---

title: "Text Mining"

author: "Sabrina Michl"

date: "2023-01-27"

categories: [code, analysis]

bibliography: ref.bib

image: "image.jpg"

---

# 1. Preliminary Note

For this analysis we use the dataset from @data/0B5VML\_2019 out of the zip archive @data/0B5VML/XIUWJ7\_2019. The data are licensed according to Attribution 4.0 International (CC-BY-4.0). The used wordembeddings are from @grave2018learning. The data are licensed according to Attribution-ShareAlike 3.0 Unported (CC-BY-SA 3.0). The picture, that is used is from @bildquelle.

# 2. Load The Packages

```{r output=FALSE}

library(tidyverse)

library(rio)

library(tidymodels)

library(tidytext)

library(textrecipes)

library(lsa)

library(discrim)

library(naivebayes)

library(tictoc)

library(fastrtext)

library(remoji)

library(tokenizers)

```

# 3. Load Dataset And Minor Changes

## 3.1 Train Dataset

```{r output=FALSE}

d\_train <- read\_tsv("C:/Users/sapi-/OneDrive/Studium/5. Semester/Data Science II/Data\_Science\_Blog/daten/germeval2018.training.txt", col\_names = FALSE)

```

### Rename Columns

```{r}

names(d\_train) <- c("text", "c1", "c2")

```

### Add ID Column

```{r}

d\_train <- d\_train %>%

mutate(id = row\_number()) %>%

select(id, everything())

```

## 3.2 Test Dataset

```{r output=FALSE}

d\_test <- read\_tsv("C:/Users/sapi-/OneDrive/Studium/5. Semester/Data Science II/Data\_Science\_Blog/daten/germeval2018.test.txt", col\_names = FALSE)

```

### Rename Columns

```{r}

names(d\_test) <- c("text", "c1", "c2")

```

### Add ID Column

```{r}

d\_test <- d\_test %>%

mutate(id = row\_number()) %>%

select(id, everything())

```

# 4. Explore Dataset

```{r}

train\_toc <- d\_train %>%

unnest\_tokens(output = token, input = text)

train\_toc

```

> First we tokenize the dataset d\_train. \## Insert `Stopwords\_de`

```{r}

data(stopwords\_de, package = "lsa")

stopwords\_de <- tibble(word = stopwords\_de)

stopwords\_de <- stopwords\_de %>%

rename(token = word)

```

> After that we use the stopwords\_de to `anti\_join` this with train\_toc.

```{r}

train\_toc2 <- train\_toc %>%

anti\_join(stopwords\_de)

```

## Show The Important Words

```{r}

train\_toc2 <- train\_toc2 %>%

count(token, sort = TRUE)

```

### Plot

```{r}

train\_toc2 %>%

slice\_head(n=20) %>%

ggplot()+

aes(y=reorder(factor(token), n), x = n, color = token)+

geom\_col(aes(fill = token, alpha = 2.5)) +

ggtitle("The most used words") