professional trolls, however, are suspected to write their comments with a computer keyboard and thus use less emojis. This feature was also used in similar projects [39] [82].	Yes
15	Number of exclamation marks	This feature was used in similar projects [39] [82].	Yes

Table 4.37: Other features

Nr.	Feature	Description and Source	Used?
16	Number of identical comments in the data set	Multiple comments might be identical (for the same article or different articles) because a troll copy-pastes the same content [86].	Yes

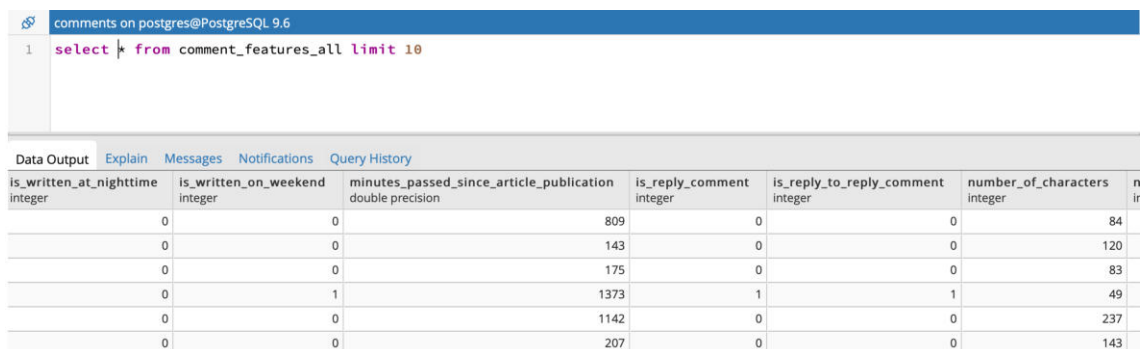
Some additional features not found in existing literature were brainstormed:

Table 4.38: Brainstormed features

Nr.	Feature	Hypothesis & Validation	Used?
17	Number of comments from the same user during 12 hours before and 12 hours after the comment	Hypothesis: Trolls are suspected to have a higher overall activity during 12 hours before and 12 hours after a comment. Validation: The hypothesis is accepted. In the training data, troll users have a median value of 2 comments within 24 hours, while non-troll users have a median value of 1. A full hypothesis test that compares the distribution can be found in section C.1	Yes
18	Seconds passed since article publication	Hypothesis: Troll users publish their comments more shortly after the article publication than non-troll users. Validation: The hypothesis is accepted. The median value of this feature for troll comments is 5.5 hours and 7.3 hours for non-troll comments. A full hypothesis test that compares the distribution can be found in section C.2	Yes
19	Seconds passed since the previous comment from the same user	Hypothesis: Troll users have a shorter timespan between a comment and the previous comment. Validation: The hypothesis is accepted. A full hypothesis test that compares the distribution can be found in section C.3	Yes
20	Seconds passed until the next comment from the same user	Hypothesis: Troll users have a shorter timespan between a comment and the next comment. Validation: A full hypothesis test that compares the distribution can be found in section C.4	Yes
21	Average sentence length	Hypothesis: Trolls make shorter sentences. Validation: This hypothesis is refused. A full hypothesis test that compares the distribution can be found in section C.5.	No

Data Preparation

Since the classifier works with data from the database, a database view was created that not only contains the comments but also some pre-calculated features per comment. This way, the extracted features can not only be used for the classifier, but also later to be shown in the application proof of concept.



```

1 select * from comment_features_all limit 10

```

is_written_at_nighttime	is_written_on_weekend	minutes_passed_since_article_publication	is_reply_comment	is_reply_to_reply_comment	number_of_characters	n
integer	integer	double precision	integer	integer	integer	ir
0	0	809	0	0	84	
0	0	143	0	0	120	
0	0	175	0	0	83	
0	1	1373	1	1	49	
0	0	1142	0	0	237	
0	0	207	0	0	143	

Figure 4.73: Feature extraction using SQL

Features that could not be expressed using SQL-Syntax were added in Python. From balanced dataset of 1'000 comments, 700 comments were used as training data and 300 comments were used as test data.

The features were scaled using python's `StandardScaler`. `None`-values were replaced by the median-value for the whole column, since Scikit-learn models do not support `None`-values.

Modeling

Multiple binary classifier algorithms were trained and optimized using Scikit-learn. The comparison using a ROC curve shows that a Random Forest Classifier overall achieved the best result and was therefore selected for further optimization:

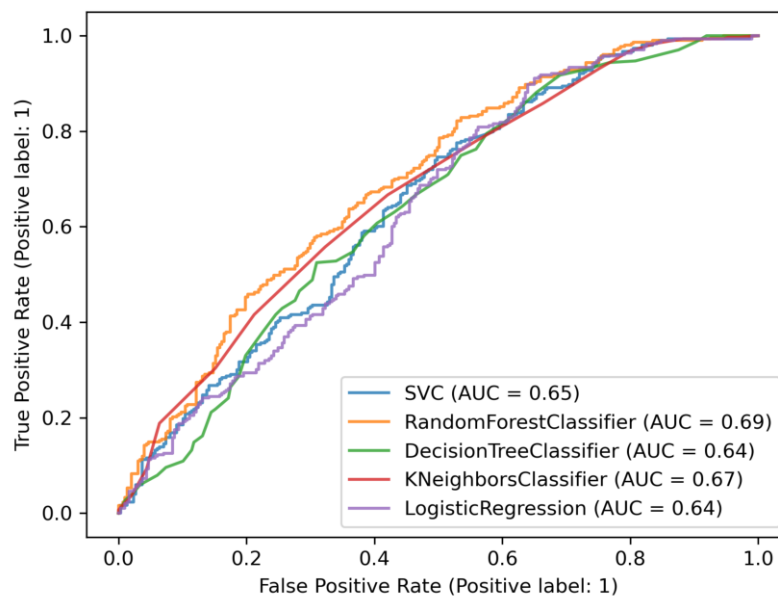


Figure 4.74: ROC curve for different metadata classifiers

Optimization

An exhaustive grid search was used to train multiple random forest classifiers (1000 classifiers in total with 3-fold cross-validation for each) with different hyperparameters. This step increased the precision and recall a little bit. The grid search was configured to find parameters that optimize precision (not accuracy).

A manual selection of features did not increase the scores. The random forest classifier achieved better results by automatically choosing the best features itself – most likely because the number of features available is relatively small.

The importance of the individual features that were selected by the random forest classifier were surprising. The three most important features are:

- seconds passed until the next comment from the same author
- minutes passed since article publication
- seconds passed since the previous comment from the same author

Evaluation

Here are the reported scores for the classifier:

Table 4.39: Scores for the metadata classifier (Der Standard)

Metric	Score
precision	62%
recall	70%
f1-score	66%
accuracy	63%

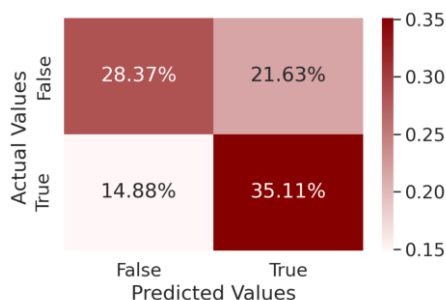


Figure 4.75: Confusion matrix for the metadata classifier (der Standard)

This classifier has a worse performance compared to the "metadata troll detector". A reason for the lower scores are the biased features that were not included in the classifier. The same classifier trained with the biased features received similar results with a recall and precision of 70%.

In comparison to the paper "Hunting for Troll Comments in News Community Forums", the classifier got a worse performance as well [39]. A part of the difference might be explained because this classifier focusses on metadata features while the classifier from the paper also includes much more unstructured features as well.

Conclusion and Limitations

The results from this classifier are mixed. However, analyzing similar projects and studies has shown that the metadata can indeed be an indicator to detect troll comments. The main problem for this classifier was the lack of information in the training data. Since most inappropriate comments from "Der Standard" were either never published or later removed, many features such as the amount of likes and dislikes were not available for usage.

4.6.4 Comment Metadata Classifier for "20 Minuten"

A metadata classifier was also implemented for 20 Minuten comments but the accuracy was only 57%. Therefore, the classifier was not included in the final application. A description of the metadata classifier for 20 Minuten comments can be found in the appendix in Section A.3.

4.7 Computing Performance Challenges

4.7.1 Computing Performance Optimizations

Performance was a critical aspect of this work. Measures had to be taken since the training of the classifiers needed much computing resources.

Parallelization was an essential part of the development of the classifiers. Often, multicore parallelization could be activated with the `n_jobs` parameter from the `sklearn` library²⁸. This worked especially well for random forests, which have been used extensively in this thesis. The `SentenceTransformers`²⁹ library took advantage of the parallelization potential of graphics card, which proved useful for encoding large amounts of text for *BERT*-embeddings. During training of the classifiers, it was crucial to activate parallelization in order to save time. But as parallelization is not advantageous for only small amounts of data, it had to be turned off on the production server.

For classifiers with many individual features, such as the state-linked classifier, it usually took a long time to gather all the features. This was noticeable during the feature engineering as well as in production. For some classifiers, it was therefore important to first create the design matrix and save it as a *CSV* file, so that the training step of the classifiers did not suffer from the computationally intensive feature calculations.

In general, it quickly became clear that the local computing power was not sufficient. Google Colab³⁰ was a valuable tool that allowed results from experiments to be obtained more quickly. The Colab platform was upgraded to access more memory (up to 27 GB) and faster graphic cards (e.g. the Tesla P100). The downside of Google Colab was that the duration of the runtime instances was arbitrary at times. Occasionally, they would run for 20 hours and other times they would stop after only 3 hours.

One issue on the deployed *API* server was the instantiation of the classifiers. Initially, the classifiers were re-instantiated for each query, which resulted in an excessively long response time. To reduce response time, the classifiers had to be cached. The biggest improvement was seen on the hater classifier, where the response time dropped from 10 seconds to 200 ms. Furthermore, to keep the database queries performant, indexes had to be created. This was necessary due to the large number of comments stored in the database.

4.7.2 Storage Space and Memory

During runtime, all classifiers together require about 12 GB of memory. Most of it is used by the random forests classifiers. Because of this, the server could not always be fully started locally, which is why detours had to be taken during development. When training neural networks with many features and data samples, the *RAM* was exploding as well because of the quickly growing design matrices. The embedding vectors calculated on the graphics card by the `SentenceTransformers` library also caused some problems because the graphics card ran out of memory. This was especially a challenge when fine-tuning a *BERT*-architecture when the amount of memory quickly exceeded 16 GB. To counter this, the batch size had to be reduced during training.

²⁸<https://scikit-learn.org/stable/computing/parallelism.html>

²⁹<https://www.sbert.net>

³⁰<https://colab.research.google.com>

CHAPTER 5

Application Proof of Concept

Developing an *API* and a frontend was not the main goal and focus of this project. The proof of concept was created to demonstrate the ability of the classifiers for lay users.

5.1 Functional Requirements

According to the task description, the analysis from the classifiers should be presented in an easy-to-use and easy-to-understand form.

There are two ways that allow a user to analyze a comment:

- The user provides a new comment text for analysis
- The user analyzes a comment from the explorable dataset

The first option works with those classifiers that do not require having additional metadata about a comment. The second way works with all classifiers (including the metadata classifier and the off-topic classifier). The requirements result in the following use cases:

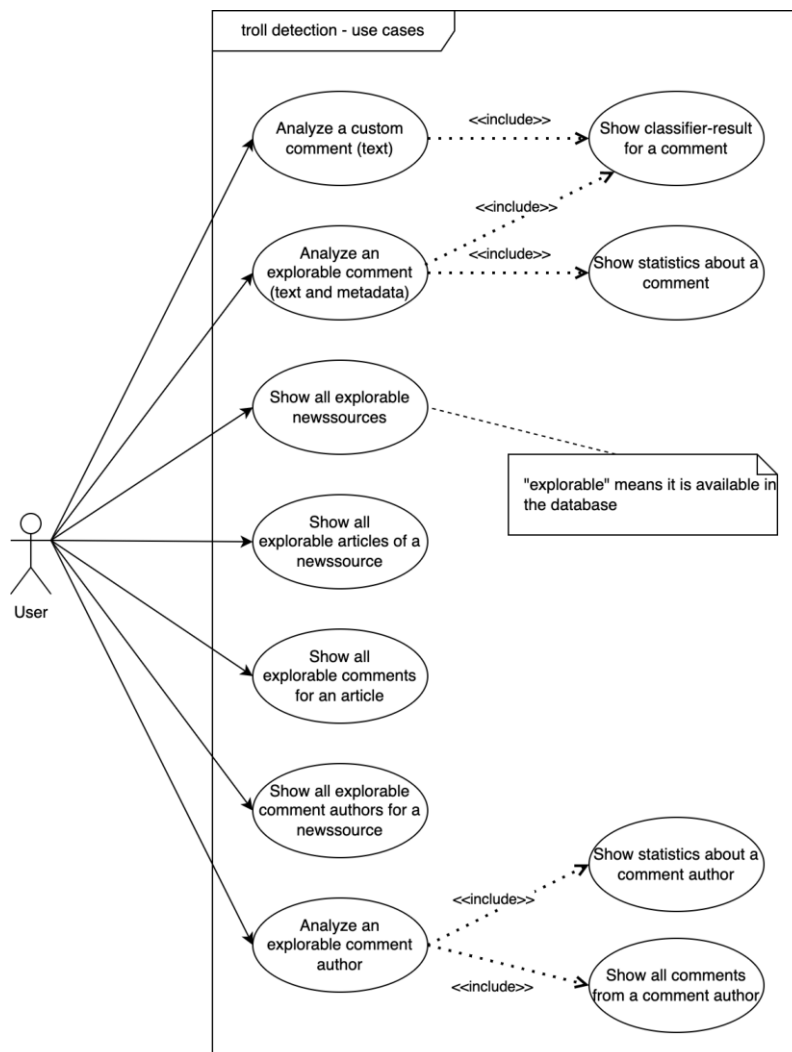


Figure 5.1: Use cases

On the one hand, the user interface gives the user the possibility to run the classifiers. On the other hand, it also allows users to access additional information that is not available on the newspapers’ websites: For example, users can view all comments from a particular comment author, or see how active a comment author is. This should enable users to detect suspicious activity that has not been detected by the classifiers.

5.2 Non-Functional Requirements

Requirement Compliance: The contract with 20 Minuten is respected. Sensitive data is not displayed publicly.

Result Fulfilled. The application is password protected and only self-collected data is shown.

Requirement Installability: Frontend and *API*-Backend are independent from the runtime environment.

Result Fulfilled. The frontend and the *API* run in a Docker-Container

Requirement Maintainability: Backend and frontend can be updated independently of each other.

Result Fulfilled. The frontend and the backend are two separate projects that run independently and only communicate over the *API*.

Requirement Maintainability: The classifiers are built in a modular way and can easily be re-used for other projects (no close coupling).

Result Fulfilled. The classifiers have no dependency to frontend- or *API*-specific modules and can therefore easily be moved to other projects.

Requirement Operability: The *API* provides the information in an interoperable format (preferably *JSON*).

Result Fulfilled. The responses are provided as *JSON*.

Requirement Performance: Displaying a list of newssources, articles, comments, or comment authors takes less than two seconds.

Result Fulfilled.

Requirement Performance: Analyzing a comment takes less than 10 seconds.

Result Not Fulfilled. On the deployment server, first results of the analysis are displayed after 2 seconds. After 10 seconds, 3 out of 6 classifiers display a result. After 25 seconds, all classifiers display a result.

Requirement Security: All queries to the database are protected against *SQL* injection attacks.

Result Fulfilled.

Requirement Security: All data from the database and from the user (article headlines, comment content, usernames, ...) is escaped before being displayed in the frontend to prevent cross-site-scripting attacks.

Result Fulfilled.

5.3 Development Setup

5.3.1 Overview

A virtual Ubuntu Server is provided by OST. Code is pushed to a GitLab instance hosted by OST as well.

University Infrastructure

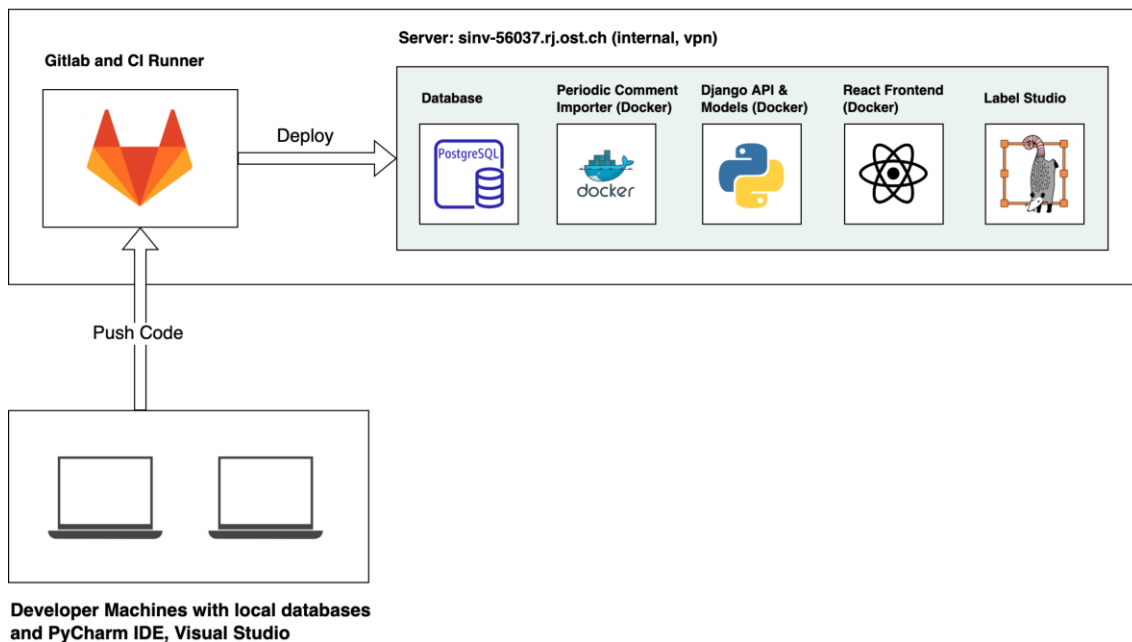


Figure 5.2: Hardware infrastructure for the project

5.3.2 Backup plan

Git is used as version control system which means that the code is redundantly stored on GitLab and on the developer's machines. Database backups are automatically created periodically and sent to two private servers.

5.4 Backend / API

5.4.1 Overview

The *API* provides the endpoints to navigate through the explorable dataset and to analyze comments:

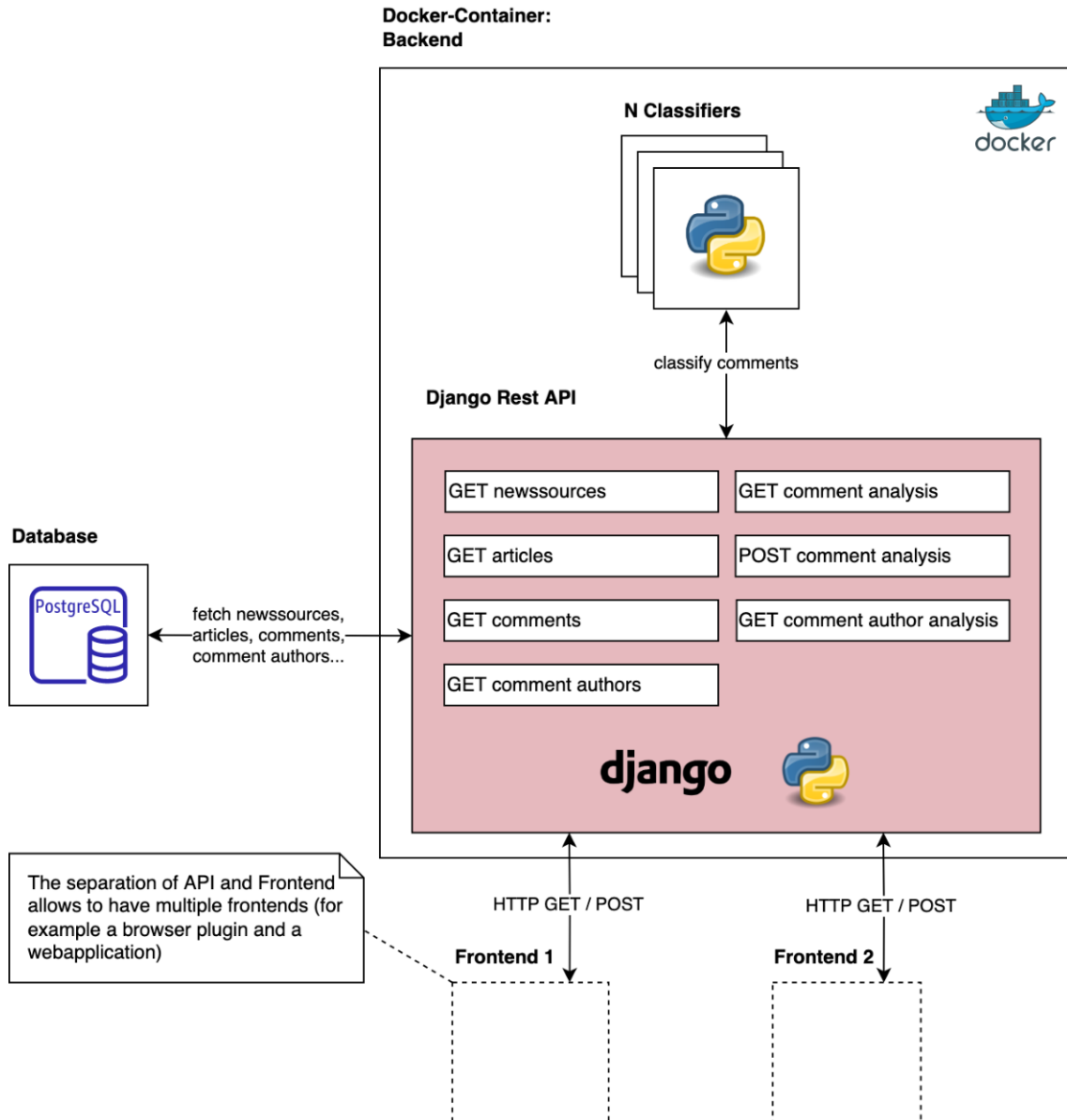


Figure 5.3: Overview of the API

5.4.2 Architectural Decisions

Client-Side Rendering or Server-Side Rendering

The *API* was built independently of the frontend. Since the frontend is not the main focus of this project, the client-side-rendering approach was chosen. The *API* provides all information using the *JSON* format, which makes it more interoperable and not coupled to a specific kind of frontend.

Choice of a Web Framework

Since the classifiers were all written in python, it made sense to build the *API* using python too. Several web frameworks were evaluated:

Table 5.1: Comparison of technologies for the API

Framework	Advantages	Disadvantages
Django	open-source, large community, much functionality comes out of the box, good IDE support	large framework
Flask	minimalistic framework, new, IDE support	less functionality, small community
Bottle.py	very lightweight, extensible	no IDE support
web2py	IDE support	less intuitive documentation, small community

After weighing the pros and cons, it was decided to go with Django. A main reason was the large community.

5.4.3 Endpoints

The *API* provides the following endpoints to navigate through newssources, articles and comments and to get statistics about comments and comment authors:

- GET `api/newssources`
- GET `api/newssources/<newssource_id>/articles`
- GET `api/newssources/<newssource_id>/articles/<article_id>/comments`
- GET `api/newssources/<newssource_id>/articles/<article_id>/comments/<comment_id>`
- GET `api/newssources/<newssource_id>/users`
- GET `api/newssources/<newssource_id>/users/<user_id>`

For a given comment, all the classifiers can be invoked individually:

- POST `api/newssources/<newssource_id>/articles/<article_id>/comments/<comment_id>/metadata_classifier`
- POST `api/newssources/<newssource_id>/articles/<article_id>/comments/<comment_id>/off_topic_classifier`
- POST `api/newssources/<newssource_id>/articles/<article_id>/comments/<comment_id>/hater_classifier`
- POST `api/newssources/<newssource_id>/articles/<article_id>/comments/<comment_id>/state_linked_classifier`
- POST `api/newssources/<newssource_id>/articles/<article_id>/comments/<comment_id>/sarcasm_classifier`
- POST `api/newssources/<newssource_id>/articles/<article_id>/comments/<comment_id>/sentiment_classifier`

- POST `api/newssources/<newssource_id>/articles/<article_id>/comments/<comment_id>/rejection_classifier`

Some of the classifiers can also be invoked by providing a custom text in the POST body:

- POST `api/off_topic_classifier`
- POST `api/hater_classifier`
- POST `api/state_linked_classifier`
- POST `api/sarcasm_classifier`
- POST `api/sentiment_classifier`
- POST `api/rejection_classifier`

Also, additional text features can be queried:

- POST `dapi/text_features`

A full description of the endpoints including the available query-parameters and the structure of the responses can be found on GitLab¹.

The *API* is explorable when accessed via web-browser:

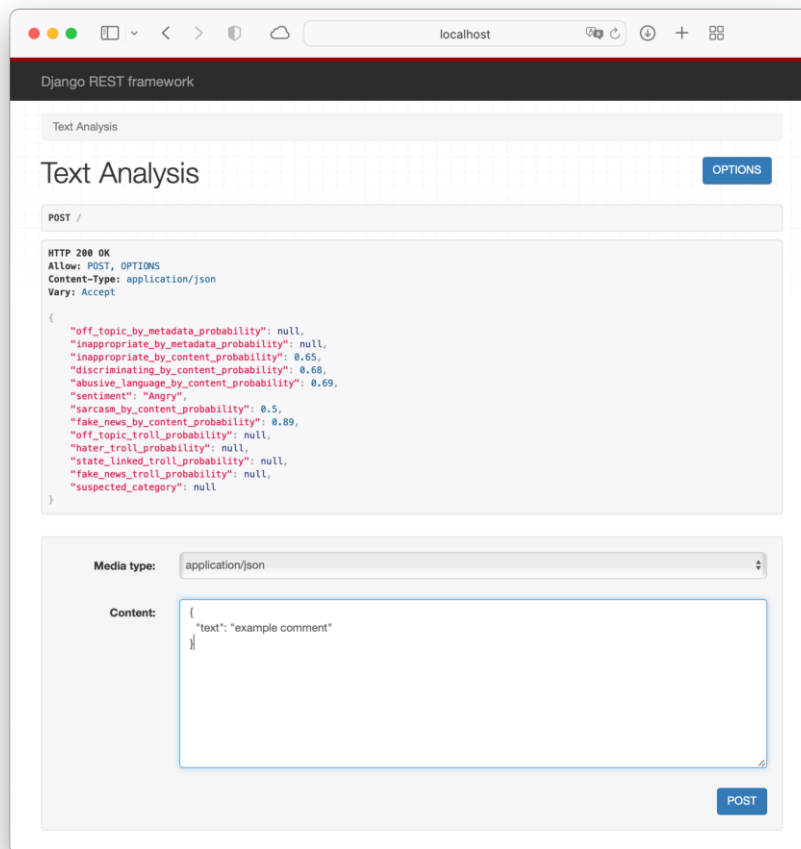


Figure 5.4: Explorable API

¹<https://gitlab.ost.ch/ba-troll-detection/troll-api>

The endpoints support paging to limit the number of entries in the response. The caller of the *API* can configure the number of results per page and the current page via query-parameters. For example:

```
api/newssources/20%20Minuten/articles?page_size=3&page=2
```

5.4.4 Implementation

The Django Rest Framework² was used to provide the *API*. Since the structure of the database does not correspond exactly with the models used in the backend, raw queries were used to query the database. Django converts the data from the database into python model classes. The models are then enriched with further information, serialized and sent as response.

To analyze a specific comment, the comment and additional metadata is first fetched from the database. Afterwards, each classifier is called using the comment as parameter. The results from the classifiers are collected, formatted and sent as a response.

5.4.5 Code Quality

- Each *API*-Endpoint is tested using automated integration tests.
- Python documentation strings were used.
- An installation guide can be found in the appendix of this thesis.

5.4.6 Deployment

The backend (*API*) is dockerized to make the deployment easier. The application can be run locally or on the server using different environment keys. `pip` is used to install the python requirements, which are documented in a `requirements.txt` file.

²<https://www.django-rest-framework.org>

5.5 Frontend

5.5.1 Creation of the Frontend

The user interface is considered a functional proof-of-concept. It allows lay users to experiment with the classifiers created for during this thesis. Also, the webpage presents additional metadata and statistics about the comments. User can either analyze their own comments or browse through existing comments from 20Minuten.ch or "Der Standard".

Different technologies were used to implement the frontend:

- **Tailwind.css:** A collection of .css files to save time styling the application
- **React:** To build reusable components
- **Typescript:** To access the API and convert the results
- **Blender:** To create an animated 3D newspaper
- **Nginx:** To provide basic password authentication

To maintain the code quality, an automated linter was used. The code is organized in different encapsulated modules.

In a first step, the backend / API was mocked. Later, the mock was replaced with the real backend-implementation. **The application was developed and tested to work on Google Chrome³ and Microsoft Edge⁴ Desktop Browsers.**

5.5.2 Result

This section presents the functionality and screenshots from the user interface. The main page shows a title and a navigation bar that allows a user to switch to the analysis or explore page:



Figure 5.5: Screenshot: main page

Troldejæger stands for troll hunter in the Danish language. Originally, a troll is a being in nordic folklore [87].

³<https://www.google.com/chrome/>

⁴<https://www.microsoft.com/en-us/edge>

On the analysis card, a textfield is provided to enter custom comments:

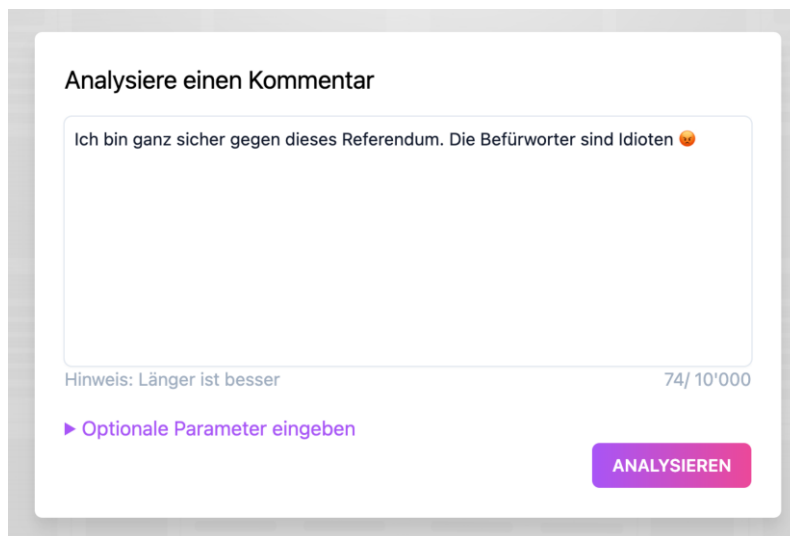


Figure 5.6: Screenshot: analyzing a comment

After clicking on "ANALYSIEREN", the results from the classifiers is shown:

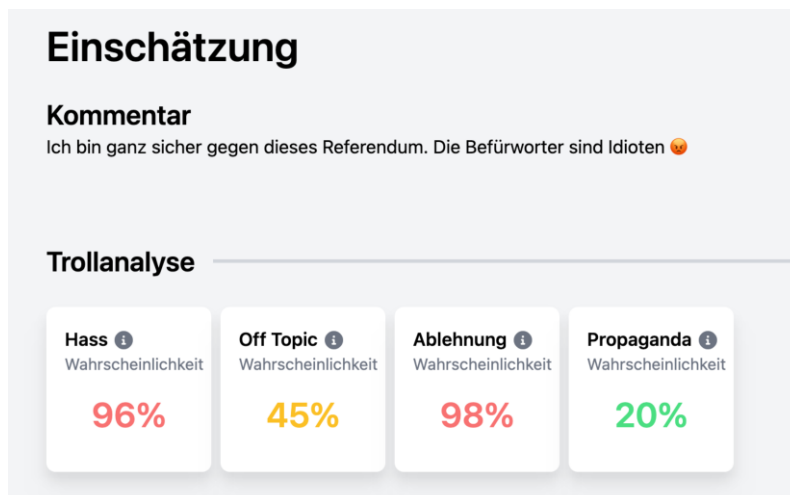


Figure 5.7: Screenshot: an analyzed comment

The results include probabilities for the three defined troll categories (Hate troll, Off-topic troll and state-linked propaganda troll) as well as the probability for a comment to be rejected by 20 Minuten moderation.

The results displayed depend on the information that was provided about the comment. For example, for comments from "Der Standard", an additional metadata-classifier is shown:



Figure 5.8: Screenshot: an analyzed comment (2)

For each analyzed comment, the report also contains some features that were calculated from the text. The following screenshot only shows a part of the calculated and displayed features:

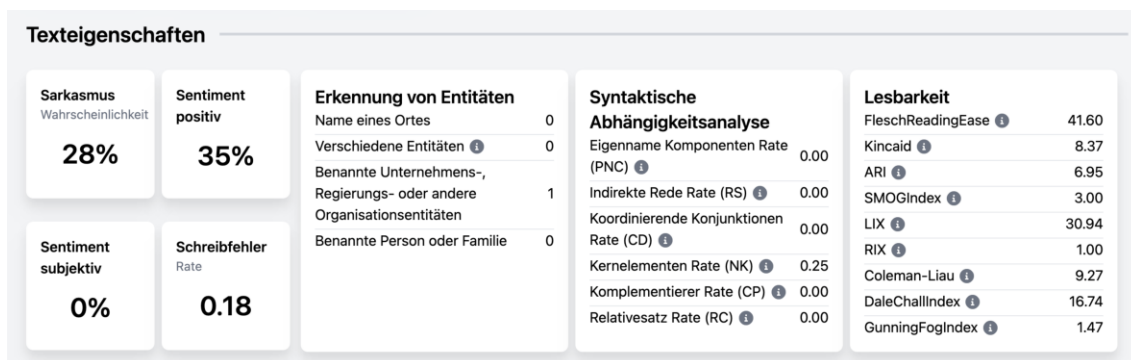


Figure 5.9: Screenshot: text-features of a comment

To explore existing comments, the news sources from the database are shown:

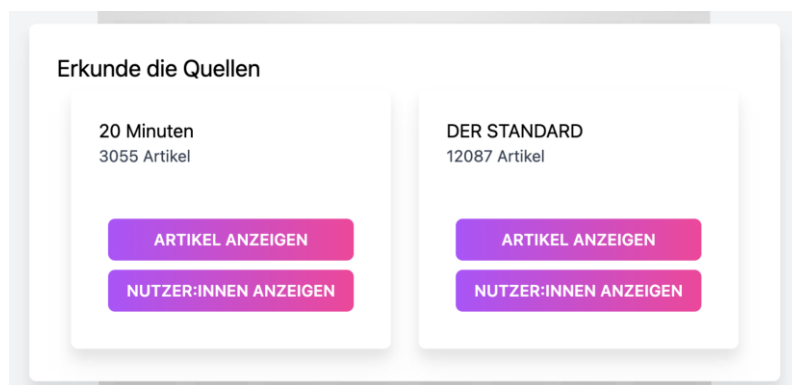


Figure 5.10: Screenshot: explore newssources

For both news sources, comments and users can be displayed:

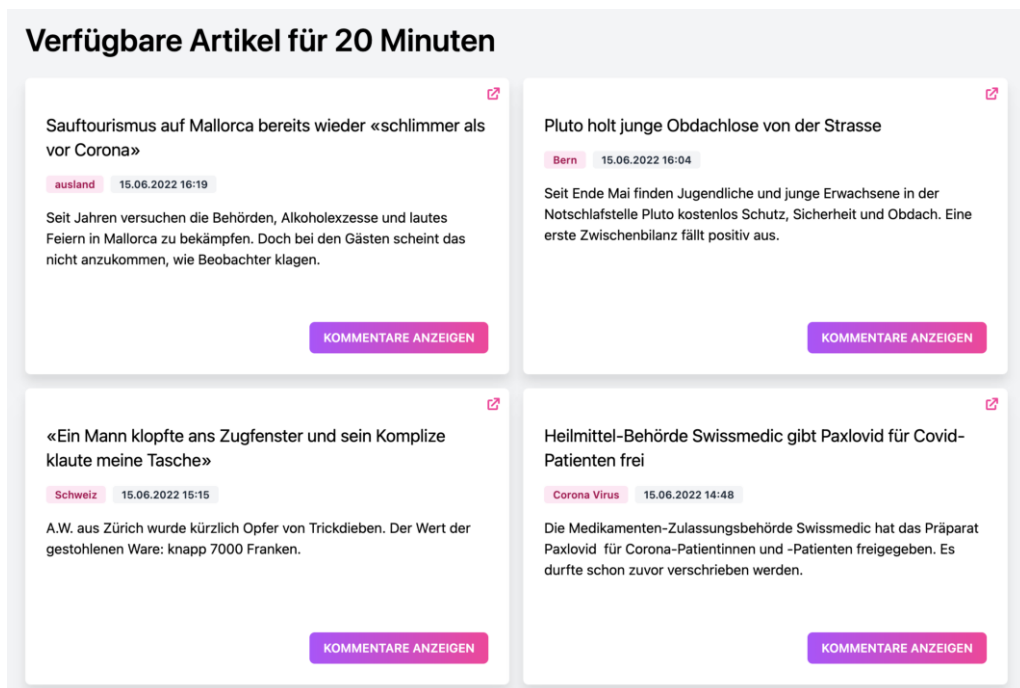


Figure 5.11: Screenshot: articles

Clicking on an article reveals the comments that were written for this article:

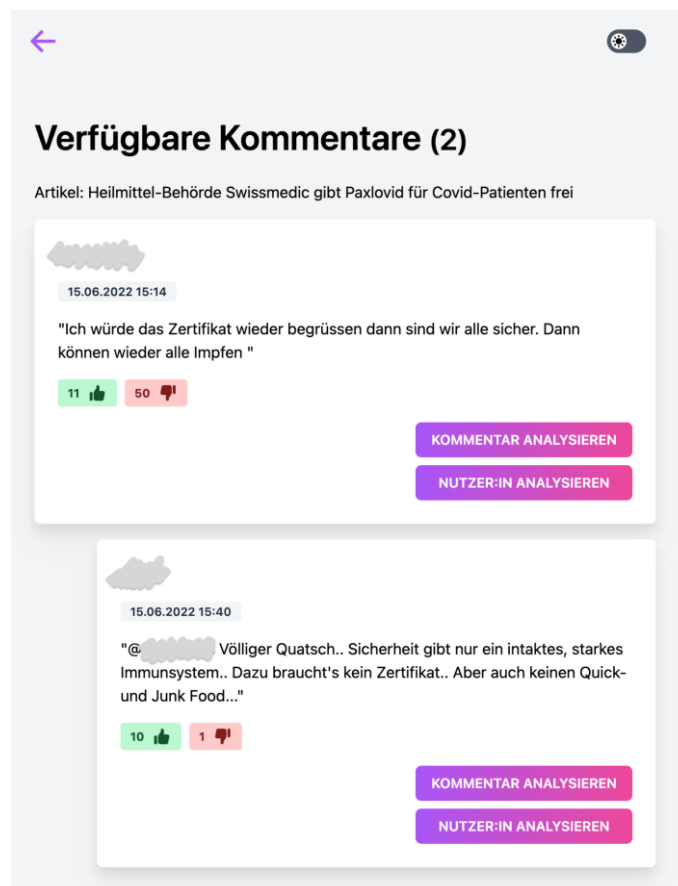


Figure 5.12: Screenshot: comments

From there, each comment can be analyzed using a single click. Along with the results from the classifiers, additional statistics are displayed for existing comments. Those statistics try to help the user of the application understand the context a comment was written in:

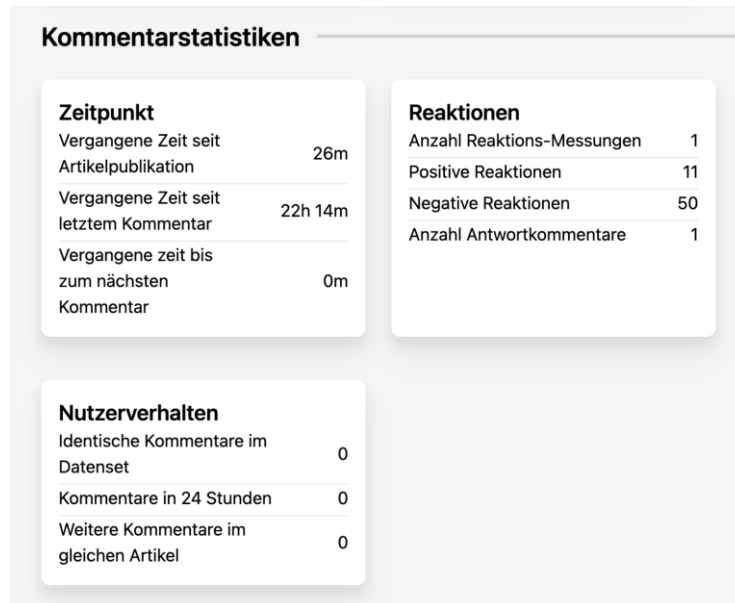


Figure 5.13: Screenshot: comment features

Statistics can not only be shown for individual comments, but also for existing comment authors. The following statistics, for example, are generated for a very active user. The screenshot only shows a part of the statistics displayed on the web application.

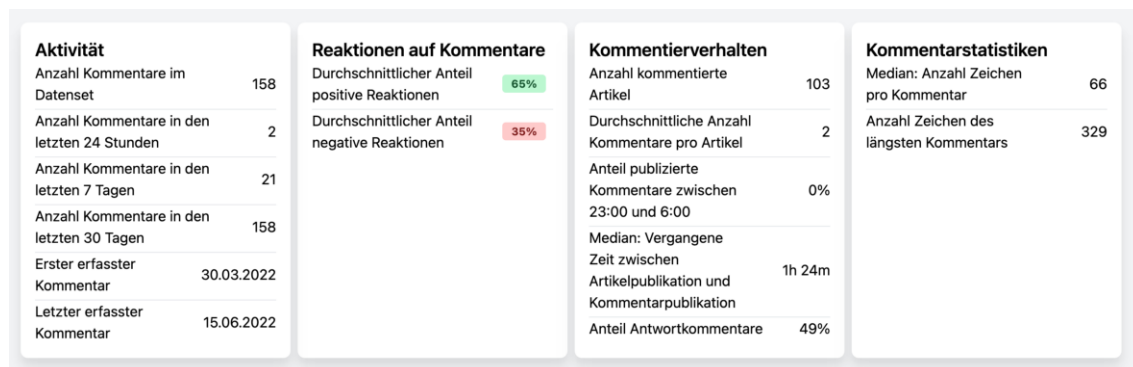


Figure 5.14: Screenshot: user features

Again, the statistics are intended to help users of the web application to better understand the behavior of a comment author. For example, the web application allows displaying a list of other comments from the same comment author. This functionality is usually not provided on the webpages of most German-language / Swiss online newspapers. Such information can help to decide whether a comment author is legit or not.

CHAPTER 6

Conclusion

This chapter wraps up the project. The work is summarized and the main hypothesis is validated. In addition, an outlook on possible future work is provided.

6.1 Summary

All contacted newspapers have confirmed that trolling is a problem they encounter. Different newspapers work very differently in detecting and removing inappropriate comments. While some news websites such as `Watson.ch` rely on a manual review process, other news websites such as `NZZ.ch` work with machine learning tools to support human content moderators.

The first step was to clarify the definition of the word "troll". In the literature, "trolling" is used in different contexts. For this project, all undesirable behavior in comments was referred to as "trolling". The comment rules of the various newspapers were used as a guide to define troll-behavior. Three trolling categories were defined: hate trolls, off-topic trolls and state-linked propaganda trolls.

This project focused on German-language user comments in online newspapers. The project focused on the news platform 20 Minuten. The goal was to write classifiers that can detect inappropriate comments. Existing scientific approaches were combined and applied to concrete examples. Among the features analyzed were the use of abusive language, sentiment, sarcasm, non-article related language and suspicious metadata.

The analysis revealed that the available metadata on the comments alone is not sufficient to reliably detect troll comments. Since rejected comments are not publicly visible on 20 Minuten, important features like the number of likes and dislikes or replies to a comment are missing. With an analysis of the text, hate trolls can be detected reliably. Often, offensive words and insults are the reason that a comment is rejected. The cosine similarity of a comment to the article text and to the other comments is also an effective feature to detect off-topic comments.

There was no training data on state-linked propaganda trolls from online newspapers in German. However, there was a dataset from Twitter with numerous German propaganda tweets written by paid trolls on behalf of various states. A classifier trained on this data uses sentiment analysis and sarcasm detection. The classifier performs very well on the Twitter data but it is unknown if the results can be transferred to newspaper comments as well.

The various algorithms were made accessible via a web application. This allows lay users to try out the machine learning models with their own or existing comments. The tool can display the latest articles and comments from 20 Minuten. Besides the results of the classifiers, statistics are displayed for each comment and comment author. For example, it is possible to find out how active a specific comment author has been in the last 24 hours, or how many comments an author writes on average per article. Such metrics help the user to better understand comments in an overall context.

6.2 Validation of the Main Hypothesis

One question that remains is whether the main hypothesis can be accepted. As stated in the introduction, the hypothesis is formulated as follows:

The proposed classifiers are helpful in supporting the manual moderation of inappropriate content in newspaper comments.

This hypothesis is accepted. Although the performance of the developed classifiers are not enough to completely replace the manual moderation, the classifiers can support the human moderators in their work. For 20 Minuten comments, the results have shown that about 20% of the troll comments can be filtered out with almost no false positive. Setting a threshold for the classifiers is a tightrope walk: If the threshold is set too low, too many unproblematic comments are automatically removed. Today, most of the contacted newspapers already use a combination of automated and manual moderation. Blick.ch works with the approach suggested by this thesis: they reject about 10% of the comments automatically using an automated algorithm, while the other 90% of the comments are reviewed manually.

6.3 Outlook and Further Research

Further research could focus more on unsupervised machine learning algorithms. From 20 Minuten, more than 2 million comments from the year 2021 are available as training data. This is significantly more data than similar projects have worked with. Various other analyses could be carried out on this dataset.

This project has mainly focused on the detection of individual troll comments (post-based approach). Presumably, the recall rate and the precision rate could be further improved by analyzing all comments of a comment author (user-based approach). Some future ideas for a user-based approach are already given in Chapter D.

The hate classifiers are currently strongly focused on individual words. This can sometimes lead to irritating results: Comments containing words like "Ausländer" (Foreigners) or "Schwarze" (Black people) are sometimes wrongly detected as hate speech. Therefore, it could be investigated on how to make the classifiers more aware about the whole context of a comment instead.

The classifiers trained for this project use different training data. It could be investigated how those classifiers perform for different data in different domains. An ensemble classifier could be created in order to combine those different approaches.

Other approaches are conceivable for the user interface. For example, a browser integration would be interesting. A browser extension could display analyzes for comments automatically on `20minuten.ch`. The current user interface could be further improved, for example by implementing functionality like paging, filtering and sorting. Charts and visualization could be used to describe the behavior of a comment author.

Bibliography

- [1] M. Tomaiuolo, G. Lombardo, M. Mordonini, S. Cagnoni, and Poggi, “A Survey on Troll Detection,” vol. 12, p. 31.
- [2] T. K. Kim, “T test as a parametric statistic,” vol. 68, no. 6, p. 540.
- [3] N. Cressie and H. Whitford, “How to use the two sample t-test,” vol. 28, no. 2, pp. 131–148.
- [4] Student’s t-test. [Online]. Available: https://en.wikipedia.org/wiki/Student%27s_t-test
- [5] Welch’s t-test. [Online]. Available: https://en.wikipedia.org/wiki/Welch%27s_t-test
- [6] Kolmogorov–smirnov test. [Online]. Available: https://en.wikipedia.org/wiki/Kolmogorov%27s_test#cite_note-AK-4
- [7] Scipy ks 2 samples. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ks_2samp.html#r2a7d47e1a68b-1
- [8] J. L. Hodges, “The significance probability of the smirnov two-sample test,” *Arkiv för Matematik*, vol. 3, no. 5, pp. 469–486, 1958.
- [9] Anderson-darling and shapiro-wilk tests. [Online]. Available: <https://www.itl.nist.gov/div898/handbook/prc/section2/prc213.htm>
- [10] Scipy shapiro test. [Online]. Available: <https://scipy.github.io/devdocs/reference/generated/scipy.stats.shapiro.html?highlight=shapiro>
- [11] H. Levene *et al.*, “Contributions to probability and statistics,” *Essays in honor of Harold Hotelling*, pp. 278–292, 1960.
- [12] Levene test for equality of variances. [Online]. Available: <https://www.itl.nist.gov/div898/handbook/eda/section3/eda35a.htm>
- [13] Scipy levene test. [Online]. Available: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.levene.html#r7cdc7a5c4c19-2>
- [14] R. Gorwa and D. Guilbeault, “Unpacking the Social Media Bot: A Typology to Guide Research and Policy,” vol. 12, no. 2, pp. 225–248. [Online]. Available: <https://onlinelibrary.wiley.com/doi/10.1002/poi3.184>
- [15] Digital Influence Engineering. [Online]. Available: <https://www.cardiff.ac.uk/crime-security-research-institute/publications/research-briefings/digital-influence-engineering>
- [16] A. Birkbak, “Into the Wild Online: Learning from Internet Trolls.” [Online]. Available: <https://www.firstmonday.org/ojs/index.php/fm/article/view/8297>

- [17] <https://www.tagesschau.de/faktenfinder/inland/organisierte-trolle-101.html>. Rechte troll-fabrik: Infokrieg mit allen mitteln. tagesschau.de. [Online]. Available: <https://www.tagesschau.de/faktenfinder/inland/organisierte-trolle-101.html>
- [18] European Parliament. Directorate General for Parliamentary Research Services., *Polarisation and the Use of Technology in Political Campaigns and Communication*. Publications Office. [Online]. Available: <https://data.europa.eu/doi/10.2861/167110>
- [19] Henry Farrell, “The Chinese Government Fakes Nearly 450 Million Social Media Comments a Year. This Is Why.” [Online]. Available: <https://www.washingtonpost.com/news/monkey-cage/wp/2016/05/19/the-chinese-government-fakes-nearly-450-million-social-media-comments-a-year-this-is-why/>
- [20] Mike Firn, “North Korea Builds Online Troll Army of 3,000.” [Online]. Available: <https://www.telegraph.co.uk/news/worldnews/asia/northkorea/10239283/North-Korea-builds-online-troll-army-of-3000.html>
- [21] Russlands Informationskrieg - So funktioniert eine Troll-Fabrik. [Online]. Available: <https://www.srf.ch/news/international/so-funktioniert-eine-troll-fabrik>
- [22] L. Bazley. Combating Jihadists and Free Speech: How the U.S. Military Is Using Fake Online Profiles to Spread Propaganda. [Online]. Available: <https://www.dailymail.co.uk/news/article-1367535/U-S-military-using-fake-online-profiles-spread-propaganda.html>
- [23] A. Reynolds, *Web of Deceit: Misinformation and Manipulation in the Age of Social Media*. NATO Strategic Communications Centre of Excellence. [Online]. Available: <https://www.ceeol.com/search/book-detail?id=775113>
- [24] Latvian Institute of International Affairs, Riga Stradins University, “Internet Trolling as a Hybrid Warfare Tool: The Case of Latvia.” [Online]. Available: <https://stratcomcoe.org/publications/internet-trolling-as-a-hybrid-warfare-tool-the-case-of-latvia/160>
- [25] Clemson University Media Forensics Hub. Profile 2 Analysis: Harmony. [Online]. Available: <https://spotthetroll.org/resources/2>
- [26] NZZ Redaktion. Die Kommentarfunktion bei der NZZ ist zurück – die Neuerungen. [Online]. Available: <https://www.nzz.ch/social-media/die-neue-kommentarfunktion-der-nzz-das-sind-die-neuerungen-ld.1523759>
- [27] E. Noelle-Neumann, *The Spiral of Silence: Public Opinion, Our Social Skin*. University of Chicago Press.
- [28] The Bandwagon Effect: Why People Tend to Follow the Crowd – Effectiviology. [Online]. Available: <https://effectiviology.com/bandwagon/>
- [29] S. Dollberg, “The Metadata Troll Detector,” p. 18.
- [30] J. Cheng, C. Danescu-Niculescu-Mizil, and J. Leskovec, “Antisocial Behavior in Online Discussion Communities.” [Online]. Available: <http://arxiv.org/abs/1504.00680>
- [31] L. Goode. Google just made its troll-detecting software available to developers. The Verge. [Online]. Available: <https://www.theverge.com/2017/2/23/14713496/google-jigsaw-perspective-software-ai-machine-learning-developers>
- [32] B. Schafer. Hamilton 2.0 Methodology & FAQs. Alliance For Securing Democracy. [Online]. Available: <https://securingdemocracy.gmfus.org/hamilton-2-0-methodology-faqs/>
- [33] T. De Smedt, P. Voué, S. Jaki, E. Duffy, and L. El-Khoury. A feast for trolls – Engagement analysis of counternarratives against online toxicity. [Online]. Available: <http://arxiv.org/abs/2111.07188>
- [34] J. Skowronski. Trolls and bots are disrupting social media — Here’s how AI can stop them (Part 1). [Online]. Available: <https://towardsdatascience.com/trolls-and-bots-are-disrupting-social-media-heres-how-ai-can-stop-them-d9b969336a06>
- [35] “Liste von schweizer zeitungen.” [Online]. Available: https://de.wikipedia.org/w/index.php?title=Liste_von_Schweizer_Zeitungen&oldid=220964786

- [36] O. Roeder. Why We’re Sharing 3 Million Russian Troll Tweets. FiveThirtyEight. [Online]. Available: <https://fivethirtyeight.com/features/why-were-sharing-3-million-russian-troll-tweets/>
- [37] D. Schabus, M. Skowron, and M. Trapp, “One Million Posts: A Data Set of German Online Discussions,” in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pp. 1241–1244. [Online]. Available: <https://dl.acm.org/doi/10.1145/3077136.3080711>
- [38] RP-Mod & RP-Crowd: Moderator- and Crowd-Annotated German News Comment Datasets | Zenodo. [Online]. Available: <https://zenodo.org/record/5291339#.YjiEu5rMJB1>
- [39] T. Mihaylov and P. Nakov, “Hunting for Troll Comments in News Community Forums.” [Online]. Available: <http://arxiv.org/abs/1911.08113>
- [40] G. Sarna and M. P. S. Bhatia, “Content based approach to find the credibility of user in social networks: An application of cyberbullying,” vol. 8, no. 2, pp. 677–689. [Online]. Available: <https://doi.org/10.1007/s13042-015-0463-1>
- [41] D. Martin. (08:00:00-07:00) Are you sure that’s a probability? Stacked Turtles. [Online]. Available: <https://kiwidamien.github.io>
- [42] E. Papegnies, V. Labatut, R. Dufour, and G. Linares, “Detection of abusive messages in an on-line community,” in *14ème Conférence En Recherche d’Information et Applications (CORIA)*, ser. Conférence En Recherche d’Information et Applications, pp. 153–168. [Online]. Available: <https://hal-univ-avignon.archives-ouvertes.fr/hal-01505017>
- [43] Naganna Chetty and Sreejith Alathur, “Hate speech review in the context of online social networks.”
- [44] Jaime Rodríguez. Hate speech. [Online]. Available: <https://rm.coe.int/168071e53e>
- [45] D. Assenmacher, M. Niemann, K. Müller, M. V. Seiler, D. M. Riehle, and H. Trautmann, “RP-Mod & RP-Crowd: Moderator- and Crowd-Annotated German News Comment Datasets.” [Online]. Available: <https://zenodo.org/record/5242915>
- [46] S. Sadiq, A. Mehmood, S. Ullah, M. Ahmad, G. S. Choi, and B.-W. On, “Aggression detection through deep neural model on twitter,” *Future Generation Computer Systems*, vol. 114, pp. 120–129, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X19330717>
- [47] Cross English & German RoBERTa for Sentence Embeddings. Hugging Face. [Online]. Available: <https://huggingface.co/T-Systems-onsite/cross-en-de-roberta-sentence-transformer>
- [48] German Hatespeech Dataset - HuggingFace. [Online]. Available: <https://huggingface.co/deepset/bert-base-german-cased-hatespeech-GermEval18Coarse>
- [49] Deepset/Gbert-Base · Hugging Face. [Online]. Available: <https://huggingface.co/deepset/gbert-base>
- [50] M. Alizadeh, J. N. Shapiro, C. Buntain, and Joshua A. Tucker, “Content-based features predict social media influence operations,” vol. 6, no. 30, p. eabb5824. [Online]. Available: <https://www.science.org/doi/abs/10.1126/sciadv.abb5824>
- [51] Nicola Davis and Alex Hern, “AI system detects posts by foreign trolls on Facebook and Twitter.” [Online]. Available: <https://www.theguardian.com/technology/2020/jul/22/ai-system-detects-posts-by-foreign-trolls-on-facebook-and-twitter>
- [52] Facebook takedowns reveal sophistication of Russian trolls. [Online]. Available: <https://www.pbs.org/newshour/politics/facebook-takedowns-reveal-sophistication-of-russian-trolls>
- [53] B. Ghanem, D. Buscaldi, and P. Rosso, “TexTrolls: Identifying Russian trolls on Twitter from a textual perspective.”
- [54] Dependency parsing in natural language processing with examples. [Online]. Available: <https://www.analyticsvidhya.com/blog/2021/12/dependency-parsing-in-natural-language-processing-with-examples/>

- [55] S. Albert, J. Anderssen, R. Bader, S. Becker, and T. Bracht, “Tiger annotationsschema,” p. 148.
- [56] Indirect speech. [Online]. Available: https://en.wikipedia.org/wiki/Indirect_speech
- [57] R. Huddleston, G. K. Pullum, and B. Reynolds, *A student’s introduction to English grammar*. Cambridge University Press, 2021.
- [58] “LIX.” [Online]. Available: [https://en.wikipedia.org/wiki/Lix_\(readability_test\)](https://en.wikipedia.org/wiki/Lix_(readability_test))
- [59] “Gunning fog index.” [Online]. Available: https://en.wikipedia.org/wiki/Gunning_fog_index
- [60] Flesch–Kincaid readability tests. [Online]. Available: https://en.wikipedia.org/wiki/Flesch\T1\textendashKincaid_readability_tests
- [61] “Automated readability index.” [Online]. Available: https://en.wikipedia.org/wiki/Automated_readability_index
- [62] “SMOG.” [Online]. Available: <https://en.wikipedia.org/wiki/SMOG>
- [63] “RIX.” [Online]. Available: <https://readable.com/blog/the-lix-and-rix-readability-formulas/>
- [64] “Coleman–Liau index.” [Online]. Available: https://en.wikipedia.org/wiki/Coleman\T1\textendashLiau_index
- [65] “Dale–Chall readability.” [Online]. Available: https://en.wikipedia.org/wiki/Dale\T1\textendashChall_readability_formula
- [66] R. T. T. H. Gareth James, Daniela Witten, *An Introduction to Statistical Learning*. Springer New York Heidelberg Dordrecht London.
- [67] E. Riloff, A. Qadir, P. Surve, L. De Silva, N. Gilbert, and R. Huang, “Sarcasm as contrast between a positive sentiment and negative situation,” in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 704–714.
- [68] F. Barbieri, H. Saggion, and F. Ronzano, “Modelling sarcasm in twitter, a novel approach,” in *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pp. 50–58.
- [69] Sarcasm classification (using fasttext). [Online]. Available: <https://towardsdatascience.com/sarcasm-classification-using-fasttext-788ffbacb77b>
- [70] A. G. H. S. Sowmya Vajjala, Bodhisattwa Majumder, *Practical Natural Language Processing*. O’Reilly Media, Inc.
- [71] H. Weller and J. Woo, “Identifying russian trolls on reddit with deep learning and bert word embeddings,” *Comput. Sci.*, pp. 1–11, 2019.
- [72] A. Atanasov, G. D. F. Morales, and P. Nakov, “Predicting the role of political trolls in social media,” 2019. [Online]. Available: <https://arxiv.org/abs/1910.02001>
- [73] J. Alshehri, M. Stanojevic, E. C. Dragut, and Z. Obradovic, “Stay on topic, please: Aligning user comments to the content of a news article,” vol. abs/2103.06130. [Online]. Available: <https://arxiv.org/abs/2103.06130%7D>
- [74] J. B. Singer, “Separate spaces: Discourse about the 2007 Scottish elections on a national newspaper web site,” vol. 14, no. 4, pp. 477–496.
- [75] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence embeddings using siamese BERT-Networks.” [Online]. Available: <https://arxiv.org/abs/1908.10084>
- [76] Adrien Sieg. Text Similarities : Estimate the degree of similarity between two texts. Medium. [Online]. Available: <https://medium.com/@adriensieg/text-similarities-da019229c894>
- [77] S. Tata and J. M. Patel, “Estimating the selectivity of $\text{j}_i\text{Tf-Idf}_i/i_i$ based cosine similarity predicates,” vol. 36, no. 2, pp. 7–12. [Online]. Available: <https://doi.org/10.1145/1328854.1328855>

- [78] Sentence transformers: Meanings in disguise. [Online]. Available: <https://www.pinecone.io/learn/sentence-embeddings/>
- [79] Sentence transformers library. [Online]. Available: https://www.sbert.net/docs/package_reference/SentenceTransformer.html
- [80] Tfidf and sparse matrices. [Online]. Available: http://www.xavierdupre.fr/app/onnxcustom/helpsphinx/auto_examples/plot_usparse_xgboost.html
- [81] S. Niwattanakul, J. Singthongchai, E. Naenudorn, and S. Wanapu, "Using of Jaccard coefficient for keywords similarity," in *Proceedings of the International Multiconference of Engineers and Computer Scientists*, vol. 1, no. 6, pp. 380–384.
- [82] T. Mihaylov, T. Mihaylova, P. Nakov, L. Mârquez, G. D. Georgiev, and I. K. Koychev, "The dark side of news community forums: Opinion manipulation trolls," vol. 28, no. 5, pp. 1292–1312. [Online]. Available: <https://doi.org/10.1108/IntR-03-2017-0118>
- [83] T. Mihaylov, G. Georgiev, and P. Nakov, "Finding Opinion Manipulation Trolls in News Community Forums," in *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pp. 310–314. [Online]. Available: <https://aclanthology.org/K15-1032>
- [84] M. Jones, "Internet Article Comment Classifier," p. 7.
- [85] K. Machová, M. Porezaný, and M. Hreškova, "Algorithms of Machine Learning in Recognition of Trolls in Online Space," in *2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMII)*, pp. 000 349–000 354.
- [86] S. Suryawanshi, B. R. Chakravarthi, M. Arcan, S. Little, and P. Buitelaar, "TrollsWithOpinion: A Dataset for Predicting Domain-specific Opinion Manipulation in Troll Memes." [Online]. Available: <http://arxiv.org/abs/2109.03571>
- [87] Troll. [Online]. Available: <https://en.wikipedia.org/wiki/Troll>
- [88] Sklearn SVM. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>
- [89] M. Fazil and M. Abulaish, "A Hybrid Approach for Detecting Automated Spammers in Twitter," vol. 13, no. 11, pp. 2707–2719.
- [90] Catherine Cote. 7 Data Collection Methods in Business Analytics. Harvard Business School. [Online]. Available: <https://online.hbs.edu/blog/post/data-collection-methods>
- [91] S. d. S. Sirisuriya, "A Comparative Study on Web Scraping," p. 6.
- [92] What is Exploratory Data Analysis? [Online]. Available: <https://www.ibm.com/cloud/learn/exploratory-data-analysis>
- [93] S. Pandey and S. K. Dubey, "A Comparative Analysis of Partitioning Based Clustering Algorithms and Applications," vol. 2, no. 12, p. 13.
- [94] C. Manning, P. Raghavan, and H. Schuetze, "Introduction to Information Retrieval," p. 452.
- [95] R. Grosse, A.-m. Farahmand, and J. Carrasquilla, "Expectation-Maximization," p. 33.
- [96] G. Gan, C. Ma, and J. Wu, *Data Clustering Theory, Algorithms, and Applications*, ser. ASA-SIAM Series on Statistics and Applied Probability. SIAM, Society for Industrial and Applied Mathematics ; American Statistical Association, no. 20.
- [97] The DBSCAN Algorithm: Background and Theory. [Online]. Available: <https://jaredgerschler.blog/category/applied-statistics/>
- [98] G. Gan, C. Ma, and J. Wu, *Data Clustering Theory, Algorithms, and Applications*.
- [99] A. Hinneburg and H.-H. Gabriel, "DENCLUE 2.0: Fast Clustering Based on Kernel Density Estimation," in *Advances in Intelligent Data Analysis VII*, ser. Lecture Notes in Computer Science, M. R. Berthold, J. Shawe-Taylor, and N. Lavrač, Eds. Springer Berlin Heidelberg, vol. 4723, pp. 70–80. [Online]. Available: http://link.springer.com/10.1007/978-3-540-74825-0_7

- [100] F. Kovács, C. Legány, and A. Babos, “Cluster Validity Measurement Techniques,” p. 11.
- [101] A. Bhardwaj. Silhouette Coefficient : Validating clustering techniques. Medium. [Online]. Available: <https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c>
- [102] E. Briscoe and J. Feldman, “Conceptual complexity and the Bias/Variance tradeoff,” vol. 118, no. 1, pp. 2–16. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0010027710002295>
- [103] Jason Brownlee. A gentle introduction to K-fold cross-validation. [Online]. Available: <https://machinelearningmastery.com/k-fold-cross-validation/>
- [104] Area under the ROC curve (AUC). [Online]. Available: <https://evispot.ai/area-under-the-roc-curve-auc/>
- [105] I. T. Jolliffe and J. Cadima, “Principal component analysis: a review and recent developments,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, 2016.
- [106] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne.” *Journal of machine learning research*, vol. 9, no. 11, 2008.
- [107] L. McInnes, J. Healy, and J. Melville, “Umap: Uniform manifold approximation and projection for dimension reduction,” *arXiv preprint arXiv:1802.03426*, 2018.
- [108] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [109] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *arXiv preprint arXiv:1609.08144*, 2016.

Glossary

AUC *Area Under (the ROC) Curve (AUC)* is a performance measure for classification schemes. 206

classification In statistics, classification is the problem of identifying to which category an observation belongs to. 198

confusion matrix A confusion matrix reports the number of true positives, false negatives, false positives, and true negatives. This allows a detailed analysis of the classifier. 205

degrees of freedom The degrees of freedom is a quantity that summarizes the flexibility of a model. 198

dendrogram A tree representing hierarchical relationship between objects. 193

hyperparameter A hyperparameter is a parameter that is set before the learning process begins. These parameters are tunable and can directly affect how well a model performs. 204

linear regression Linear regression in machine learning assumes a linear relationship between the input variables and the output variable. 198

null hypothesis In statistical tests, the null hypothesis suggests that no statistical relationship exists between two variables. 136

object relational mapper Converts a non-object-relational datastructure (often parts of database) into an object-relational structure. 25

overfitting More complex models can lead to a phenomenon known as overfitting, which essentially means they follow the errors, or noise, too closely. 198, 204

p-value The p-value tells how likely it is, that a result could have occurred under the *null hypothesis*. 198

regex search pattern specified by a sequence of characters. 191

regression In statistical modeling, regression analysis is a set of statistical processes to estimate the relationships between a dependent variable and one or more independent variables. 198

ROC A *Receiver Operating Characteristic (ROC)* curve is a graph showing the performance of a classification model at all classification thresholds. 206

scikit Machine Learning Toolkit for Python. 16

sigmoid function A mathematical function having a characteristic "S"-shape. 198, 200

supervised Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. 204

underfitting The model is not flexible enough to adequately represent the underlying structure of the data. 198

unix An operating system based on the UNIX specification. 191

unsupervised Unsupervised learning is a category of algorithms that learns patterns from unlabeled data. 204

XQuery XQuery is a functional programming language to query data. 191

Appendix

APPENDIX **A**

Unused Classifiers

This chapter contains classifiers that were developed during the project but were not included in the final product.

A.1 Rejection Classifier Unused Building Blocks

Overview

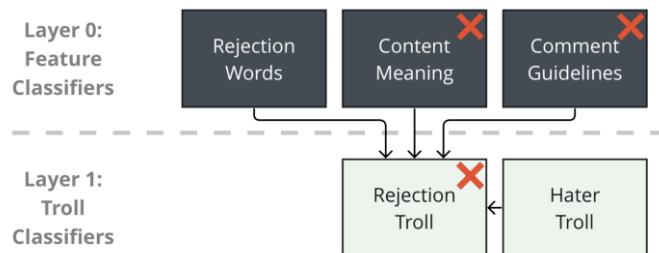


Figure A.1: Rejection classifier former overview

The classifier to predict whether a 20 Minuten comment gets rejected or not was initially supposed to be an ensemble classifier. In the end it turned out that the ensemble model performed worse than the Rejection Words classifier (Chapter 4.2) alone. For this reason, the building blocks that are marked with a cross have been discarded, including the ensemble model. Instead, the Rejection Words classifier has been included in the final big picture as a standalone classifier named Rejection Troll classifier. For completeness, the discarded classifiers are described in this section. The Hater Troll classifier is described in Chapter 4.3.

Idea

The building blocks are chosen to cover the 20 Minuten commentary guidelines¹. The guidelines are as follows:

1. **Defamation/insult** To ensure pleasant, factual and fair interaction, posts that are offensive in tone are not published. This includes the use of insulting expressions as well as personal attacks on other users.
2. **Racism/discrimination** It is not permitted to disseminate content that falls under the Swiss criminal law on racism and disparages persons based on their race, ethnicity, or religion or incites hatred. Discriminatory statements will not be published.
3. **Accusation/Defamation** Accusations against individuals or companies are not tolerated.
4. **Vulgarity** Comments that contain curse words or are vulgar are not published.
5. **Advertising** Self-promotion, advertising for commercial products or political advertising have no place in online comments. Calls for protests, associations or political actions are also inadmissible.
6. **Links** Posting links should be avoided. 20 Minuten reserves the right to edit or not publish posts with links.
7. **Rules Discussions** The comment section is not a place to negotiate comment policies. Unlock discussions (Freischaltdiskussionen) are generally not published.

¹The guidelines can be found here: <https://www.20min.ch/story/die-kommentarrichtlinien-von-20-minuten-119025471145>

The following table shows how the building blocks from Figure A.1 are intended to cover these guidelines.

Table A.1: 20 Minuten comments guidelines coverage

Classifier	Guidelines
Rejection Words	1, 2, 4
Content Meaning	1, 2, (3), 4, (5), (7)
Comment Guidelines	6
Hater Troll	1, 2, 4

The reason guidelines 3, 5 and 7 are written in parentheses, is because no literature could be found showing that these categories can be classified well using a *BERT* architecture.

In addition to the above guidelines, it is also mentioned that comments must be written in German². This is covered by the Comment Guidelines classifier.

²Comments need to be in German: <https://www.20min.ch/story/so-kannst-du-der-community-deine-meinung-sagen-786176008273>

A.1.1 Rejection Troll Content Meaning

Overview

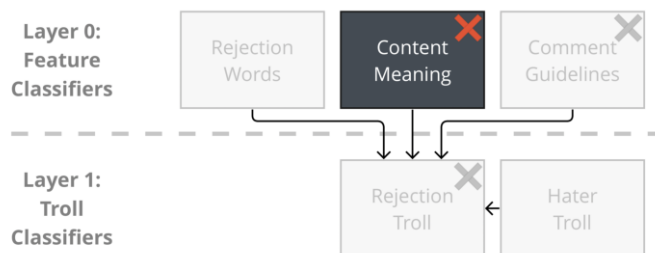


Figure A.2: Rejection content meaning classifier former overview

Feature Engineering

SBERT models can be fine-tuned to achieve better performance [79]. This was done using the SentenceTransformers Python library³.

Before fine-tuning the model, the dataset had to be cleaned because duplicate comments were found, one of which was accepted and the other rejected. After shuffling the dataset, the pre-trained "Cross English & German RoBERTa for Sentence Embeddings" model⁴ was fine-tuned. Figure A.3 illustrates the result of the trained model reduced from 768 dimensions to 2 using *PCA* with an explained variance of 12%.

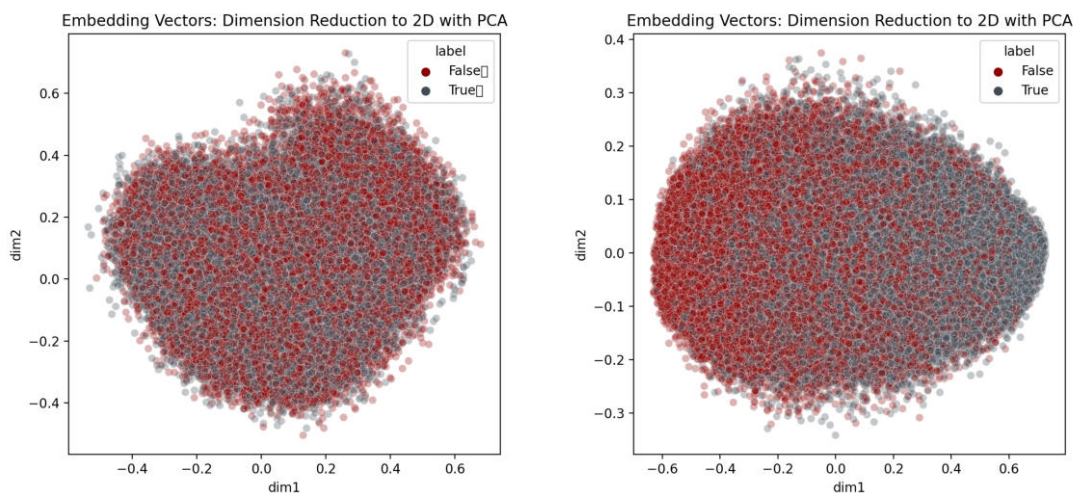


Figure A.3: Untrained SBERT (left) and fine tuned SBERT (right)

To counteract the Curse of Dimensionality, a dimension reduction to 100 dimensions using *PCA* with an explained variance of 80% has been applied. Figure A.4 shows the number of principal components versus the classifier score with a QDA scheme using 10-fold cross validation on a subset of 10'000 training samples.

³<https://www.sbert.net>

⁴<https://huggingface.co/T-Systems-onsite/cross-en-de-roberta-sentence-transformer>

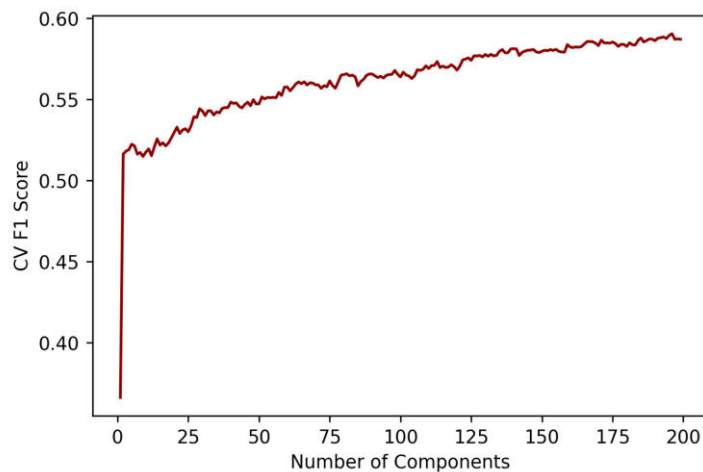


Figure A.4: Rejection troll content meaning optimal number of principal components

Modeling

A hyperparameter tuning was performed for multiple classification schemes to find the best performing model. This was done on a subset of 10'000 samples out of the 800'000 to save time. The summarized results are illustrated in Figure A.6 as *ROC*-curves.

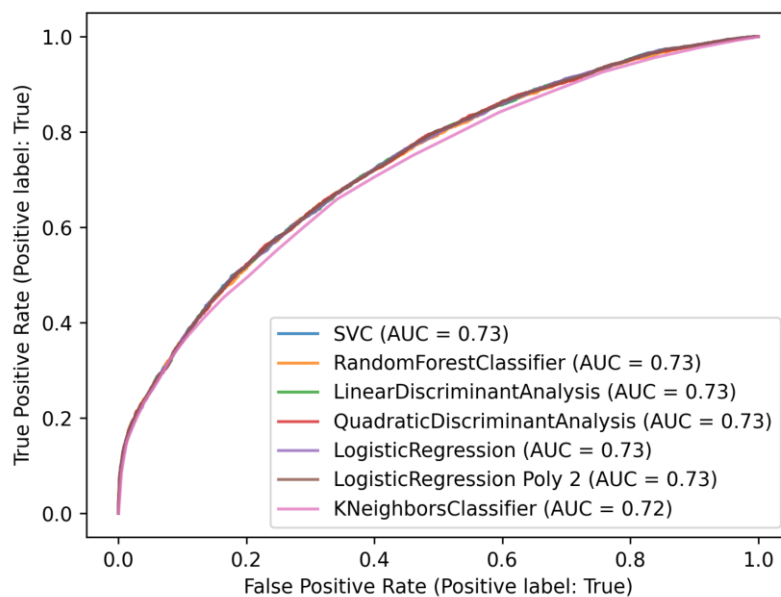


Figure A.5: ROC curve for different SBERT rejection classifiers

All classifiers roughly achieve about the same performance. Thus, Linear Discriminant Analysis, Logistic Regression and Random Forest were chosen to be trained on a bigger subset of 100'000 data. Random Forest now performed the best and was thus picked to train the final model.

Evaluation

The classifier achieves the following scores:

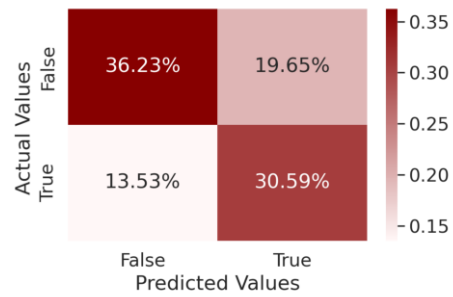


Figure A.6: Rejection troll content meaning classifier

Table A.2: Rejection troll content meaning classifier scores

Metric	Score
precision	61%
recall	69%
f1-score	65%
accuracy	67%

A.1.2 Rejection Troll Comment Guidelines

Overview

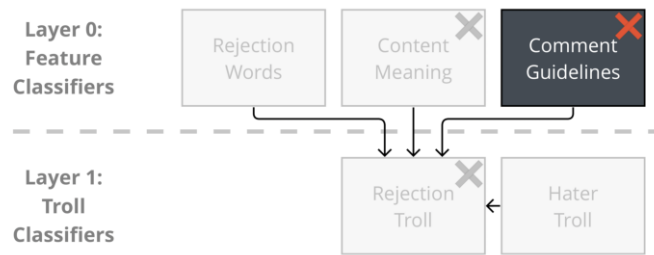


Figure A.7: Rejection classifier former overview

Idea

The purpose of this classifier is to cover guideline rules that are not covered by other classifiers.

Feature Engineering

This classifier only consists of the three features shown in the following table.

Table A.3: Features overview for the comment guidelines classifier

Category	Features	Number
Links	number of links	1
Language	is it German?	1
Length	number of words	1
Total		3

The number of words feature is not explicitly mentioned in the 20 Minuten guidelines, but it supports the idea from `Blick.ch` where very short comments are rejected because they do not contribute to a fruitful discussion (see Chapter 2.2.2). The check for German language was done using the Python package `Polyglot`⁵. Hypothesis tests are not performed, since this classifier was dropped anyway.

⁵<https://polyglot.readthedocs.io/en/latest/Detection.html>

Modeling

Different classification schemes have been tried with this dataset.

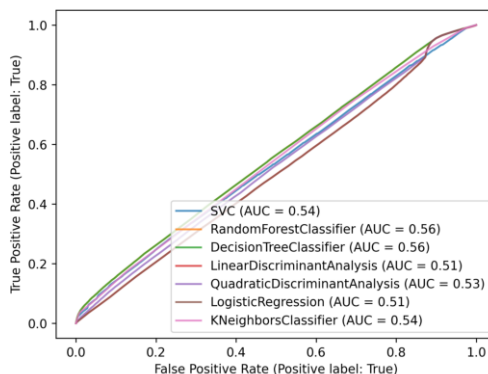


Figure A.8: ROC curves for different comment guideline classifiers

The decision tree classifier has been chosen as the classification scheme since it achieved the same score as the random forest but has a simpler implementation. It does not come as a surprise that a decision tree classifier performed best since the correlation between the features and the label is quite simple: For example, if a comment contains links, it is usually rejected.

Evaluation

The performance of this classifier is quite poor, but this is fine since the idea is to use it in the ensemble model. Figure A.9 shows the confusion matrix.

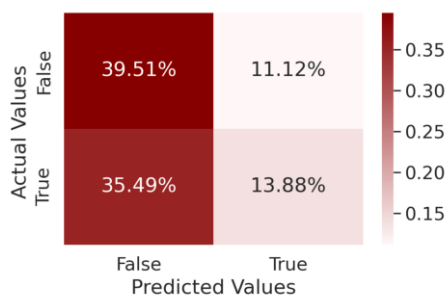


Figure A.9: Comment guidelines classifier confusion matrix

The following table shows the scores:

Table A.4: Rejection troll comment guidelines classifier scores

Metric	Score
precision	56%
recall	28%
f1-score	37%
accuracy	53%

A.1.3 Rejection Troll Ensemble

Overview

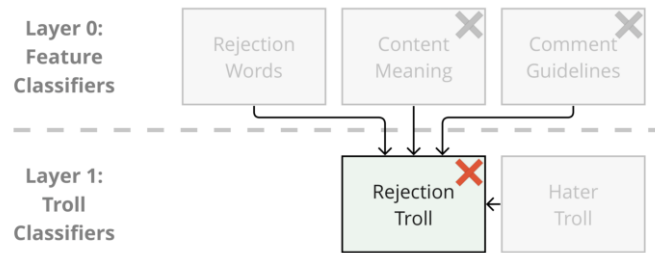


Figure A.10: Rejection classifier ensemble overview

Idea

To achieve better performance, the rejection classifiers described above and the hater classifier discussed in Chapter 4.3 were combined into one ensemble classifier.

Hypothesis

The hypothesis is as follows:

1. An ensemble method will improve the classifier relative to the individual components.

This hypothesis is discussed in the following sections.

Modeling

A stacking and a voting classifier were tried as an ensemble method. Their respective ROC-curves can be seen in the following figure:

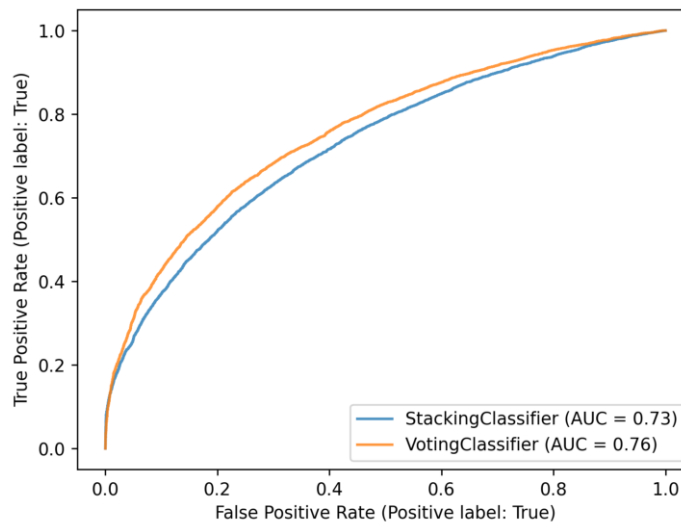


Figure A.11: ROC curves for rejection ensemble classifiers

The voting classifier was chosen to as the final model.

Evaluation

Testing the ensemble classifier on a dataset which was never used during training or optimization let to the following results:

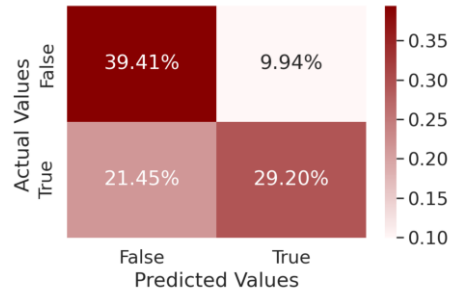


Figure A.12: Rejection troll ensemble confusion matrix

As it can be seen in Table A.5, the scores are not better than the Rejection Words classifier from Chapter 4.2. For this reason, the ensemble method was dropped in the final application.

Table A.5: Rejection troll content meaning classifier scores

Metric	Score
precision	75%
recall	58%
f1-score	65%
accuracy	69%

The following graph shows the precision vs. recall trade-off.

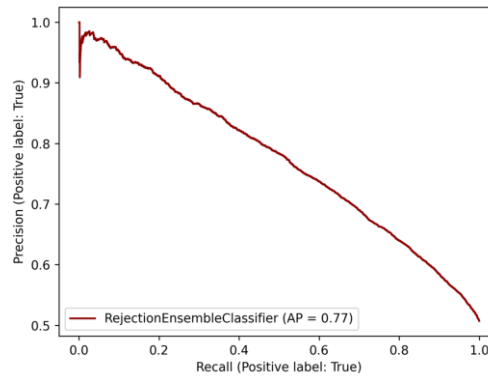


Figure A.13: Rejection troll ensemble precision vs recall

Further Research

Comments should also be related to the content of the article⁶. It could be investigated whether the off-topic classifier described in Chapter 4.5 would help to improve the result of the ensemble model.

⁶<https://www.20min.ch/story/so-kannst-du-der-community-deine-meinung-sagen-786176008273>

A.2 Sarcasm Classifier Version 1

Overview

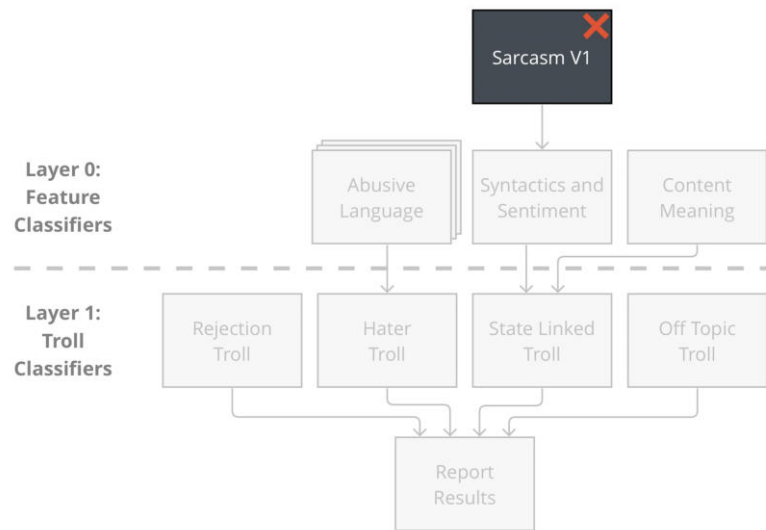


Figure A.14: Sarcasm classifier V1 overview

The sarcasm classifier in this section describes the first version of the classifier which was not included in the final application. A description of the second version can be found in Section 4.4.3.

Literature

Francesco Barbieri et al. showed that it is possible to distinguish ironic from non-ironic tweets [68]. They focused on features like part of speech counts, sentiment analysis, emojis, or the ambiguity of the text. Their irony classifier achieved a F1 score of 60%.

Hypothesis

The hypothesis for the sarcasm classifier version 1 is as follows:

1. Sarcasm in German can be detected with similar performance to Francesco Barbieri et al.'s classifier using English data.

This hypothesis is discussed in the following sections.

Data Acquisition

The same dataset is used as for the sarcasm classifier version 2. The description about the data acquisition can be found in Chapter 4.4.3.

The dataset was intentionally not preprocessed before feature engineering. This is because the different features require different preprocessing steps for optimal extraction. The individual preprocessing steps are thus left as part of the feature engineering.

Feature Engineering

An overview of all the features that were considered during training are listed in Table A.6. The idea was to include both the promising features from other papers and new brainstormed features. Ultimately, a feature selection algorithm should decide which features contribute the most to the final classifier.

Table A.6: Considered features for the sarcasm classifier

Category	Features	Amount
Ambiguity	Synset mean, Synset max	2
Part of Speech	Percentages of adjectives, adpositions, adverbials, auxiliary verbs, comparative conjunctions, determiners, interjections, nouns, numbers, particles, pronouns, proper nouns, punctuations, subordinate conjunctions, spaces, verbs	16
Frequency	Number of words, Number of exclamation marks, Number of question marks, Number of uppercase words, Number of emojis, Number of smilies :), Number of big smilies :D, Number of wink smilies ;), Number of sad smilies :(, Number of mentions, Number of links, Number of characters per word, Number of syllables per word, Number of long words, Number of complex words	15
Readability	FleschReadingEase, Kincaid, ARI, SMOGIndex, RIX, LIX, Coleman-Liau, DaleChallIndex, Gunning-FogIndex	9
Sentiment	Polarity, Subjectivity	2
Misspelling	Number of misspelled words	1
<i>SBERT</i>	50-Dimensional sentence embedding vectors	50
Total		95

Ambiguity An interesting aspect of sarcasm is ambiguity. According to Barbieri et al., sarcastic comments tend to use words with more meanings [68]. The underlying assumption is that if a word has many meanings, then the probability of sending more than one message is higher. Hence, there is a higher possibility of sarcasm. To extract this feature, the Odenet Python library⁷ has been used. This library contains a German lexicon for words with their respective synsets. A synset, also synonym ring, in this context is a group of semantically similar words. The more synsets a word belongs to, the more ambiguous it is. Two extracted features aim to capture this aspect: synset mean and synset max. The first is the mean of the number of synsets of each word in a text. The second one is the highest number of synsets a word in a text has. The problem with this feature is that if a comment contains a lot of misspelled words, it becomes infeasible because no words or even the wrong words can be found in the lexicon. As the dataset was translated, the spelling mistakes should have been mostly eliminated. Unfortunately, that inherently implies that the classifier will not be robust in production, where spelling mistakes still can occur. Another problem arises because of the use of conjugated verbs (like *have* or *had*) or different adjective forms (like *good* or *better*). Lemmatization resolves that problem. Overall, the following preprocessing steps were performed: Removal of punctuation and lemmatization. The hypothesis is that synset max will be relevant because according to Barbieri et al.’s classifier, this feature was meaningful [68].

⁷<https://github.com/hdaSprachtechnologie/odenet>

Part of Speech It has been shown that these features also contribute a lot to sarcasm classifiers [68]. Features like number of nouns and number of adjectives proved to contain useful information [68]. Other parts of speech in German were also counted. These include verbs, pronouns, adpositions (e.g. prepositions), auxiliary verbs, conjunctions, determiners, interjections, numerals, particles, proper nouns (name of a specific individual, place, or object). These features are extracted with the Spacy library⁸ which supports many languages, including German.

Frequency Different characteristics have been counted as done by Francesco Barbieri et al. [68]. The complete list can be found in Table A.6. Most characteristics could be counted with the help of regular expressions. The number of syllables per word, long and complex words were counted by the Readability library described in the next paragraph. Words are considered long if they exceed 6 characters [58], and they are considered complex if they consist of three or more syllables [59].

Readability Whether a text is sarcastic or not should intuitively not have a big impact on readability. Nevertheless, these features are included so that the feature selection algorithm can decide in the end. The readability indexes are described in more detail in Table 4.18. The effective scores in this project are calculated using the Python package Readability⁹ which also supports German.

Sentiment Barbieri et al. also included sentiment features in their sarcasm classifier, which encapsulated positive or negative emotions. They have shown that the polarity feature is meaningful [68]. Additionally, this classifier also uses a subjectivity feature, which tells how subjective or objective a comment is. The features are calculated with the textblob-de python library¹⁰. There was no need to perform any preprocessing steps, as this was already done by textblob-de.

Misspelling The number of misspelled words are counted and used as a feature. The pyspellchecker python library¹¹ is used to extract this feature. This library supports 6 languages, including German. No preprocessing steps were applied, since numbers and punctuation did not bother the library. Also, no lowercasing was applied because the library does not check for the correct capitalization but only wrong characters inside a word.

SBERT This feature was included to encapsulate the meaning of the content. The *SBERT* features are generated using the SentenceTransformer library¹² which is optimized for performance. The cross-en-de-roberta-sentence-transformer¹³ from huggingface has been used as the pre-trained model. *PCA* has been applied to keep the number of dimensions relatively small. Figure A.15 shows the number of principal components versus the *QDA* classifier score using 10-fold cross validation on a subset of 20'000 training samples.

⁸<https://spacy.io>

⁹<https://pypi.org/project/readability/>

¹⁰<https://github.com/markuskiller/textblob-de>

¹¹<https://github.com/barrust/pyspellchecker>

¹²<https://github.com/UKPLab/sentence-transformers>

¹³<https://huggingface.co/T-Systems-onsite/cross-en-de-roberta-sentence-transformer>

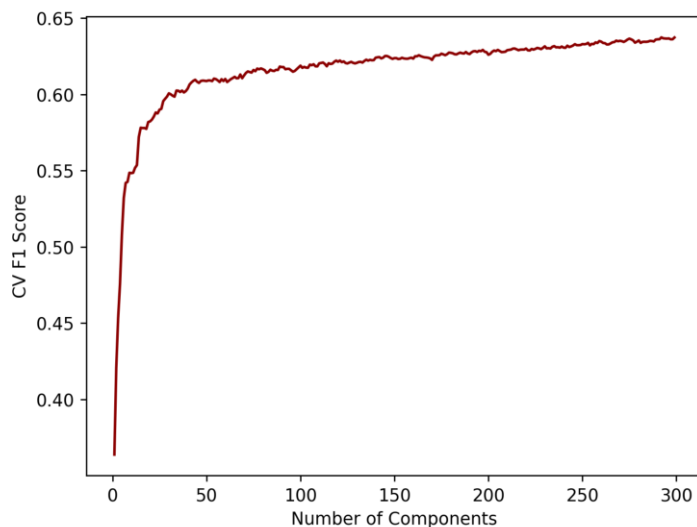


Figure A.15: Sarcasm classifier BERT optimal number of principal components

50 features have been chosen as the number of principal components. Hence, every 768-dimensional embedding vector output from the SentenceTransformer library is reduced to 50 dimensions while keeping about 8% of the original variance. No preprocessing steps were applied to keep as much context as possible for the transformer.

Feature Selection To get the most valuable features, an algorithm called Forward Stepwise Selection was used, which is described in more detail in Appendix G.3.3. This algorithm achieves a quadratic runtime rather than an exponential one by not trying to find the optimal overall solution. The procedure was as follows: A 100'000 x 47 design matrix was created, which results in $47 * (47 + 1) / 2 = 1128$ model fittings. A support vector machine with a polynomial kernel of degree 3 was then chosen to evaluate different feature combinations. Accuracy was selected as the optimization score because recall alone does not work that well as it was almost constantly at 100% while accuracy could not exceed 40%. The idea was to tune the recall score in the end by adjusting the decision threshold. After starting the algorithm, it was found that a Support Vector Machine scales at least quadratically with the number of samples [88]. The runtime performance can get even worse with the choice of hyperparameters like a large C. Thus the Quadratic Discriminant Analysis scheme has been selected to replace the Support Vector Machine. In the end, the algorithm returned this result:

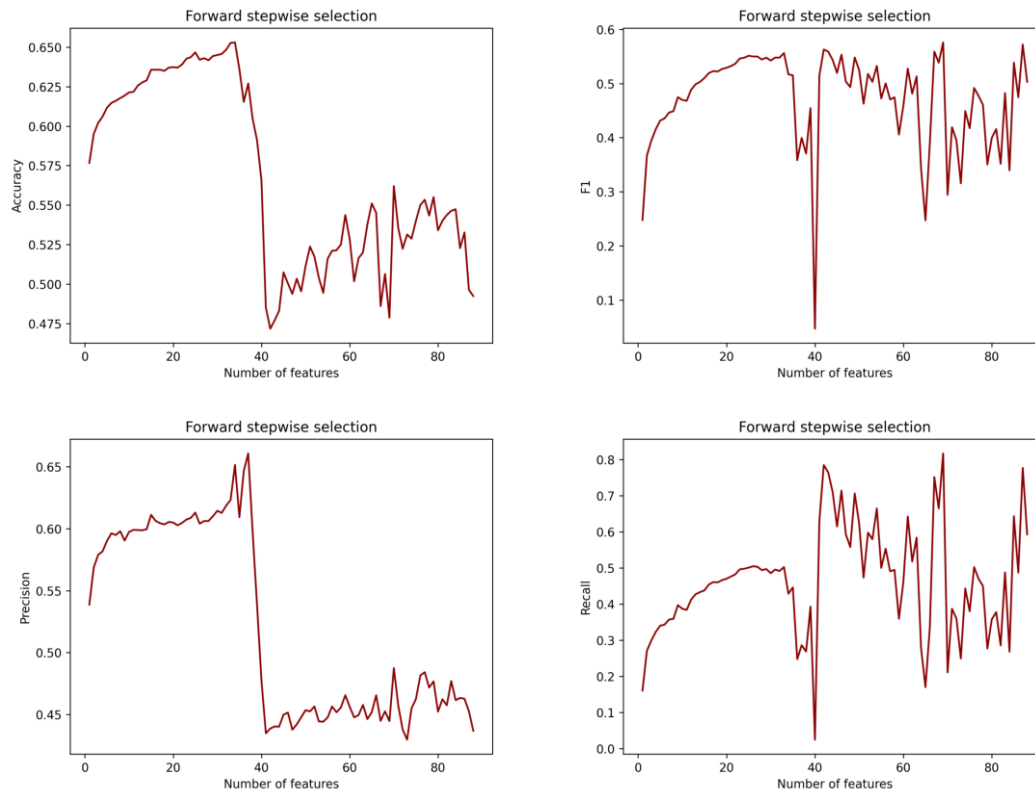


Figure A.16: Feature selection scorings

The first 25 features were finally selected. These features, sorted from highest to lowest relevance, are:

Table A.7: Sarcasm classifier: optimal features sorted

1. Features Sorted ↓	2. Features Sorted ↓	3. Features Sorted ↓
emb_dim_13	emb_dim_17	emb_dim_12
verb_in_%	emb_dim_35	emb_dim_2
pron_in_%	emb_dim_38	emb_dim_44
emb_dim_41	emb_dim_33	emb_dim_10
adj_in_%	emb_dim_25	emb_dim_11
emb_dim_19	emb_dim_32	emb_dim_36
emb_dim_37	emb_dim_39	emb_dim_43
emb_dim_7	conj_in_%	
synset_mean	emb_dim_42	

Modeling

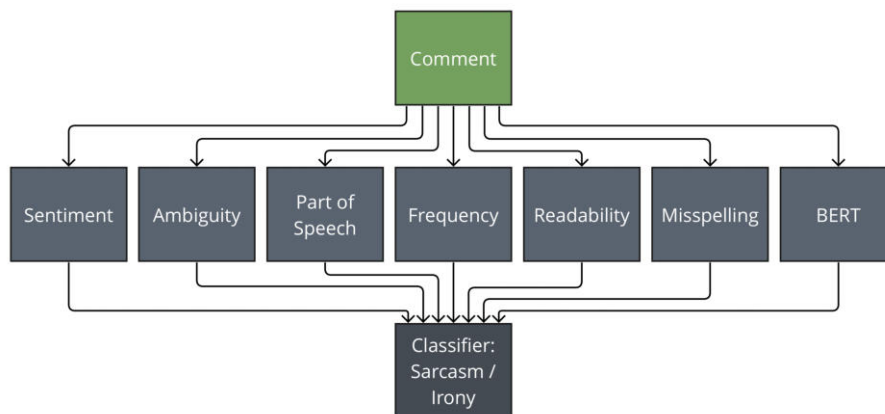


Figure A.17: Sarcasm classifier model

To better evaluate which classification scheme best fits this kind of data, the most relevant features have been plotted as a pair-plot. Since there is no clear decision boundary, it is difficult to tell by eyeballing which scheme is the most appropriate. Thus, an automated approach was needed to find the optimal classification scheme. For every classification scheme, different tuning parameters have been tested on a subset of 20'000 samples. The rest was put aside for an independent evaluation in the end. For many combinations, this process can be CPU-intensive. Hence, a randomized approach was used, which sklearn's `RandomizedGridSearchCV` already implements using cross validation. The best performing models are listed in Table A.8:

Table A.8: Sarcasm classifier: hyperparameter tuning results

Classification Scheme	Best Hyperparameters	Accuracy	Recall
<i>SVM</i>	kernel: <i>poly</i> , degree: 3, C: 100	0.54	0.5
Random Forest	n_estimators: 1600, min_samples_split: 10, min_samples_leaf: 1, max_features: <i>sqrt</i> , max_depth: 60	0.61	0.34
LDA	solver: <i>svd</i>	0.57	0.35
<i>QDA</i>	reg_param: 0.001	0.56	0.47
Logistic Regression	penalty: <i>l2</i> , C: 4799095	0.57	0.35
<i>KNN</i>	n_neighbors: 3	0.51	0.46

Figure A.18 shows the *ROC* curve of the classifiers with their best respective hyperparameters.

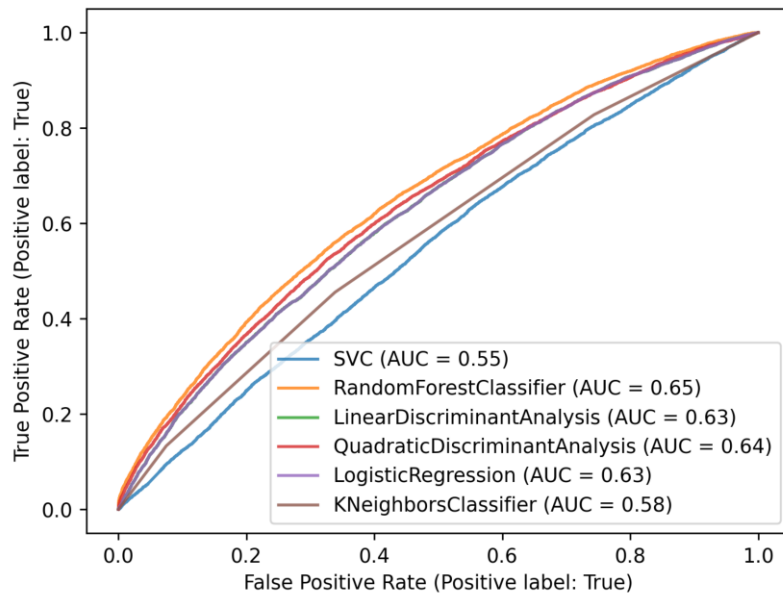


Figure A.18: Sarcasm classifier ROC curves

The random forest scheme has been picked to represent the sarcasm classifier version 1 because it achieves the best *AUC* score.

Evaluation

The previously trained random forest classifier reaches a recall of 39% and an accuracy of 63%. The following figure shows the corresponding confusion matrix:

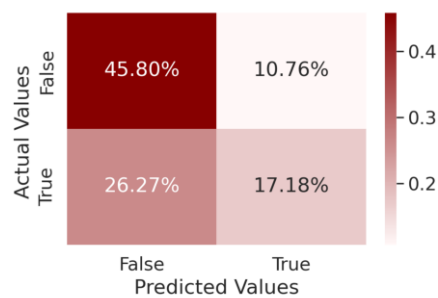


Figure A.19: Sarcasm classifier: confusion matrix

Because we are more interested to find the sarcastic comments, the decision threshold can be lowered. By doing that, the current point on the *ROC*-curve would move closer to the upper-right corner. The relationship between precision and recall is shown in the next figure:

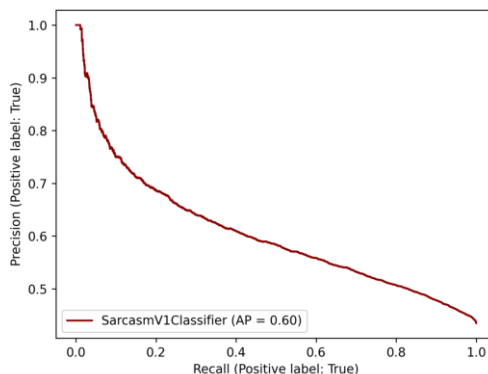


Figure A.20: Sarcasm classifier version 1 precision vs recall

Table A.9: Sarcasm classifier: scores

Metric	Score
precision	61%
recall	39%
f1-score	48%
accuracy	63%

To get a feeling of how the classifier performs, some made-up comments have been tested manually. This is how the classifier interprets the comments for a decision threshold of 50%:

Table A.10: Sarcasm classifier: test comments

Comment	Personal Oppinion	Pred. Prob.	Pred. Class
Freude, Kapitalismus!	Sarcastic	66%	True
Die gute alte britische Küche.	Grey Area	46%	False
Schlaf ist für die Schwachen.	Sarcastic	44%	False
Das Preis-Leistungsverhältnis stimmt bei Netflix nicht in der Schweiz :(Not Sarcastic	40%	False
Ich arbeite 40 Stunden pro Woche, um so arm zu sein.	Sarcastic	46%	False
Mir geht es großartig! Ich hoffe, dass ich mein ganzes Leben lang in der Käsekuchen Fabrik Kellnerin sein werde!	Sarcastic	45%	False
Ich liebe dieses Kleid. Das Design betont wirklich Ihr Doppelkinn.	Sarcastic	44%	False
Im Norden geht es mit viel Sonnenschein weiter, im Süden dominieren die Wolken.	Not Sarcastic	49%	False
Sie ist von Beruf her Lehrerin.	Not Sarcastic	55%	True
Was für eine Überraschung.	Sarcastic	99%	True

Limitations

The same limitations apply to this classifier as to the sarcasm classifier version 2. Those limitations can be found in Chapter 4.24.

A.3 Comment Metadata Classifier for "20 Minuten"

A.3.1 Idea

The metadata classifier for 20 Minuten follows the same idea as the metadata classifier for "Der Standard" described in Section 4.6.3. Therefore, the classifier for 20 Minuten is only described briefly in this section. The idea of this classifier is to predict whether a comment will be accepted or rejected by the 20 Minuten moderation by only analyzing the metadata of a comment.

A.3.2 Data Acquisition

This classifier uses the two million comments from the dump that 20 Minuten provided. This training data set is described in more detail in Section 3.5.2.

A.3.3 Feature Engineering

Although the data structure is the same as for the "der Standard" metadata classifier, not every feature could be reused. The comments from the 20 Minuten dump include a link to the article, but no additional information about the article itself (publication date, headline, ...). Therefore, features that are based on such information were not included. Also, features about the replies for a comment and features about the positive and negative reactions for a comment were not included: Those features are biased for rejected comments which were never publicly available or only available for a short period of time.

The set of remaining features is thus relatively small. From the set of features from the metadata classifier for "Der Standard", described in Section 4.6.3, only the following subset of features was used:

- 1: Number of comments from the same user on the same article
- 7: Is reply comment (0 or 1)?
- 8: Automated Readability Index
- 9: Number of characters
- 10: Number of Words
- 11: Average length of words
- 12: Percentage of all capital words with more than three letters
- 14: Contains emojis (0 or 1)
- 15: Number of exclamation marks
- 17: Number of comments from the same user during 12 hours before and 12 hours after the comment
- 19: Seconds passed since the previous comment from the same user
- 20: Seconds passed until the next comment from the same user
- 21: Average sentence length

A.3.4 Data Preparation

The features, again, were created with *SQL* queries and python. The features were scaled using python's `StandardScaler`.

A balanced dataset with 300'000 rejected comments and 300'000 non-rejected comments were used to build the classifier (randomly selected). Out of those 600'000 comments, 430'000 were used as training data (70%) and 180'000 as test data (30%)

A.3.5 Modeling

In a first step, multiple classifiers with default settings were trained on the data. The classifiers were trained to classify whether a comment is "ACCEPTED" or "REJECTED". The performance was very poor for all the classifiers:

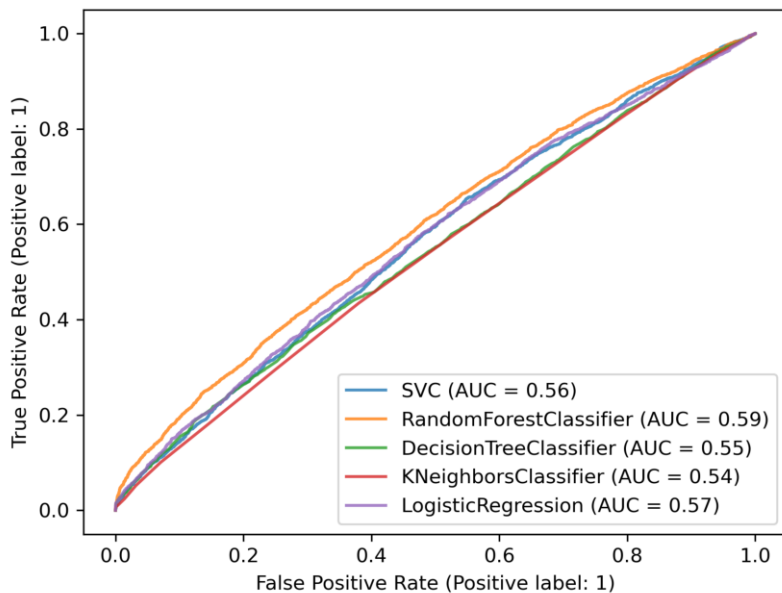


Figure A.21: ROC curve for different metadata classifiers

A random forest classifier got the best results overall. An exhaustive grid search was used to tune the hyperparameters for the random forest classifiers. In total, 1500 models were compared. The model with the highest precision was finally chosen.

A.3.6 Evaluation

Even after hyperparameter-optimization, the model performed poorly:

Table A.11: Scores for the metadata classifier (20 Minuten)

Metric	Score
precision	58%
recall	50%
f1-score	54%
accuracy	57%

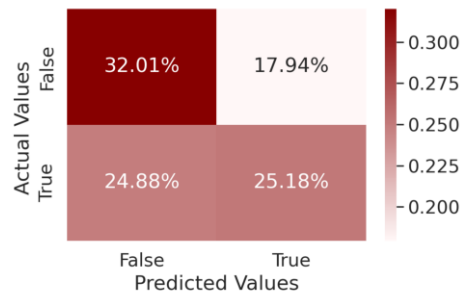


Figure A.22: Confusion matrix for the metadata classifier (20 Minuten)

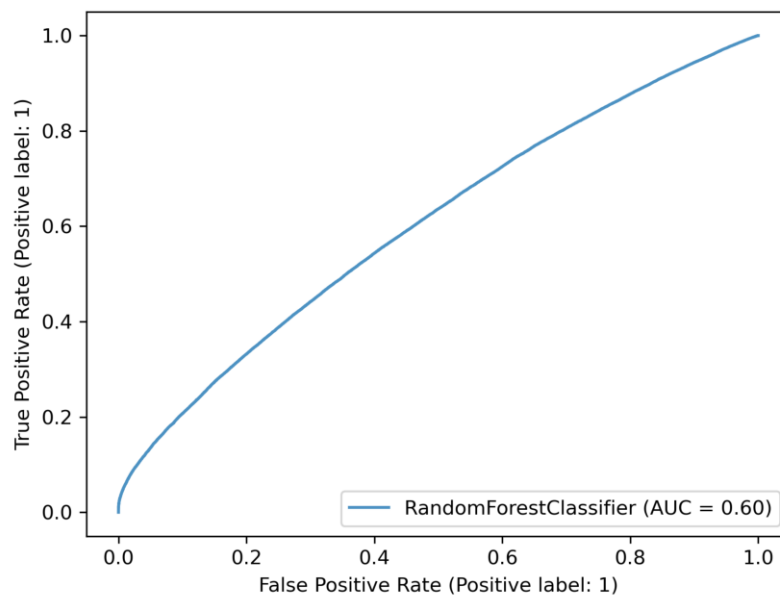


Figure A.23: ROC curve for the metadata classifier (20 Minuten)

A.3.7 Conclusion

Because of the poor performance, this classifier is not be included in the final application. Suspected reasons for the poor performance are:

- Not enough features were available. Rejected comments lack information about "likes" and "dislikes" which would probably be a good indicator.
- The training dataset contains mostly German comments but also some French comments, which distorts the results.
- The metadata of rejected comments on 20 Minuten is, overall, not very different from the metadata of accepted comments.

To conclude, a classifier for 20 Minuten working primarily with available metadata is not helpful in practice. For a better classification, the content of the comment must be analyzed as demonstrated for other classifiers in this thesis.

APPENDIX **B**

Code Structure

This chapter is intended for people who work with the code created for this thesis. It contains a description on how the code is organized.

B.1 Code Statistics

For this project, roughly ~17'000 lines of codes were written. The code is split between the different task areas as follows:

Table B.1: Distribution of the code per task

Data Acquisition	~3'000 lines of code
Data Understanding / Experiments	~4'500 lines of code
Classifiers / Training	~5'500 lines of code
Backend (API)	~2'000 lines of code
Frontend	~2'000 lines of code

The following languages were used:

Table B.2: Distribution of the code per language

Python	~11'000 lines of code
Java	~3'000 lines of code
TSX (React)	~1'500 lines of code
Configuration	~1'000 lines of code
Typescript	~500 lines of code

B.2 Overview of the repositories

The following image shows an overview of the code repositories created for this project:

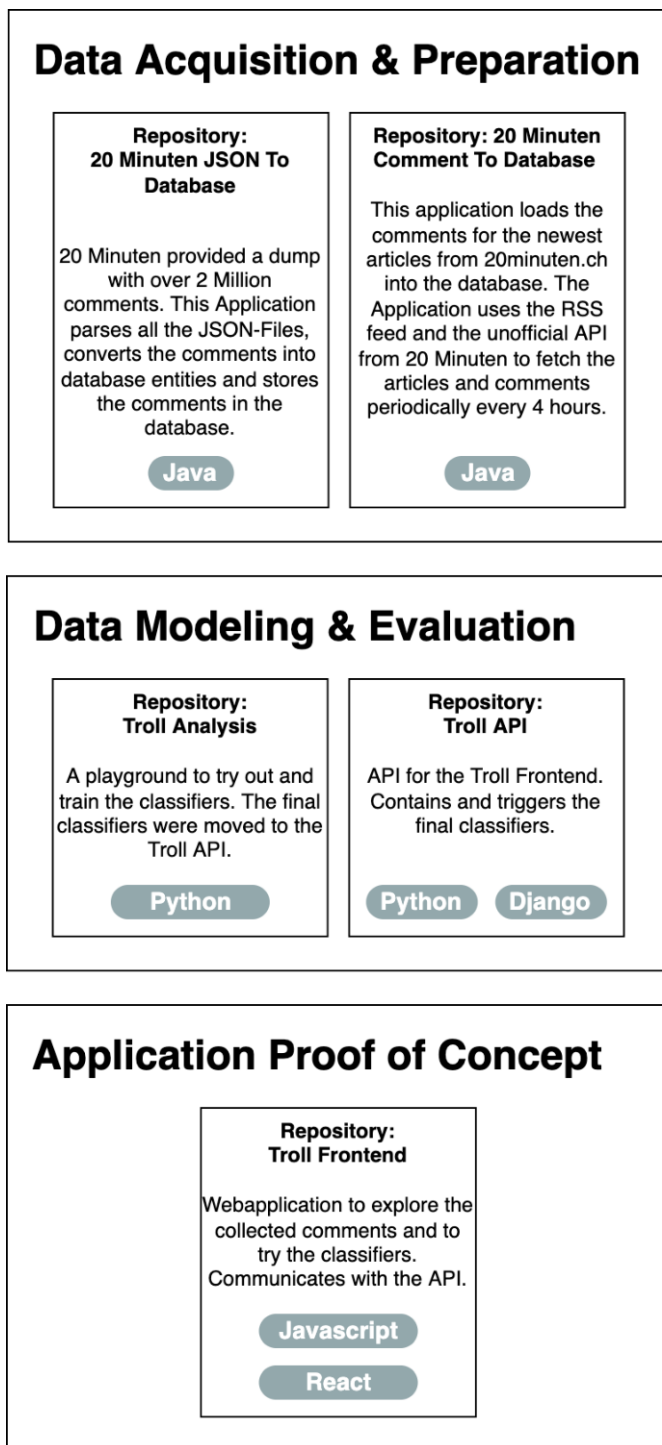


Figure B.1: Overview of the repositories

B.2.1 Repository: 20 Minuten JSON To Database

Described in Section: 3.6

Repository: <https://gitlab.ost.ch/ba-troll-detection/20-minuten-json-to-database>

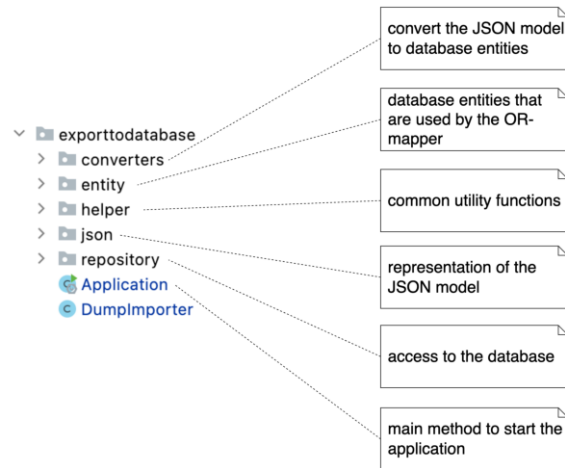


Figure B.2: Code structure of the 20min-JSON-to-database repository

B.2.2 Repository: 20 Minuten Comment To Database

Described in Section: 3.3

Repository: <https://gitlab.ost.ch/ba-troll-detection/newspaper-comment-to-database-fetcher>

Installation Guide: ??

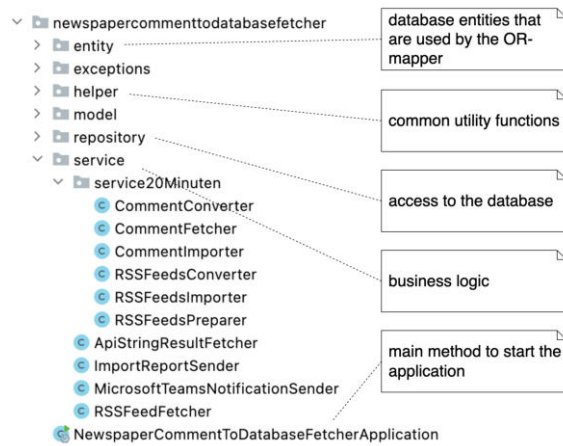


Figure B.3: Code structure of the 20min-Comment-to-database repository

B.2.3 Repository: Troll Analysis

Described in Sections: 4 and Appendix A

Repository: <https://gitlab.ost.ch/ba-troll-detection/troll-analysis>

Installation Guide: ??

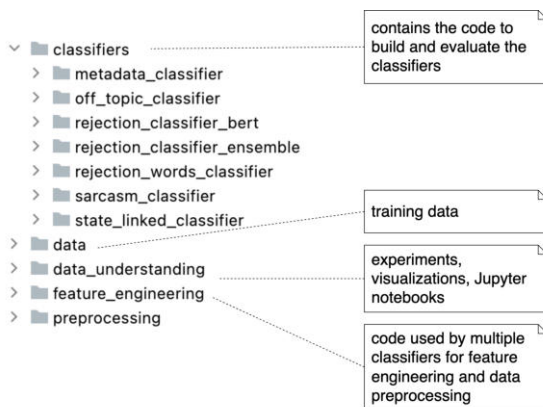


Figure B.4: Code structure of the troll analysis repository

B.2.4 Repository: Troll API

Described in Section: 5.4

Repository: <https://gitlab.ost.ch/ba-troll-detection/troll-api>

Installation Guide: ??

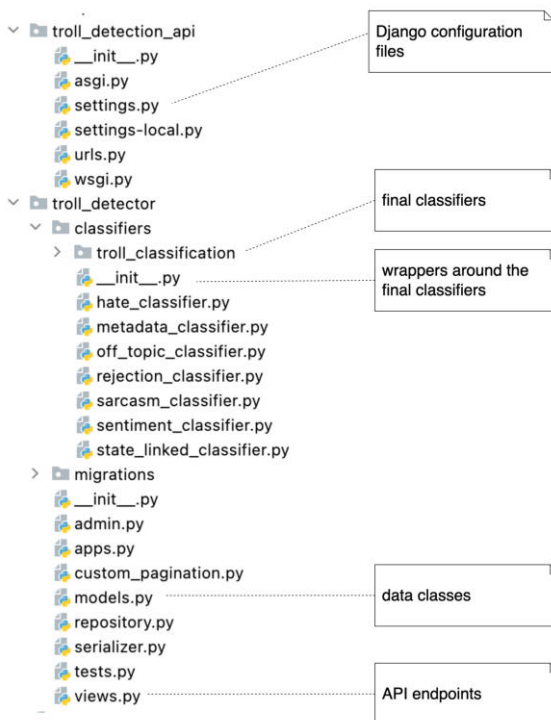


Figure B.5: Code structure of the troll api repository

B.2.5 Repository: Troll Frontend

Described in Section: 5.5

Repository: <https://gitlab.ost.ch/ba-troll-detection/troll-frontend>

Installation Guide: ??

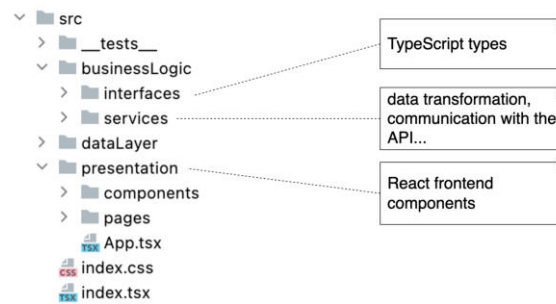


Figure B.6: Code structure of the troll frontend repository

APPENDIX C

Statistical Hypothesis Tests for the Metadata Classifier

This Chapter contains some hypothesis tests to decide whether or not a feature is meaningful. The hypothesis tests are referenced from Section 4.6.

C.1 Hypothesis Tests: Number of comments from the same user during 12 hours before and 12 hours after the comment

Hypothesis Test Number of comments from the same user during 12 hours before and 12 hours after the comment

Let

X = Values for this feature for troll comments

Y = Values for this feature for non-troll comments

be the two sample sets. The Shapiro scores

$$ShapiroScore_X \approx 0.59, \text{ with } p \approx 0$$

$$ShapiroScore_Y \approx 0.45, \text{ with } p \approx 0.0$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The hypotheses are formulated as:

H_0 : X and Y come from the same distribution

H_1 : X and Y come from different distributions

The Kolmogorov–Smirnov statistic calculated with SciPy¹

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.089}, \text{ with } p \approx \mathbf{7.40 * 10^{-4}}$$

shows that H_0 can be rejected because $p < 0.05$ [7]. Thus, the two distributions appear to be significantly different and the feature is **meaningful**.

¹<https://docs.scipy.org/doc/scipy/index.html>

C.2 Hypothesis Tests: Seconds passed since article publication

Hypothesis Test Seconds passed since article publication

Let

X = Values for this feature for troll comments

Y = Values for this feature for non-troll comments

be the two sample sets. The Shapiro scores

$$ShapiroScore_X \approx 0.070, \text{ with } p \approx 0$$

$$ShapiroScore_Y \approx 0.116, \text{ with } p \approx 0.0$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The hypotheses are formulated as:

H_0 : X and Y come from the same distribution

H_1 : X and Y come from different distributions

The Kolmogorov–Smirnov statistic calculated with SciPy²

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.1218}, \text{ with } p \approx \mathbf{5.2243 * 10^{-7}}$$

shows that H_0 can be rejected because $p < 0.05$ [7]. Thus, the two distributions appear to be significantly different and the feature is **meaningful**.

²<https://docs.scipy.org/doc/scipy/index.html>

C.3 Hypothesis Tests: Seconds passed since the previous comment from the same user

Hypothesis Test Seconds passed since the previous comment from the same user

Let

X = Values for this feature for troll comments

Y = Values for this feature for non-troll comments

be the two sample sets. The Shapiro scores

$$ShapiroScore_X \approx 0.3043, \text{ with } p \approx 0$$

$$ShapiroScore_Y \approx 0.2958, \text{ with } p \approx 0.0$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The hypotheses are formulated as:

H_0 : X and Y come from the same distribution

H_1 : X and Y come from different distributions

The Kolmogorov–Smirnov statistic calculated with SciPy³

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.1245}, \text{ with } p \approx \mathbf{4.4749 * 10^{-7}}$$

shows that H_0 can be rejected because $p < 0.05$ [7]. Thus, the two distributions appear to be significantly different and the feature is **meaningful**.

³<https://docs.scipy.org/doc/scipy/index.html>

C.4 Hypothesis Tests: Seconds passed until the next comment from the same user

Hypothesis Test Seconds passed until the next comment from the same user

Let

X = Values for this feature for troll comments

Y = Values for this feature for non-troll comments

be the two sample sets. The Shapiro scores

$$\text{ShapiroScore}_X \approx 0.6718, \text{ with } p \approx 0$$

$$\text{ShapiroScore}_Y \approx 0.4227, \text{ with } p \approx 0.0$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The hypotheses are formulated as:

H_0 : X and Y come from the same distribution

H_1 : X and Y come from different distributions

The Kolmogorov–Smirnov statistic calculated with SciPy⁴

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.141994}, \text{ with } p \approx \mathbf{0.00}$$

shows that H_0 can be rejected because $p < 0.05$ [7]. Thus, the two distributions appear to be significantly different and the feature is **meaningful**.

⁴<https://docs.scipy.org/doc/scipy/index.html>

C.5 Hypothesis Tests: Average sentence length

Hypothesis Test Average sentence length

Let

X = Values for this feature for troll comments

Y = Values for this feature for non-troll comments

be the two sample sets. The Shapiro scores

$$\text{ShapiroScore}_X \approx 0.8773, \text{ with } p \approx 0$$

$$\text{ShapiroScore}_Y \approx 0.7042, \text{ with } p \approx 0.0$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The hypotheses are formulated as:

H_0 : X and Y come from the same distribution

H_1 : X and Y come from different distributions

The Kolmogorov–Smirnov statistic calculated with SciPy⁵

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.042515}, \text{ with } p \approx \mathbf{0.286874}$$

shows that H_0 can be accepted because $p > 0.05$ [7]. Thus, the two distributions appear not to be significantly different and the feature is **not meaningful**.

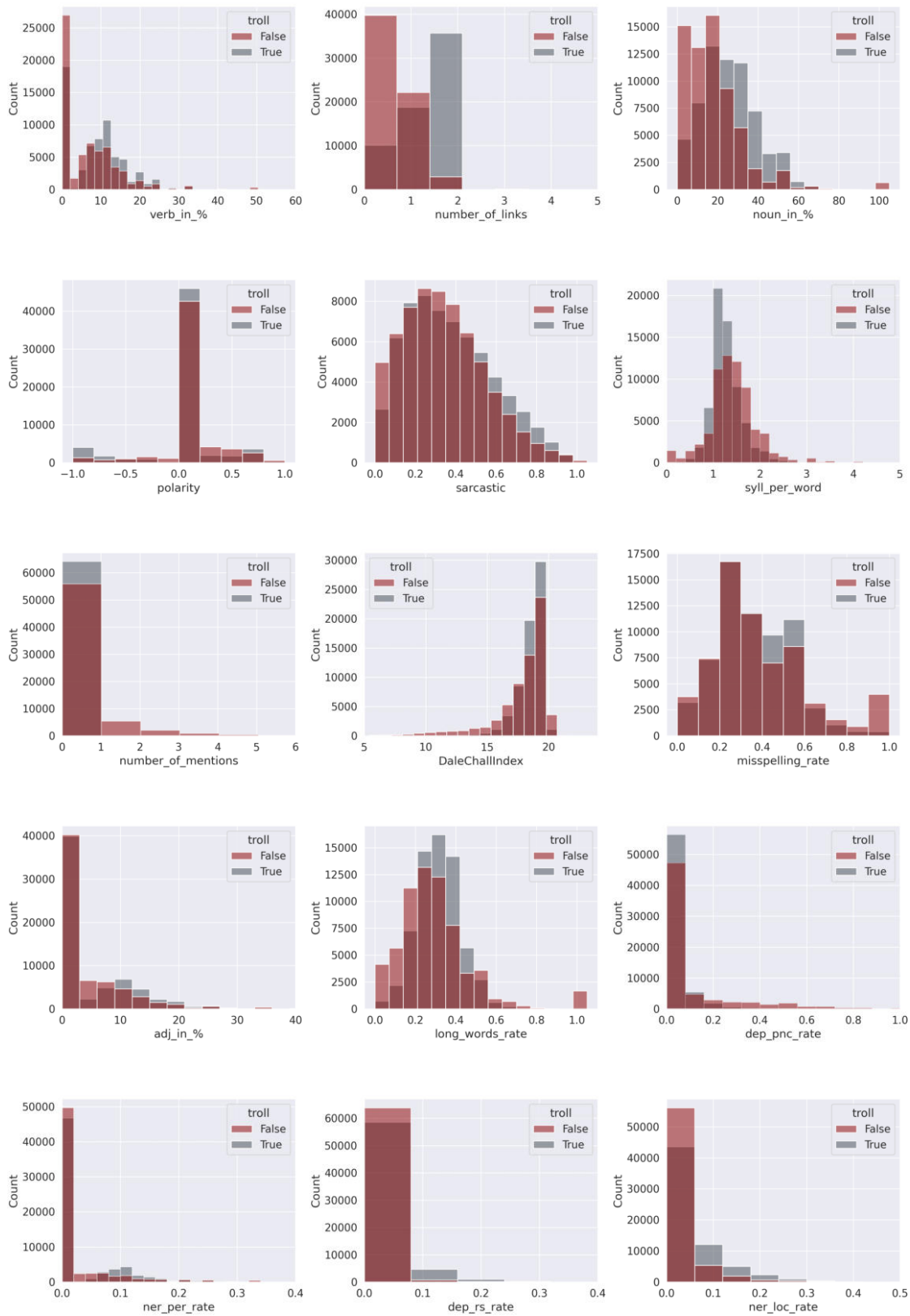
⁵<https://docs.scipy.org/doc/scipy/index.html>

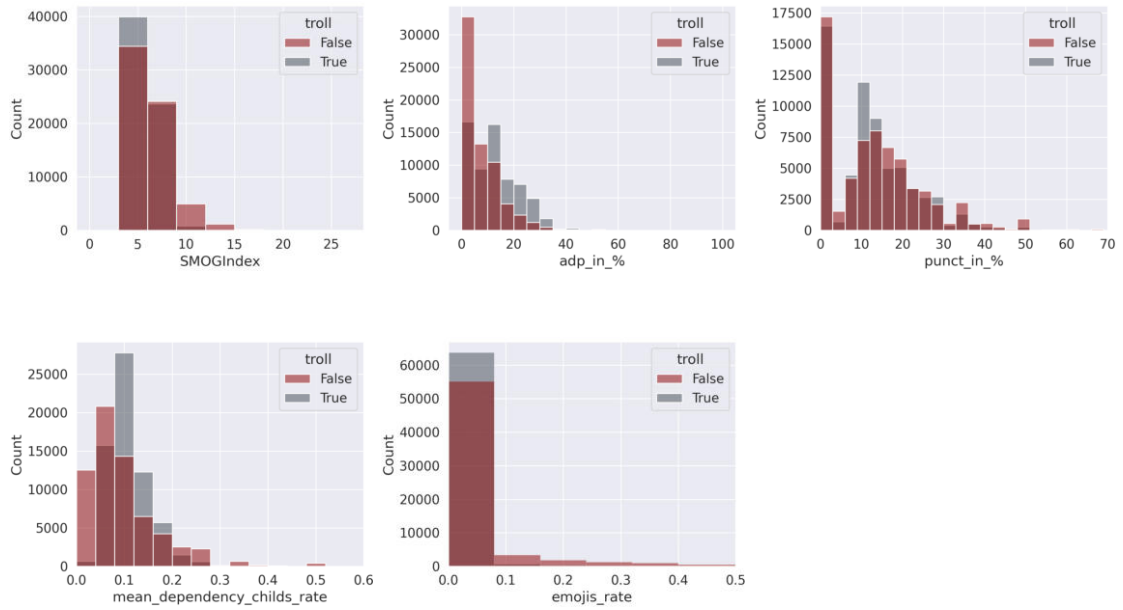
APPENDIX **D**

Statistical Tests and Feature Plots State-Linked Classifier

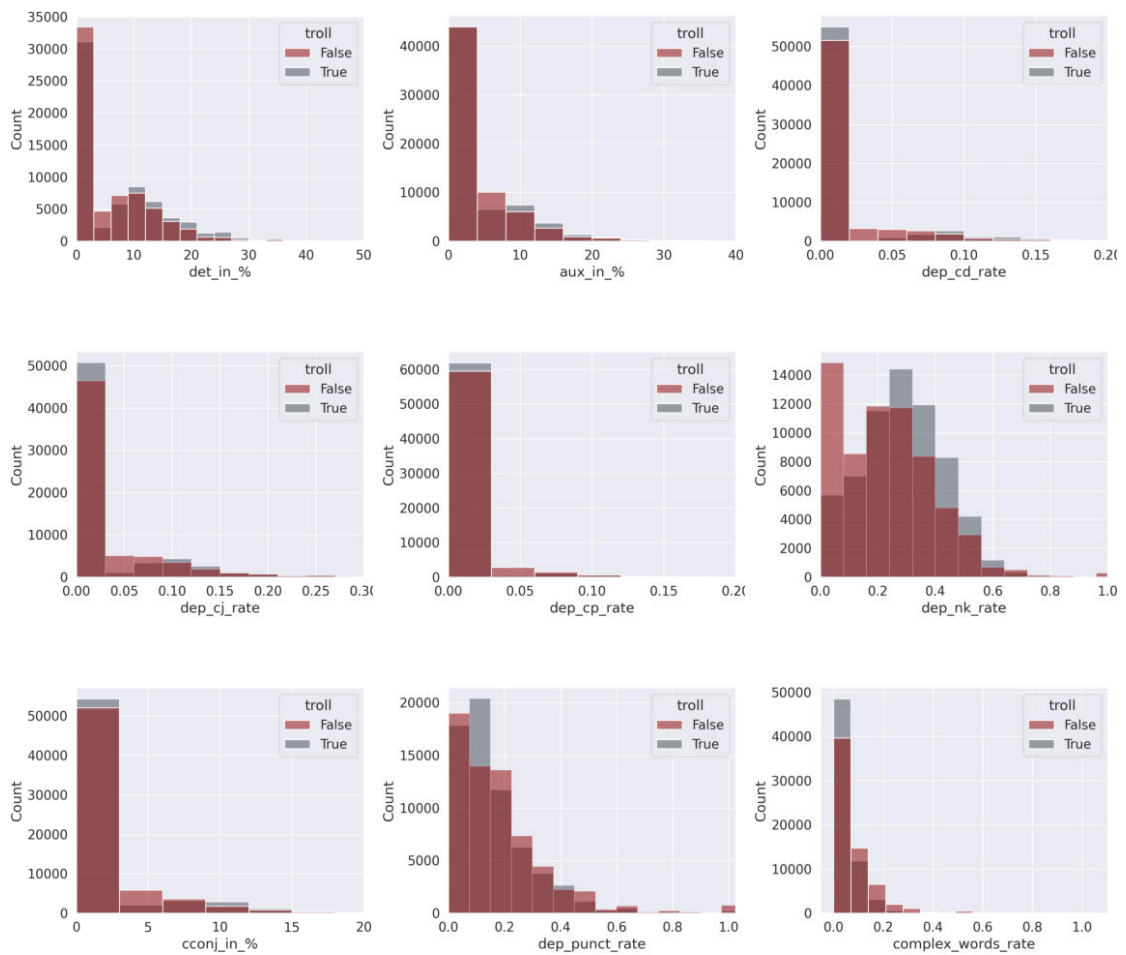
The plots from subsection "Used Features" show how the features contribute to the overall prediction of the syntactic and sentiment classifier.

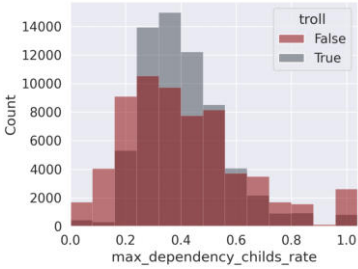
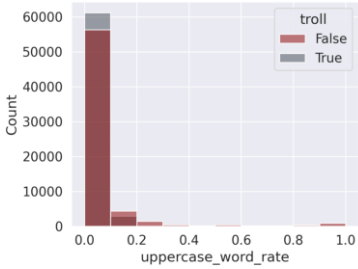
D.1 Used Features





D.1.1 Unused Features





D.2 Hypotheses Tests

Unless stated otherwise, the following applies for all hypothesis tests in this subsection. Let

$$X = \textit{State Linked Trolls}$$

$$Y = \textit{Regular Users}$$

be the two sample sets, then the hypotheses are formulated like

$$H_0 : X \textit{ and } Y \textit{ come from the same distribution}$$

$$H_1 : X \textit{ and } Y \textit{ come from different distributions}$$

All hypothesis tests were performed with 2000 randomly picked samples. That is 1000 samples for each class.

Number of Links

Hypothesis Test number_of_links

The Shapiro scores

$$\textit{ShapiroScore}_X \approx 0.729, \text{ with } p \approx 1.909 * 10^{-37}$$

$$\textit{ShapiroScore}_Y \approx 0.678, \text{ with } p \approx 7.294 * 10^{-40}$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.524}, \text{ with } p \approx \mathbf{9.135 * 10^{-126}}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Sarcastic

Hypothesis Test sarcastic

The Shapiro scores

$$\textit{ShapiroScore}_X \approx 0.972, \text{ with } p \approx 4.853 * 10^{-13}$$

$$\textit{ShapiroScore}_Y \approx 0.973, \text{ with } p \approx 8.519 * 10^{-13}$$

show, that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.105}, \text{ with } p \approx \mathbf{3.210 * 10^{-5}}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Number of Mentions

Hypothesis Test number_of_mentions

The Shapiro scores

$$ShapiroScore_X \approx 0.972, \text{ with } p \approx 0.000$$

$$ShapiroScore_Y \approx 0.973, \text{ with } p \approx 0.000$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.387}, \text{ with } p \approx \mathbf{3.655 * 10^{-67}}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Dependency Parsing Proper Noun Component Rate

Hypothesis Test dep_pnc_rate

The Shapiro scores

$$ShapiroScore_X \approx 0.972, \text{ with } p \approx 0.000$$

$$ShapiroScore_Y \approx 0.973, \text{ with } p \approx 2.102 * 10^{-44}$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.182}, \text{ with } p \approx \mathbf{6.955 * 10^{-15}}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Misspelling Rate

Hypothesis Test misspelling_rate

The Shapiro scores

$$ShapiroScore_X \approx 0.982, \text{ with } p \approx 1.143 * 10^{-9}$$

$$ShapiroScore_Y \approx 0.930, \text{ with } p \approx 2.102 * 10^{-22}$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.075}, \text{ with } p \approx \mathbf{0.007}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Syllables per Word

Hypothesis Test syll_per_word

The Shapiro scores

$$ShapiroScore_X \approx 0.925, \text{ with } p \approx 5.758 * 10^{-22}$$

$$ShapiroScore_Y \approx 0.927, \text{ with } p \approx 1.155 * 10^{-22}$$

show, that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.245}, \text{ with } p \approx \mathbf{9.521 * 10^{-27}}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Punctuation in %

Hypothesis Test punct_in_%

The Shapiro scores

$$ShapiroScore_X \approx 0.919, \text{ with } p \approx 1.113 * 10^{-22}$$

$$ShapiroScore_Y \approx 0.900, \text{ with } p \approx 5.647 * 10^{-25}$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.074}, \text{ with } p \approx \mathbf{0.008}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Nouns in %

Hypothesis Test noun_in_%

The Shapiro scores

$$ShapiroScore_X \approx 0.975, \text{ with } p \approx 4.913 * 10^{-12}$$

$$ShapiroScore_Y \approx 0.566, \text{ with } p \approx 1.680 * 10^{-28}$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.283}, \text{ with } p \approx \mathbf{1.141^{-35}}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Adpositions in %

Hypothesis Test adp_in_%

The Shapiro scores

$$\text{ShapiroScore}_X \approx 0.929, \text{ with } p \approx 2.415 * 10^{-21}$$

$$\text{ShapiroScore}_Y \approx 0.797, \text{ with } p \approx 1.496 * 10^{-33}$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.316}, \text{ with } p \approx \mathbf{1.602}^{-44}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Emojis Rate

Hypothesis Test emojis_rate

The Shapiro scores

$$\text{ShapiroScore}_X \approx 0.122, \text{ with } p \approx 0.000$$

$$\text{ShapiroScore}_Y \approx 0.353, \text{ with } p \approx 0.000$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.175}, \text{ with } p \approx \mathbf{8.685}^{-14}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Dale-Chall Readability Index

Hypothesis Test DaleChallIndex

The Shapiro scores

$$\text{ShapiroScore}_X \approx 0.715, \text{ with } p \approx 4.137 * 10^{-38}$$

$$\text{ShapiroScore}_Y \approx 0.749, \text{ with } p \approx 2.323 * 10^{-36}$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.200}, \text{ with } p \approx \mathbf{2.418}^{-19}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Named Entity Recognition Location Rate

Hypothesis Test ner_loc_rate

The Shapiro scores

$$ShapiroScore_X \approx 0.679, \text{ with } p \approx 9.650 * 10^{-40}$$

$$ShapiroScore_Y \approx 0.367, \text{ with } p \approx 0.000$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.196}, \text{ with } p \approx \mathbf{3.290}^{-17}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Named Entity Rate Person Rate

Hypothesis Test ner_per_rate

The Shapiro scores

$$ShapiroScore_X \approx 0.593, \text{ with } p \approx 2.915 * 10^{-43}$$

$$ShapiroScore_Y \approx 0.411, \text{ with } p \approx 0.000$$

show, that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.114}, \text{ with } p \approx \mathbf{4.441}^{-6}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Long Words Rate

Hypothesis Test long_words_rate

The Shapiro scores

$$ShapiroScore_X \approx 0.973, \text{ with } p \approx 1.214 * 10^{-12}$$

$$ShapiroScore_Y \approx 0.888, \text{ with } p \approx 2.929 * 10^{-26}$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.200}, \text{ with } p \approx \mathbf{6.613}^{-18}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Mean Dependency Childs Rate

Hypothesis Test mean_dependency_childs_rate

The Shapiro scores

$$ShapiroScore_X \approx 0.854, \text{ with } p \approx 2.376 * 10^{-29}$$

$$ShapiroScore_Y \approx 0.790, \text{ with } p \approx 5.094 * 10^{-34}$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.289}, \text{ with } p \approx \mathbf{3.347}^{-37}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

SMOG Readability Index

Hypothesis Test SMOGIndex

The Shapiro scores

$$ShapiroScore_X \approx 0.773, \text{ with } p \approx 5.151 * 10^{-35}$$

$$ShapiroScore_Y \approx 0.827, \text{ with } p \approx 1.612 * 10^{-31}$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.203}, \text{ with } p \approx \mathbf{1.942}^{-18}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Verbs in %

Hypothesis Test verb_in_%

The Shapiro scores

$$ShapiroScore_X \approx 0.862, \text{ with } p \approx 8.916 * 10^{-29}$$

$$ShapiroScore_Y \approx 0.734, \text{ with } p \approx 3.355 * 10^{-37}$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.218}, \text{ with } p \approx \mathbf{3.203}^{-21}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Conjunctions in %

Hypothesis Test conj_in_%

The Shapiro scores

$$ShapiroScore_X \approx 0.470, \text{ with } p \approx 0.000$$

$$ShapiroScore_Y \approx 0.514, \text{ with } p \approx 0.000$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.059}, \text{ with } p \approx \mathbf{0.062}$$

shows that H_0 can **not** be rejected because $p > 0.05$ [7]. Hence this feature is omitted in the classifier.

Polarity

Hypothesis Test polarity

The Shapiro scores

$$ShapiroScore_X \approx 0.754, \text{ with } p \approx 4.291 * 10^{-36}$$

$$ShapiroScore_Y \approx 0.795, \text{ with } p \approx 1.142 * 10^{-33}$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.141}, \text{ with } p \approx \mathbf{4.389 * 10^{-09}}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Adjectives in %

Hypothesis Test adj_in_%

The Shapiro scores

$$ShapiroScore_X \approx 0.710, \text{ with } p \approx 2.307 * 10^{-38}$$

$$ShapiroScore_Y \approx 0.700, \text{ with } p \approx 7.890 * 10^{-39}$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.087}, \text{ with } p \approx \mathbf{0.001}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

Dependency Parsing Reported Speech Component Rate

Hypothesis Test dep_rs_rate

The Shapiro scores

$$ShapiroScore_X \approx 0.372, \text{ with } p \approx 0.000$$

$$ShapiroScore_Y \approx 0.060, \text{ with } p \approx 0.000$$

show that both sample sets are not normal distributed because $p < 0.05$ [10]. Hence the KS test according to Chapter 1.2.2 is taken to evaluate whether X and Y come from the same distribution. The Kolmogorov–Smirnov statistic calculated with SciPy

$$D_{X,Y} = \sup_x |F_X(x) - F_Y(x)| \approx \mathbf{0.090}, \text{ with } p \approx \mathbf{0.001}$$

shows that H_0 can be rejected because $p < 0.05$ [7].

APPENDIX **E**

Troll User Features

This chapter contains ideas for a user-based troll detection approach.

E.1 User Features

In section 4.6.3, it is described which metadata features were used to detect troll comments (post-based approach). A user-based approach was outside the scope of this project. However, a list was created with some features that could be used for a user-based approach.

Many user features can be extracted by aggregating the comment-features of a user (sum, maximum, average, median, minimum, etc.). This list only contains unique features (no aggregated comment-features).

Table E.1: Features for a user

Nr.	Feature	Description and Source
1	Number of days in which the user posted at least one comment	If a user actively comments on articles over a long period of time, this might indicate that writing comments is the job of this user [1] [83]. This requires to have data about a long time span.
2	Number of threads in which a user participated	Trolls typically comment on fewer threads but typically write more comments on a single thread than normal users [1]. This requires to have data about a long time span.
3	Number of times in which a user is among the first 10 users to comment on a discussion thread	Trolls might try to be among the first users to comment on a topic to increase their visibility [1] [82] [83].
4	Average number of comments in active days	One study has stated that paid trolls write twice as many comments on a single day than normal users [82].
5	Total number of comments	Troll users write more comments than average users [83]. This feature requires having data about a long time span.
6	Standard derivation from comment publication time	Users that write comments not only at specific times but throughout the day are suspected to be paid trolls [89].
7	Average number of posts per active thread	Trolls concentrate on a few discussions but often write more than one post per discussion [1].

APPENDIX **F**

Test Logs

F.1 Test Logs for the Periodic Import of 20 Minuten Comments (33 Tests)

F.1.1 DateHelperTests.java

✓ DateHelperTests	3 ms
✓ ISO8601DateCanBeConverted()	3 ms

Figure F.1: Test logs: DateHelperTests

F.1.2 StringHelperTests.java

✓ StringHelperTests	26 ms
✓ stringConcatenationWorksForEmptyList()	19 ms
✓ stringConcatenationWorksForFilledList()	7 ms

Figure F.2: Test logs: StringHelperTests

F.1.3 NewspaperRepositoryTests.java

✓ NewspaperRepositoryTests	2 sec 724 ms
✓ testDatabasesEmpty()	1 sec 604 ms
✓ newspaperCanBePersisted()	614 ms
✓ duplicateNewspaperIsOnlySavedOnce()	83 ms
✓ firstStoredAtAttributesAutomaticallySet()	263 ms
✓ contextLoads()	8 ms
✓ persistCascadeWorks()	152 ms

Figure F.3: Test logs: NewspaperRepositoryTests

F.1.4 CommentConverterTests.java

✓ CommentConverterTest	582 ms
✓ nullReactionsAreCorrectlySummarized()	456 ms
✓ answerCommentsAreRecursivelyConverted()	18 ms
✓ duplicateAuthorsAreOnlyConvertedOnce()	62 ms
✓ apiResponseCanBeConvertedToDatabaseEntities()	42 ms
✓ reactionsAreCorrectlySummarized()	4 ms

Figure F.4: Test logs: CommentConverterTests

F.1.5 CommentFetcherTests.java

✓ CommentFetcherTests	732 ms
✓ validApiResponseCanBeMappedToObject()	627 ms
✓ exceptionFromInvalidApiRequestIsRethrown()	75 ms
✓ invalidApiResponseThrowsException()	30 ms

Figure F.5: Test logs: CommentFetcherTests

F.1.6 RSSFeedsConverterTests.java

✓ RSSFeedsConverterTests	1sec 231ms
✓ faultyFeedGetsAddedToList()	826 ms
✓ duplicateArticlesAreMerged()	49 ms
✓ hasNoFeedWithErrors()	47 ms
✓ topicWereMerged()	49 ms
✓ articleAttributesAreCorrectlyMapped()	17 ms
✓ feedsWereExtractedCorrectly()	70 ms
✓ articleTopicsAreCorrectlyMapped()	7 ms
✓ conversionOfFeedsHasNoErrors()	7 ms
✓ articlePublicationDatelsCorrectlyConverted()	65 ms
✓ someArticlesAreConvertedEvenIfAFeedFails()	39 ms
✓ faultyFeedsAreDetected()	55 ms

Figure F.6: Test logs: RSSFeedsConverterTests

F.1.7 RSSFeedsImporterTests.java

✓ RSSFeedsImporterTests	912 ms
✓ oldArticleThatIsInDatabasesIsNotImported()	686 ms
✓ commentImporterWasCalled()	87 ms
✓ feedFetcherWasCalled()	72 ms
✓ importReportSenderWasCalled()	53 ms
✓ amountOfImportedArticlesIsCorrect()	14 ms

Figure F.7: Test logs: RSSFeedsImporterTests

F.2 Test logs for Classifiers/API (23 tests)

F.2.1 Classify Text

- analyze_text_off_topic_classifier_with_article: success
- analyze_text_hater_classifier: success
- analyze_text_state_linked_classifier: success
- analyze_text_sarcasm_classifier: success
- analyze_text_sentiment_classifier: success

F.2.2 Classify Comment From Database

- analyze_comment_metadata_classifier_is_None_for_20min: success
- analyze_comment_metadata_classifier_for_der_Standard: success
- analyze_comment_off_topic_classifier: success
- analyze_comment_hate_classifier: success
- analyze_comment_state_linked_classifier: success
- analyze_comment_sarcasm_classifier: success
- analyze_comment_sentiment_classifier: success

F.2.3 Error Handling

- analyze_text_off_topic_classifier_with_no_article_raises_exception: success
- analyze_text_off_topic_classifier_with_no_text_raises_exception: success

F.2.4 Explore Comments

- get_newssources: success
- get_articles: success
- get_articles_with_paging: success
- get_comments: success
- get_comments_with_paging: success
- get_comment_statistics: success

F.2.5 Explore Users

- get_users: success
- get_users_with_paging: success
- get_user_statistics: success

APPENDIX **G**

Literature Review on Data Analysis

This chapter was written as a study resource during the familiarization with the topic of Data Analysis. It contains more general concepts that are not directly related to the topic "troll detection" but helped us to better understand the topic in general.

This chapter focuses on the understanding of the critical concepts and algorithms helpful for the detection of trolls. The application of these concepts with adequate data follows in subsequent sections.

The first section (G.2) shows techniques to recognize patterns in the data. This can help to find useful metrics and features that help to identify troll users. Visualization of the data might give a first impression of how the data is distributed. Section G.3 discusses the use of statistical machine learning models to predict the presence of trolls with classifiers. Supervised as well as unsupervised algorithms are presented, as both might play an important role in this project.

The algorithms presented in this section do not aim to be a complete reference. However, describing the algorithms is a way of evaluating whether an algorithm might be useful for the project or not.

The last section (G.4) discusses the results and evaluates which described algorithms can actually contribute to the project.

G.1 Variations of Data Collection

Three types of data are defined by Harvard Business School [90]:

- **First-party data:** collected by the party seeking data.
- **Second-party data:** shared by another party.
- **Third-party data:** rented or sold by another party without having a link to the receiver.

For the purposes of this project, first-party and second-party data is feasible.

G.1.1 First-party Data

First-party data is difficult to collect since it requires legal clarifications to directly access the comment databases of German-language newspapers. However, data can be acquired using the newspaper's APIs or by scraping their websites. For example scraping is used to extract data from websites and transform them into an understandable format, often spreadsheets, databases or *CSV* files are used. It is divided into the following nine techniques [91]:

- **Traditional copy and paste:** manually copy pasting from websites
- **Text grapping and *regex*:** *unix* command or *regex*-matching
- ***HTML* Parsing:** parsing websites using *XQuery* and *HTQL*
- ***DOM* Parsing:** using a web browser, programs can parse web pages into a *DOM* tree
- **Web Scraping Software:** software tools allow web scraping without having to manually write code
- **Computer vision web-page analysers:** using machine learning and computer vision to retrieve data like a human would

G.1.2 Second-party Data

Second-party data can be obtained by contacting newspapers and asking for the relevant information.

G.2 Exploratory Analysis

G.2.1 Description

Exploratory data analysis (*EDA*) is used to identify characteristics in datasets, visualization methods are often used to achieve this. It allows discovering patterns, relationships, and anomalies. *EDA* is also used to validate a hypothesis and assumptions [92]. Regarding troll-detection, there are some assumptions on which metrics might have an impact in the possibility of a user being a troll.

G.2.2 Clustering

Overview

A cluster is a subset of objects which are similar to each other and as different as possible to objects from other subsets. The aim of clustering is to detect groups of similar objects and identify similarities of objects [93]. There are multiple clustering techniques, presented in the following subsections. Much of the data collected for this project is not labeled. Clustering therefore is a helpful method to validate whether the comments and data about the comments can be split into two distinct clusters (troll and not-troll) or if the differences are not that obvious.

Partition Based Clustering

Partition based clustering creates k partitions in a given dataset of n objects. Each object is assigned to a partition at the beginning. Through an iterative process, each object gets relocated to an appropriate partition. The partitions can be represented by a centroid or medoid. There exist many partition-based clustering algorithms:

K-Means K-Means is an algorithm which partitions a dataset into a predefined number of clusters. Each partition is represented by a centroid, the cluster's center. K-Means is *NP*-hard, and it is sensitive to outliers. The algorithm contains the following steps:

- Choose k objects to be the cluster-centers (random or with using the *k-means++* algorithm).
- Calculate the squared Euclidean distance (see Section G.1) for each object in the dataset to the cluster-centers.

$$d(p, q) = \sum_{i=1}^n (q_i - p_i)^2 \quad (\text{G.1})$$

- Assign each object to the cluster with the least distance to the object.
- Update the cluster centers by calculating the new cluster center using the average of all objects in that particular cluster.
- Repeat Step 2 – 4 until the cluster center does not move, or another stopping criterion is met.

K-Medoid The K-Medoid is similar to the K-Means algorithm. It uses a Medoid instead of a centroid for the cluster centers. The algorithm contains the following steps:

- Choose arbitrary k objects to be the cluster centers
- Calculate the Manhattan distance (see Section G.2) for each object in the dataset to the cluster centers.

$$d(p, q) = \sum_{i=1}^n |p_i - q_i| \quad (\text{G.2})$$

- Assign each object to the cluster with the least distance.
- Calculate the cost by summing the distance from each object to its cluster center.
- Choose a new cluster center
- Repeat steps 3 and 4 with the new cluster center
- If the cost is lower than the cost of the previous cluster centers the algorithm can be stopped or steps 5 – 7 can be repeated until a stopping criterion is met.

Hierarchical Clustering

Two approaches Hierarchical clustering can be implemented using a top-down or bottom-up approach:

- **bottom-up:** The data objects are recursively compared to other objects/ clusters and merged if they are similar enough until only one cluster remains
- **top-down:** Starts with all the data object in one cluster and splits them recursively until a cluster has only one data object.

The resulting output of either one is usually visualized as a *dendrogram*. It is assumed that the result is monotonic, meaning that if s_1, s_2, \dots, s_{K-1} are successive then $s_1 \geq s_2 \geq \dots \geq s_{K-1}$ holds, s being the similarities.

The dendrogram can either be cut by a level of similarity or by the desired number of clusters. A large gap in similarity between clusters is an indication for a natural clustering. By applying

$$K = \arg \min_{K'} [RSS(K') + \lambda K'] \quad (\text{G.3})$$

the cut in the hierarchy can be determined. K represents the cut of the hierarchy, K' is the number of clusters and λ is used for the penalty for each additional cluster. RSS can be replaced with another measure of distortion.

Agglomerative Clustering Agglomerative clustering uses bottom-up to determine which objects to merge. The following algorithms can be used:

- **Single-link:** Only the most similar members of the two clusters are considered.
- **Complete-link:** Only the most dissimilar members of the two clusters are considered.
- **Group-average:** Average similarity of all members of the two clusters are considered, even to those in the same cluster but without comparing to itself. The following equation computes the group-average of two clusters:

$$\text{sim-ga}(w_i, w_j) = \frac{1}{(N_i + N_j)(N_i + N_j - 1)} \sum_{d_k \in w_i \cup w_j} \sum_{d_l \in w_i \cup w_j, d_l \neq d_k} \vec{d}_k \cdot \vec{d}_l \quad (\text{G.4})$$

\vec{d} is the length-normalized vector of the objects. N_i and N_j are the number of objects in w_i and w_j .

- **Centroid similarity:** Similarity of the centroids of the two clusters are considered. The following equation computes the centroid similarity of two clusters:

$$\text{sim-cent}(w_i, w_j) = \frac{1}{N_i N_j} \sum_{d_k \in w_i} \sum_{d_l \in w_j} \vec{d}_k \cdot \vec{d}_l \quad (\text{G.5})$$

Divisive clustering Divisive clustering uses the top-down approach. At the beginning, all objects are in one cluster, which is then split recursively using a partition-based clustering algorithm until the desired number of clusters is reached, or the clusters only contain a single object. An advantage of divisive clustering is the efficiency. Top-down algorithms have a complexity of $O(n)$ compared to agglomerative clustering, which have at least a complexity of $O(n^2)$ [94].

Model Based Clustering

Model based clustering assumes that the dataset was generated according to a model. It then attempts to recreate the original model and creates assignments to clusters. The following equation maximizes the log likelihood of generating the data D :

$$\Theta = \arg \max_{\Theta} L(D|\Theta) = \arg \max_{\Theta} \log \prod_{n=1}^N P(d_n|\Theta) = \arg \max_{\Theta} \sum_{n=1}^N \log P(d_n|\Theta) \quad (\text{G.6})$$

The goodness of the found clusters is measured by $L(D|\Theta)$. Θ are the model parameters as in $\Theta = \{\vec{\mu}_1, \dots, \vec{\mu}_K\}$. [94]

Expectation Maximization The expectation maximization algorithm (*EM*) iteratively maximizes $L(D|\Theta)$. *EM* is often applied to a mixture model. It is assumed that the data points to differentiate follow a different probabilistic model than the other points. By estimating that model, the data points can be assigned to the respective model and hence be clustered. *EM* has generally two steps, described below: the expectation step and the maximization step. *EM* uses soft assignment, which means that the data objects have an assigned probability for each cluster to be part of [94]. The algorithm takes the following steps:

1. Guess Θ (model parameters).
2. Expectation step: Evaluate probability of each data object belonging to each model.
3. Maximization step: Use the probability to re-evaluate the models.
4. Use log likelihood and check for convergence of the parameters. If not converged, go to step 2 [95].

Density Based Clustering

Density based clustering can find arbitrary shaped clusters, by comparing dense areas with less populated areas in a dataset. One scan is enough to cluster the data object. The number of clusters is detected automatically by the number of separate dense areas.

DBSCAN Density-Based Spatial Clustering of Applications with Noise (DBSCAN) only uses two parameters to cluster a dataset. ϵ is the farthest distance a point can have to its cluster. N_{min} is the minimum number of data points to form a cluster.

ϵ -neighborhood of a point $N_{\epsilon}(x)$ is the ϵ -neighborhood of a point x , D is the dataset and d is a distance function.

$$N_{\epsilon}(x) = \{y \in D : d(x, y) \leq \epsilon\} \quad (\text{G.7})$$

Directly density-reachable A point is directly density-reachable if the following two requirements hold:

1. $x \in N_{\epsilon}(y)$
2. $|N_{\epsilon}(y)| \geq N_{min}$

$|N_{\epsilon}(y)|$ is the number of elements in $N_{\epsilon}(y)$.

Density-reachable A point x is density-reachable to a point y if there is a link of directly density-reachable points from x to y .

Density-connected A point x is density-connected to a point y if there is a point z such that x and y are density-reachable from z [96].

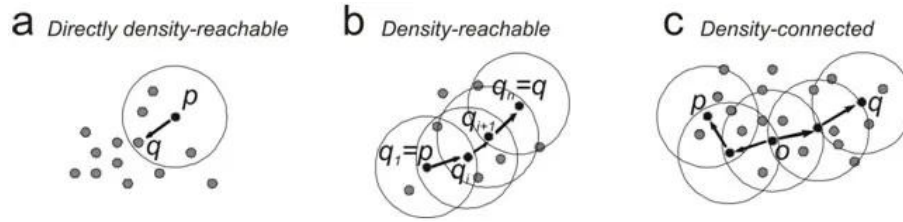


Figure G.1: Density-reachability and connectedness [97]

Cluster D is the dataset and C is a cluster. A cluster is created if the following conditions hold:

1. $\forall x, y \in D, (x \in C \wedge y \text{ is density-reachable from } x) \Rightarrow y \in C$
2. $\forall x, y \in C, x \text{ is density-connected to } y$

The points not assignable to a cluster are considered noise.

BRIDGE Bridge uses K-means to partition the data, then DBSCAN is used to find dense clusters. Afterwards, the K-means algorithm is applied again with the noise removed.

DBCLASD The Distribution-Based Clustering of Large Spatial Databases (DBCLASD) does not need any parameters (unlike the DBSCAN). It starts with an initial cluster and adds nearby points. The points in the cluster have to follow the expected distribution [98].

DENCLUE Density based clustering (DENCLUE) estimates the probability of all instances in the data space $x_t \in X \subset \mathbb{R}^d, d \in \mathbb{N}, t = 1, \dots, N$. The instances are modeled from a kernel function, for example the Gaussian kernel G.8 where $\sigma = 1$ for this example:

$$K(u, \sigma) = \frac{1}{(\sqrt{2\pi}\sigma)^d} e^{-\frac{u^2}{2\sigma^2}} \quad (\text{G.8})$$

To get the estimate of the probability for any point in the dataset, all kernels are added together. The h in Equation G.9 smoothes the kernel density. Figure G.2 shows the kernels with two different parameters for h .

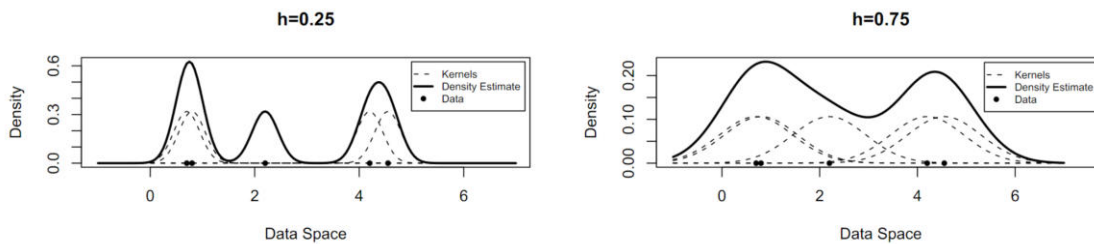


Figure G.2: Kernel density [99]

A cluster is defined by the local maxima of the kernels. A hill-climbing procedure is used to figure out the local maxima, which is guided by the gradient. In case of $\hat{p}(x)$, the equation would look like G.10.

$$\nabla \hat{p}(x) = \frac{1}{h^{d+2}N} \sum_{t=1}^N K\left(\frac{x-x_t}{h}\right) \cdot (x_t - x) \quad (\text{G.10})$$

The update step of the hill-climbing procedure to go from x^l to x^{l+1} is the Equation G.11. The updates are repeated until the density is maximized.

$$x^{l+1} = x^l + \delta \frac{\nabla \hat{p}(x^l)}{\|\nabla \hat{p}(x^l)\|_2} \quad (\text{G.11})$$

δ is used as the step size. ξ can be used to eliminate noise, as shown in Figure G.3. The data points below the threshold are considered noise [99].

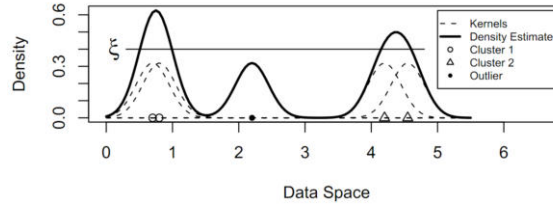


Figure G.3: Noise threshold ξ [99]

Validating a Cluster

Cluster validation is required because a clustering algorithm can introduce a bias. Three approaches exist to validate a cluster's external criteria, internal criteria, and relative criteria. External and internal criteria are statistical approaches, while the relative criteria compares different clustering schemas. The following two criteria allow the validation of a cluster:

- **Compactness:** A cluster should be densely packed, which can be measured by the variance G.12, where x_i is each data point, \bar{x} is the mean of all data points and n is the number of all data points in a cluster:

$$\sigma^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1} \quad (\text{G.12})$$

- **Separation:** The distance between clusters should be as big as possible. The distance can be measured between the cluster centers, the closest points of both clusters, and the most distant members of both clusters.

Dunn

The Dunn index assumes that clusters should have a large distance between each other, while the diameter of each cluster should be small. The function $d(\cdot, \cdot)$ is the distance function of two points, c_i is a cluster in the dataset and n_c is the number of clusters. For the clusters to be optimal, D should be large.

$$D = \min_{i=1 \dots n_c} \left\{ \min_{j=i+1 \dots n_c} \left(\frac{d(c_i, c_j)}{\max_{k=1 \dots n_c} (\text{diam}(c_k))} \right) \right\}, \text{ where} \quad (\text{G.13})$$

$$d(c_i, c_j) = \min_{x \in c_i, y \in c_j} d(x, y) \text{ and } \text{diam}(c_i) = \max_{x, y \in c_i} d(x, y)$$

Davies Bouldin Index

The Davies Bouldin Index measures the similarities of clusters. The clusters should be different to each other, therefore, the value DB should be small. The similarity measure R_{ij} can be defined freely, but has to meet the following conditions (s_i is the dispersion measure and d_{ij} is the dissimilarity measure) [100]:

- $R_{ij} \geq 0$
- $R_{ij} = R_{ji}$
- if $s_i = 0$ and $s_j = 0$ then $R_{ij} = 0$
- if $s_j > s_k$ and $d_{ij} = d_{ik}$ then $R_{ij} > R_{ik}$
- if $s_j = s_k$ and $d_{ij} < d_{ik}$ then $R_{ij} > R_{ik}$

$$DB = \frac{1}{n_c} \sum_{i=0}^{n_c} R_i, \text{ where} \quad (G.14)$$

$$R_i = \max_{j=i \dots n_c, i \neq j} R_{ij}, i = 1 \dots n_c$$

Silhouette Coefficient

The silhouette coefficient measures the distance between clusters and whether they are different from each other. The output ranges from -1 to 1. 1 means that the clusters are well apart and distinguishable, 0 means that the clusters do not have enough space between each other, while -1 means that the clusters are assigned the wrong way [101].

$$SilhouetteScore = \frac{b - a}{\max(a, b)}, \text{ where} \quad (G.15)$$

a = average distance between each data point within a cluster

b = average distance between all clusters

G.3 Predictive Analysis

G.3.1 Goal

With labeled data, machine learning models can be trained and applied to predict the labels on unseen data. This section describes some relevant machine learning models that could be used for the project. Having comments and their class ("troll" and "not-troll") allows to train models depending either on structured data (metadata) and unstructured data (text-content). In both cases, features must be extracted from the available data. The information in this section is mainly taken from the book "An Introduction to Statistical Learning" from Gareth James et al. [66].

G.3.2 Machine Learning Models

The primary goal of this study is to predict whether a user corresponds to a troll user or a regular user. Thus, the output of this prediction is a qualitative variable containing the values true or false. This inherently makes the problem at hand a *classification* problem. Therefore, only *classification* schemes are discussed in this subsection and no *regression* schemes. The schemes presented are ordered ascendingly by their flexibility. The more flexible, the better the algorithm can reflect the true underlying distribution of the data. However, a flexible model is more sensitive to changes in the dataset because there are more *degrees of freedom*. In the world of machine learning, this problem is known as the bias-variance trade-off principle [102]. A model that is too flexible and trained with too little data can lead to *overfitting*, while one that is too inflexible can result in *underfitting*. The challenge is to find the optimal trade-off between these two behaviors.

The following models are differentiated as parametric and non-parametric models. For parametric models, a strong assumption about the relationship is made. Therefore, much less training data is required to get a meaningful result and only a few parameters need to be estimated. The performance can be bad if the assumption does not hold. Non-parametric approaches, on the other hand, consist of many more degrees of freedom, thus can better adapt to an unknown underlying distribution.

Parametric Models

Logistic Regression In logistic regression, the probability is modeled with the help of the *sigmoid function* (unlike in *linear regression*). As a result, the output indicates the class and how sure the model is about the class. A further advantage of logistic regression is that the *p-value* can be calculated for each parameter, which can be used to find the relevant features. The unknown coefficients given the training data are found with the maximum likelihood approach:

$$l(\beta) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'})) \quad (\text{G.16})$$

Where $p(x_i)$ is the probability of a user being a troll given the input feature x_i .

Multiple logistic regression is the logical extension to this scheme, where x_i becomes a vector containing multiple features simultaneously. A further extension is to break the linear decision boundary by adding non-linear features (e.g. polynomial features). This step is a part of the features engineering. Other non-linear extensions are discussed in paragraph Generalized Additive Models (GAMs).

Linear Discriminant Analysis (LDA) The probability for each possible class is calculated, and the most probable class is picked as the result. LDA is preferred over logistic regression if the output can have more than two classes or if the data follows a normal distribution. This approach estimates the probability density functions of the observations given a particular class, usually by assuming a Gauss curve. Using Bayes theorem, the probabilities can then be flipped to calculate the probability of class k given observation x .

$$Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{i=1}^K \pi_i f_i(x)} \quad (\text{G.17})$$

π_k stands for the prior probability of class k . $f_k(x)$ corresponds to the probability density function of class k . The denominator represents the prior probability of observation x over all classes. Normally, a simplified version of this function, called the discriminant function, is calculated.

$$\delta_k(x) = \frac{x\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log \pi_k \quad (\text{G.18})$$

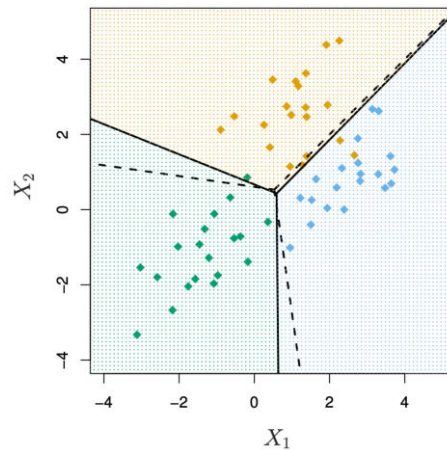


Figure G.4: Linear decision boundary of LDA
[66]

Quadratic Discriminant Analysis (QDA) *QDA* is a flexible extension to LDA. In LDA, it is assumed that the observations within each class are drawn from a Gaussian distribution with a class-specific mean μ_k and a common variance σ . For this reason, LDA always separates the classes by hyperplanes, decision boundaries that cannot bend. *QDA* allows for each class to have its own variance σ_k and thus introduces a quadratic effect on x . The decision boundaries are now hyperparabolas.

$$\delta_k(x) = -\frac{x^2}{2\sigma_k^2} + \frac{x\mu_k}{\sigma_k^2} - \frac{\mu_k^2}{2\sigma_k^2} - \frac{\log \sigma_k}{2} + \log \pi_k \quad (\text{G.19})$$

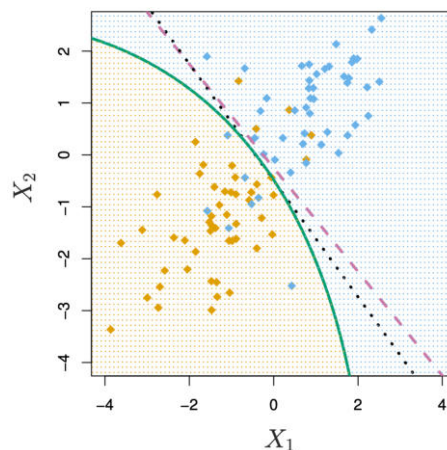


Figure G.5: Quadratic decision boundary of QDA
[66]

Naive Bayes Naive Bayes is closely related to the LDA classifier in the sense that both choose the most likely class as a result. However, LDA operates on continuous features, while Naive Bayes uses discrete features. This classifier assumes feature independency. The classifier performs best with multidimensional data where the individual features do not correlate.

Generalized Additive Models (GAMs) GAMs provide a compromise between linear and non-parametric approaches. They are a framework to extend a linear model by allowing non-linear functions for each of the variables. Examples of non-linear functions are polynomials, step-functions, natural cubic splines and smoothing splines. Polynomial functions of the feature impose a global structure and tend to wiggle strongly at the ends. Step functions, on the other hand, allow a more local approximation of the decision boundary but are limited in terms of flexibility.

Combining the local approach of step-functions and the flexibility of polynomials, a natural cubic spline results where each section corresponds to a cubic spline.

A smoothing spline is a natural cubic spline where the sections are as small as possible.

These functions can now be inserted into the *sigmoid function* of the logistic regression. The result is a highly non-linear decision boundary. One limitation of GAMs is that they assume additivity of the different predictors. That is, changing one predictor has no effect on the other predictors.

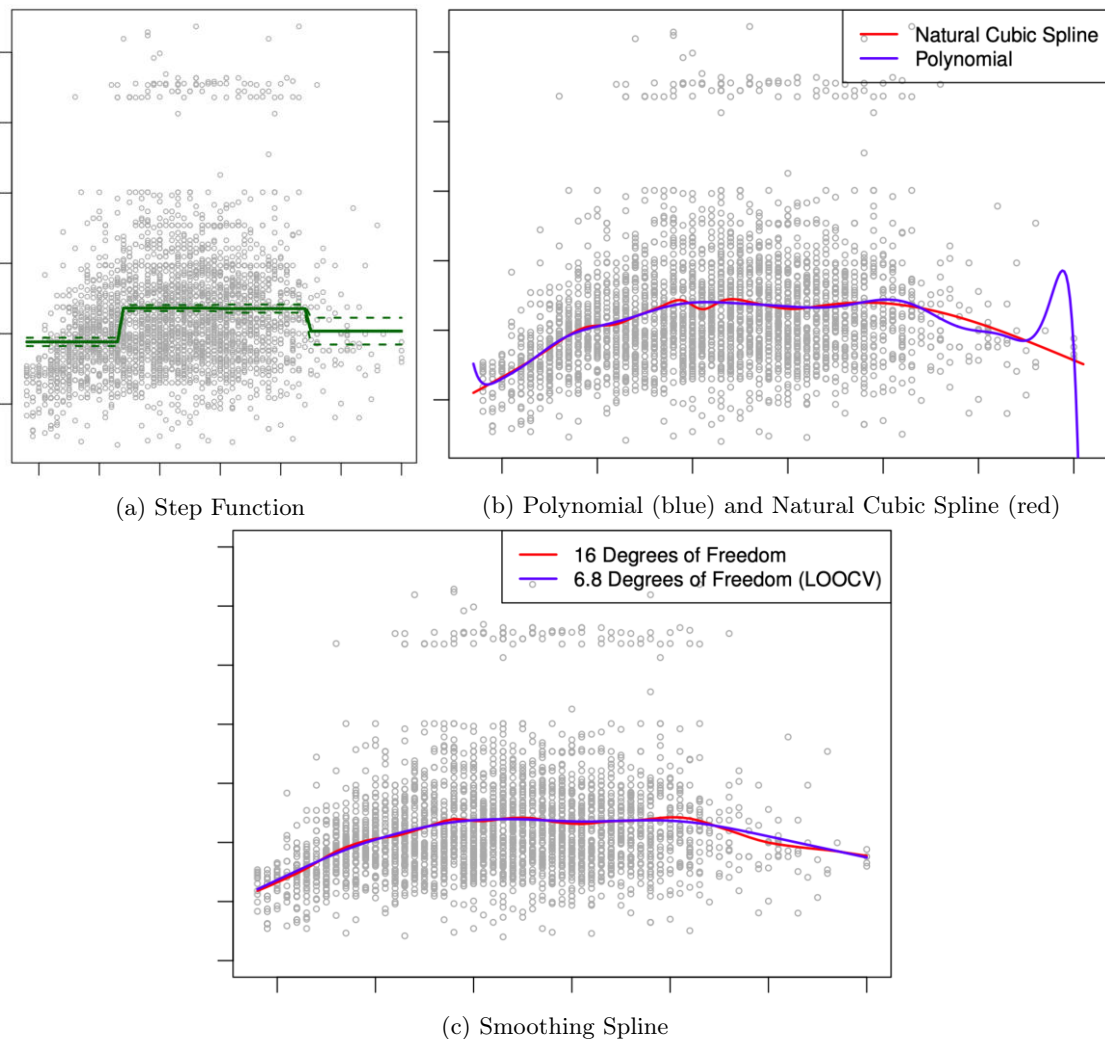


Figure G.6: Non-Linear functions
[66]

Support Vector Machines (SVM) Support Vector Machines are binary classifiers. They are an extension of Support Vector Classifiers, which separate the data with linear decision boundaries.

When a new sample needs to be classified, it can be inserted into the function seen in G.20. The sign of the result indicates the class, and the value represent how sure the classifier is about the

result. If the resulting value is near zero, there is less certainty and if the value is farther away from zero, the result is more certain.

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i) \quad (\text{G.20})$$

An advantage of this approach is that samples far away from the boundary (outside a specified margin) have no impact on the shape of the classifier. Only support vectors affect the classifier, which is why they are called "support" vector. This behavior is different from the LDA classifier, where the classifier depends on all samples.

$K(x, x_i)$ represents the kernel. A kernel is a function that quantifies the similarity of two observations (vectors). This kernel can be chosen freely. Depending on the kernel, a different decision boundary is the result. The inner product is known as a linear kernel G.21 and results in a linear decision boundary.

$$K(x_i, x_{i'}) = \sum_{j=1}^P x_{ij} x_{i'j} \quad (\text{G.21})$$

There are also polynomial kernels of degree d that add polynomial features. A popular choice is often the radial kernel G.22, which is also known as the Gaussian kernel.

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^P (x_{ij} - x_{i'j})^2) \quad (\text{G.22})$$

It should be noted that this kernel corresponds to a simplified version of the Equation G.8. These non-linear kernels enlarge the feature space. In a higher dimension, the decision boundary is still linear. More features usually imply more computational effort. The kernel trick allows going into the higher dimensional space without having to pay the computational price.

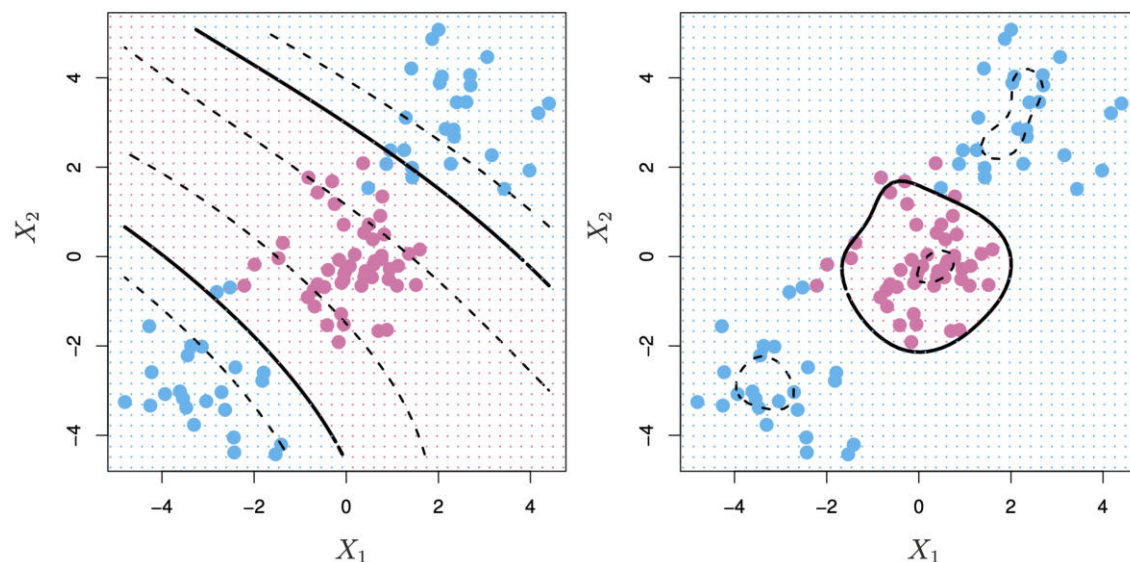


Figure G.7: Support Vector Machine with a polynomial kernel (left) and a radial kernel (right)

[66]

Non-Parametric Models

Decision Trees The idea behind this scheme is to divide the feature space into nested regions. As can be seen in Figure G.8, the data can be split multiple times in each dimension. What results

are the subregions R_i whose elements are then used to predict the class. This is also known as the stratification of the feature space.

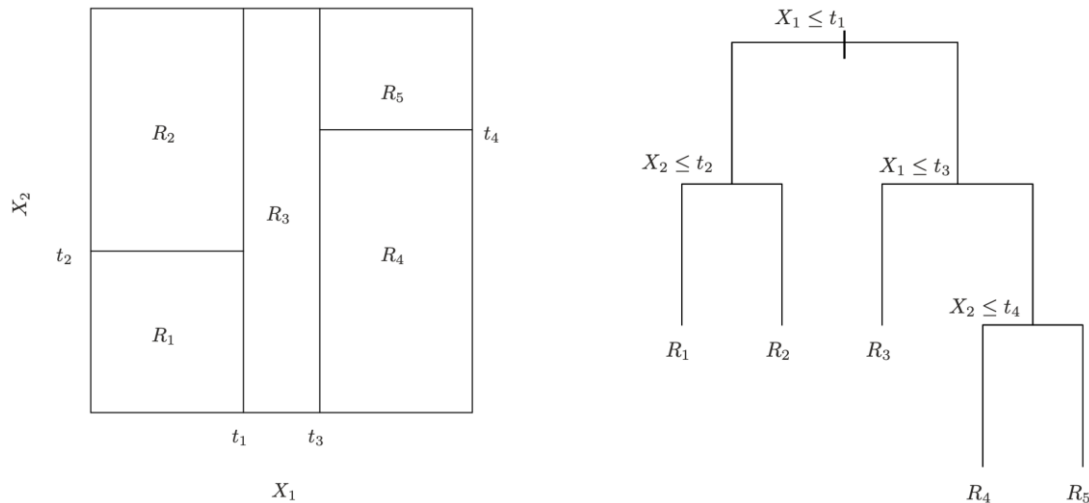


Figure G.8: Split feature space (left) and the corresponding decision tree (right)

[66]

It is called decision tree because the nested regions can be visualized as a tree. If the class of a new sample needs to be predicted, the tree can be traversed from root to leaf. A decision has to be made at each node whether to go left or right.

For example, if a decision tree aims to find out whether a comment or not is written by a troll, decisions could be:

- Does the comment have more than 20 downvotes? (Yes / No)
- Was the comment written between (1:00 am and 4:00 am)? (Yes / No)
- ...

The remaining region in the end is used to predict the final class (for example "troll" or "not-troll"). Typically, the most commonly occurring class in this region is a good predictor of the final class.

The flexibility of this scheme can be controlled by the number of samples per region. The fewer samples, the more likely this scheme is to be overtrained and vice versa.

Decision trees are straightforward to explain because they can be easily visualized. Unfortunately, decision trees are not as powerful regarding predictive accuracy as other classification approaches described in this section. This is due to the large variance of a single tree. However, this problem can be overcome by aggregating many decision trees and is further discussed in the next paragraph.

Random Forests As decision trees suffer from high variance, bagging (bootstrap aggregation) is used to reduce the variance. With bootstrapping, different training sets are generated. A model is trained on each bootstrapped training set and finally the class that occurs the most often is chosen as the final prediction. This is known as the majority vote.

A consequence of taking bootstrapped training sets is that they are correlated, thus aggregating different models does not improve variance much. Random forests use a trick that de-correlates the bagged trees, which significantly improves the performance. For every split, a random sample of predictors is chosen as split candidates. This results in more diverse trees. Therefore, the majority vote becomes more meaningful.

K-nearest Neighbor (KNN) The *KNN* algorithm tries to find the K nearest neighbors N_0 and then estimates the probabilities of class j containing the given test point x_0 .

$$Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j) \quad (\text{G.23})$$

Class j with the highest estimated probability is then selected as the winning class. This principle closely follows the Bayes classifier approach. K controls the trade-off between variance and bias. A too large value for K implies a small variance but a large bias because the classifier cannot represent the true underlying distribution $f()$ well.

G.3.3 Optimization of Machine Learning Models

Regularization

For the model to better generalize on unseen data, the model can be regularized. This can be necessary, when there is relatively little training data compared to the number of predictors. The idea is to force the coefficients of the model to be closer to zero, which leads to a smaller variance. Popular methods to achieve this are Ridge and Lasso regularizations.

Feature Selection

As described in Section 4.6.3 and E.1, there are many possible (metadata) features to identify trolls and troll comments. Having more features requires having more training data as well (curse of dimensionality), as the model will otherwise be overtrained. Since the available amount of training data is limited, measures must be taken to only pick the most essential features. Another effect is that the model becomes easier to interpret with fewer features.

It is important to note that not too many features should be included despite all the feature selection methods. Otherwise, there will be too much noise, and the feature selection methods will not be able to only select the relevant ones.

Best Subset Selection To find the best subset of features, simply all possible combinations are tested and the set of features with the best measure (see G.3.5) is picked. This conceptually simple algorithm always finds the best set of features. The major flaw though is that 2^p models need to be fitted, where p is the number of predictors. Hence, taking the approximately 40 features for identifying trolls this results in 2^{40} models. This is more than a trillion model fits.

Forward Stepwise Selection This algorithm is much more efficient than best subset selection, but doesn't guarantee to find the optimal solution. It starts with a model that contains no variables, and then proceeds by adding the most significant predictors one after the other until there are no predictors left. This results in a total of $p * (p + 1)/2$ models. With the example of 40 features, this results in only $40 * (40 + 1)/2 = 820$ models to fit.

Selection with Lasso norm Lasso is primarily a regularization method. With Lasso, some coefficients are forced to be almost zero. This property therefore leads to variable selection too.

Dimension Reduction

Dimension Reduction is not actually a feature selection method, since each of the new predictors is a linear combination of the old features. Nevertheless, the number of predictors is reduced, so this measure also helps against overfitting. These dimension reduction techniques follow the idea that variance captures information. Hence, the direction containing the most variance is searched in the feature space. This direction becomes the first principal component and thus the first dimension in the new feature space. The direction containing the second most variance and is orthogonal to the first principal component becomes the second principal component, and hence the second dimension. This can be repeated until the desired number of dimensions is reached.

Ensemble Learning

The intuition behind this technique is that training many weak classifiers and combining their results is often better than one single classifier. Ensemble methods are usually used towards the end of a project when a few good predictors are already built. The prerequisite is that the predictors are diverse (e.g., different algorithms, hyperparameters, or data), and there is a sufficient number of weak learners.

G.3.4 Prediction or Inference

The goal of a machine learning model is either "prediction" or "inference". In prediction, the concern is only on the quality of the final estimate. In this project, this would be whether a user is classified as a troll or not. How the function is constructed is not relevant, as long as the estimates are accurate. In inference, the goal is to understand the relationship between the input variables and the resulting output variable. Thus, inference requires models which can be easily interpretable. For this study, both problems are of interest and deserve further investigation.

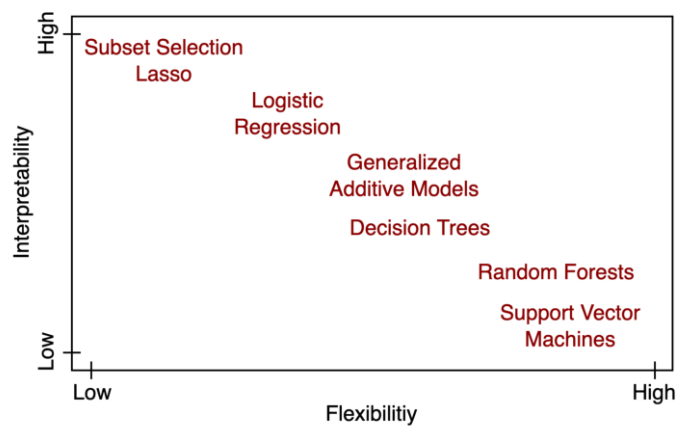


Figure G.9: Flexibility vs. Interpretability

G.3.5 Validation of the Models

After the model is built and predictions can be performed, one of the fundamental questions becomes how the model's performance can be evaluated. Since this is a common problem in machine learning, there are already proven solutions. Unlike *unsupervised* problems, *supervised* problems have access to labeled datasets. Hence, meaningful validation can be achieved more easily.

It is crucial to split the dataset in a first step into training data and test data [66]. 80/20 is a commonly used ratio, where 80% of the data is used for training and 20% is used for test purposes. If the training data is used to evaluate the model's performance, it will not be clear whether the noise is followed. Thus, model *overfitting* would not be recognized.

Cross-Validation

Overview Cross-validation is a resampling procedure and provides information about how well the classifier generalizes to data it was not trained on. It is helpful to evaluate machine learning models on a limited data sample [103]. Since the amount of labeled data in this study is limited, cross-validation will be relevant. Another use case is to find the optimal *hyperparameter* with cross-validation. For example, cross-validation can be used in logistic regression with polynomials to find the optimal flexibility setting. Since this study deals with a classification problem, the *Error* rate can be used as a performance measure (see Measures for Model Evaluation for more information). Three variations of cross-validation are discussed below.

Validation Set This is the most basic approach where the observations are randomly split into a training and a validation set. The model is trained using the training set and then tested using

the validation set. This approach is fast because only one model needs to be trained. However, it has a large bias estimating the true test *Error* rate because only a small number of observations is used for training. It also has a high variance since it is a one-shot method and no averages are performed.

Leave-one-out Cross-Validation (LOOCV) Again, the observations are split into two parts. But the validation set only consists of one observation. Thus, the model can be fitted to almost all data ($n - 1$). The single sample is used to test the model. This procedure is systematically repeated n times until each sample was once used for testing. In the end, the results are averaged and the *Error* rate is calculated. This approach is computationally the most expensive, since the model has to be fitted n times. LOOCV has a low bias for the final estimate because only one sample is not used in the creation of the model.

k -fold Cross-Validation The basic idea with this approach is to split the observations randomly in k equally sized sets. Each set is once left out while the other sets are used to fit the model. The set which was left-out is used to calculate the particular *Error* rate. After k model fittings, the k validation *Error* rates are averaged to an estimate of the test *Error* rate. LOOCV is a special case of k -fold cross-validation, where $k = n$. k -fold cross-validation has a medium bias because $100\%/k$ of the data is not used for fitting the model.

Measures for Model Evaluation

Considering the relatively small number of trolls, a "dumb" classifier that would always predict the no-troll class would still achieve high accuracy. Since the two classes, troll and no-troll, are not the same size, accuracy is not a meaningful measure to validate the model.

For the binary classifier required for this project, there can be two types of errors. Labeling a regular user as troll (False Positive), and labeling a troll as a regular user (False Negative). This types of errors can be illustrated with the following *confusion matrix* [66]:

Table G.1: Sample confusion matrix

		<i>Predicted Trolls</i>		
		Yes	No	Total
<i>Actual Trolls</i>	Yes	TP	FN	P
	No	FP	TN	N
	Total	P*	N*	total

With these resulting characteristic values, various performance measures can be calculated. Some are listed below:

$$Accuracy = \frac{TP + TN}{total}$$

$$Error = \frac{FP + FN}{total}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Sensitivity = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

Not all of these values can be optimized at the same time and depends on the business goal. Regarding troll detection, missing an actual troll might hurt the newspaper more than punishing an innocent user, especially if automatically detected trolls go through an additional manual review. A newspaper thus might want to maximize the *Sensitivity* measure, since the false negatives (*FN*) need to remain small. However, if the tool will be publicly available, it should be avoided to falsely accuse legitimate users as trolls and therefore the false positive rate should remain small. By adjusting the threshold value of the decision boundary, these measures can be optimized.

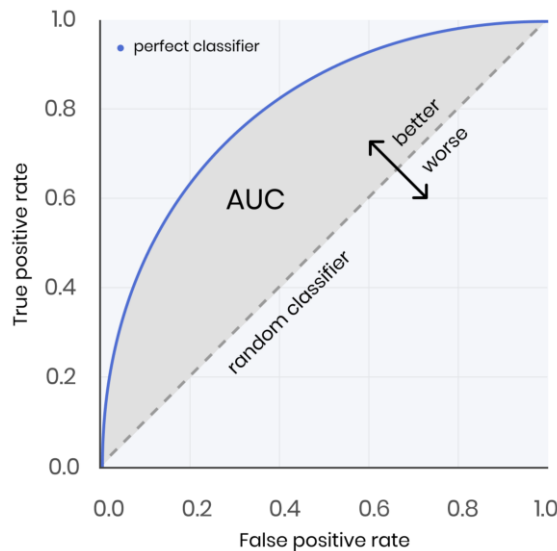


Figure G.10: Sample ROC curve
[104]

A general way to compare different classification schemes without having a fixed threshold is to use the *ROC* curve as shown in Figure G.10. An optimal classification scheme hugs the upper-left corner. Hence, the area under the curve (*AUC*) is a good measure for the fundamental performance of a binary classification scheme [66].

G.4 Take away for the project

The main idea in this study is to identify trolls based on the available metadata and content of the comments. There exists some limited amount of training data for this domain. Therefore, a mix of unsupervised and supervised algorithms should be used. Either way, the exploratory data analysis described in section G.2 is suitable to find meaningful patterns in the data and to get a better understanding of the data.

It is easier to work with a labeled troll dataset. A classifier based on a labeled troll dataset could be trained with the machine learning schemes described in section G.3.2.

G.5 Dimension Reduction Methods

Dimension reduction is used to reduce the number of input variables into a machine learning model. Dimension reduction techniques capture the essence of the data set and help to visualize multi-dimensional data.

G.5.1 PCA

Principal component analysis shifts the axis of the data-set so that the first principal component, which can be seen as an axis, gets shifted to capture the most variance. The second captures the second most and so on.

An in-depth review was written by Ian T. Jolliffe and Jorge Cadima [105].

G.5.2 t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) projects the data into the desired dimensions. The projection is initially done randomly and then moved to the correct position iteratively. In a first step, the similarities of the points to each other are calculated. The calculated values are then plotted as a normal distribution and the values are scaled to one to accommodate clusters that are further apart and have a higher standard deviation. The data points get randomly projected to the desired projection and are then moved to the correct place. This is done by calculating the similarities, which are now plotted in a t-distribution to avoid the points getting clustered together. The resulting similarity matrix is then used to iteratively move the points to match the similarity matrix before the reduction [106].

G.5.3 UMAP

Uniform Manifold Approximation and Projection (UMAP) is very similar to t-SNE. The differences are that t-SNE initializes the reduction randomly and moves every point around while UMAP uses spectral embedding meaning that the reductions are initialized the same way each time the function is run. Another advantage for UMAP is that it randomly chooses two points to move together, which is more performant on a large data set [107].

G.6 Natural Language Processing

G.6.1 BERT

BERT stands for Bidirectional Encoder Representations from Transformers, as the name implies it uses bidirectional transformers to improve the encoding. The attention mechanism used in OpenAI GPT is very similar. The difference being that in OpenAI GPT, the transformers are not connected to the previous one.

Embeddings The input for the pre-training is embedded in three layers. The input is an array of tokens starting with a [CLS] token, which is an aggregate sequence representation. The sentences are split with a [SEP] token.

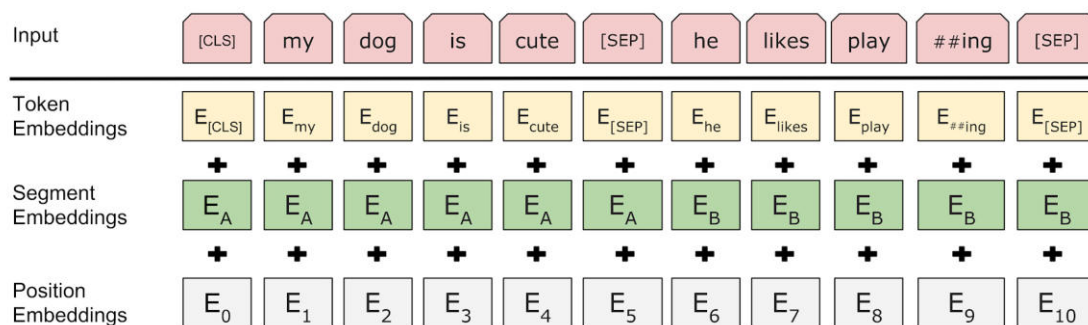


Figure G.11: Multi-Head attention (2)
[108]

The token embedding encodes the input into a vector. For the *BERT_{BASE}* and *BERT_{LARGE}*, WordPiece [109] was used. Segment embeddings are added to indicate to which sentence the token belongs. The last position embedding is used to keep positional information.

Pre-training The pre-training of *BERTs* is done with two tasks. The first task is to predict a masked word in a sentence, and the second one is to label whether the sentences come after one another. The examples used in the paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" are as follows:

Input = [CLS] the man went to [MASK] store [SEP]
he bought a gallon [MASK] milk [SEP]
Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]
penguin [MASK] are flight ##less birds [SEP]
Label = NotNext

The mask tokens are randomly chosen and include 15% of all tokens. This would result in a mismatched pre-training and fine-tuning set, as the mask token do not appear in the fine-tuning set. To prevent this, the chosen tokens have a probability to be replaced by a random token 10% of the time. Another 10% of the time, the original token will be used, and the remaining ones are masked.

Fine-tuning Fine-tuning is used to train the *BERT* for a specific task. The attention mechanism used in *BERT* allows swapping the input and output to the training-set. For the case of a single sentence classification, the model looks like figure G.12.

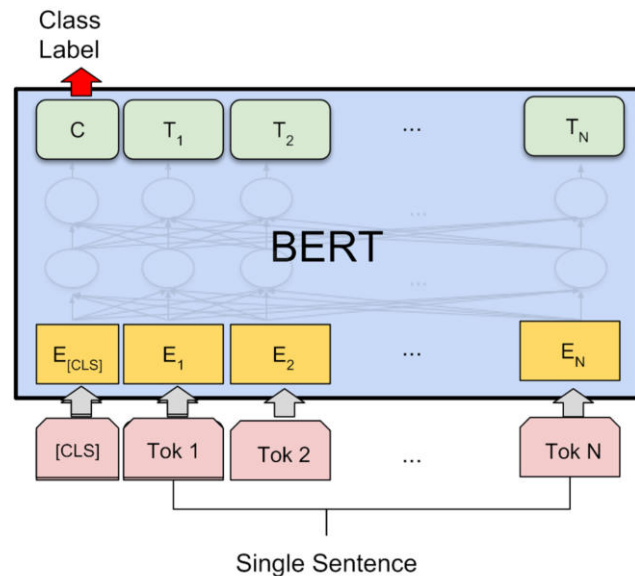


Figure G.12: Single sentence classification
[108]

Visualization To visualize the BERT embeddings, 40 sentences were used and labeled whether the sentence is food-related or not.

Food sentences

1. I am hungry.
2. I love cake.
3. I ordered a burger and a salad.
4. This cheese has an awful smell.
5. My brother prefers vegies to meat.
6. He always ate fruit for dessert.
7. We will eat a pizza for dinner.
8. I often have oatmeal for breakfast.
9. Crunchy carrot sticks or unsalted nuts make a tasty alternative to crisps.
10. Lisa ate the food and washed the dishes.
11. Help, I don't want these too spicy!.
12. The chicken is rather salty so you probably won't need salt.
13. This might need some salt and pepper.
14. She licked the chocolate off her fingers.
15. The only time my family gets to eat turkey is on special occasions.
16. Katie picked at her food.
17. I bought salami and cheese from the deli.
18. I haven't eaten real food in weeks.
19. She peeled off the skin of a banana for the child.
20. The rice and beans were bland until mixed together.

Weather sentences

1. The weather is nice.
2. The sun was warm.
3. A blowing snow storm delayed our flight north.
4. The rain seems less heavy.
5. Suddenly a storm came up.
6. By noon the snow was all gone.
7. Sunday dawned a sunny hot July day.
8. I had hail damage to my car as well.
9. The day was windy with a little light rain at times and poor visibility.
10. A cool breeze touched her cheeks and neck.
11. The tornado left a trail of destruction in its wake.
12. Fog coated the ocean, and a cold, moist wind made her eyes water.
13. Every morning it was so cool and misty that i got goosebumps.
14. The climate is damp, hot and malarious.
15. The grass is often damp in the morning.
16. It is very humid and warm in the jungle.
17. It rains cats and dogs.
18. The weather will be hot and dry.
19. It is cloudy today.
20. It's probably gonna rain tomorrow'

Embedding The sentences were embedded using $BERT_{BASE}$ in the uncased variant, meaning that there is no differentiation between upper and lower case. The model, which is hosted on HuggingFace was introduced in the same paper as BERT [108].

To visualize the resulting vector, a reduction to two dimensions with PCA was performed. The resulting plot is visible in Figure G.13.

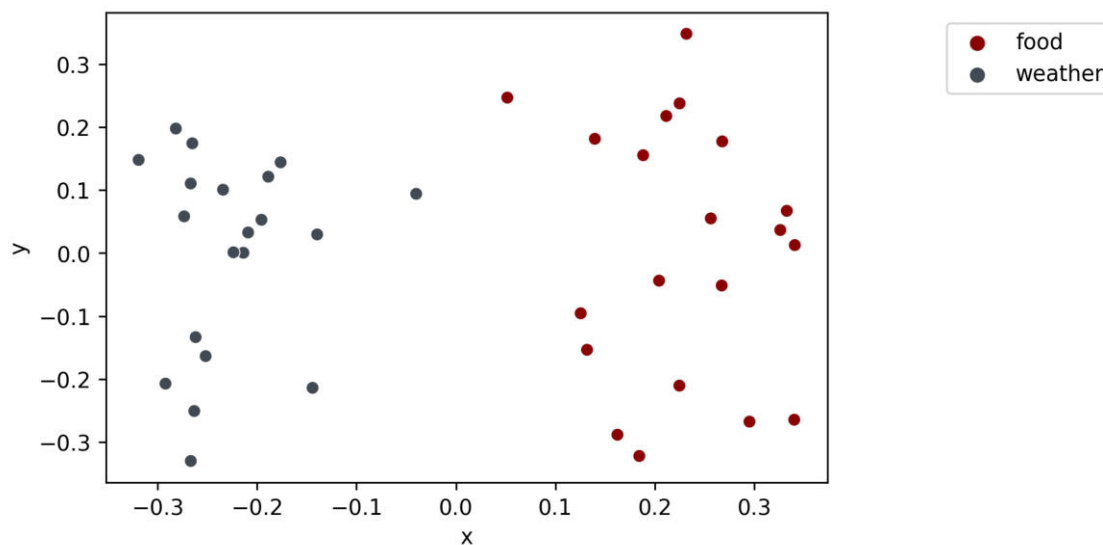


Figure G.13: BERT visualization using PCA

The explained variance is 22.11% after the reduction. Figure G.13 shows that the two labels are separable, but the single linkage distance is small.

Using UMAP to reduce the vectors to two dimensions resulted in the plot in Figure G.14.

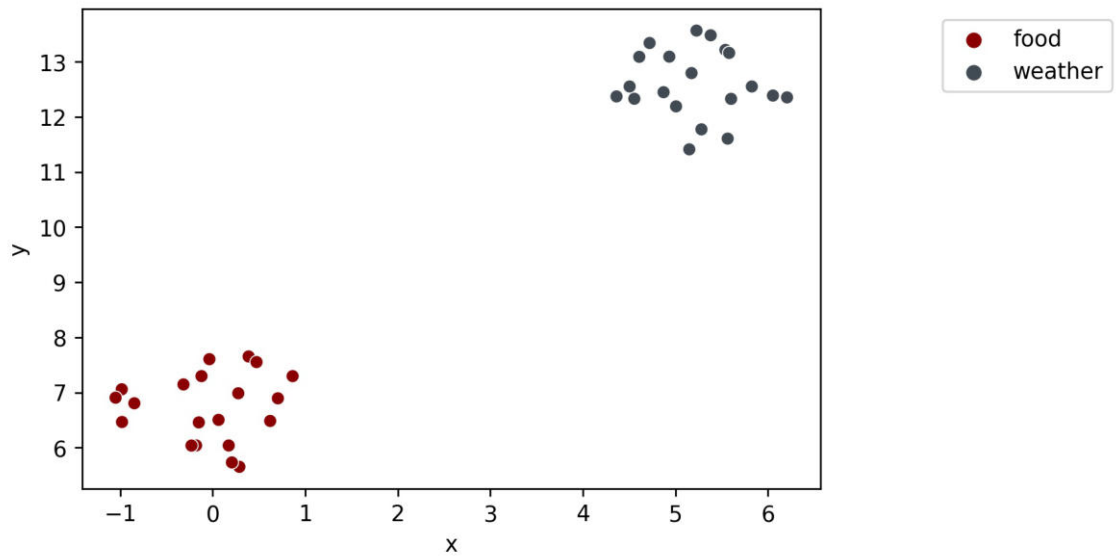


Figure G.14: BERT visualization using UMAP

The vectors are clearly separable and shows that the BERT embedding can cluster similar meanings in the sentences.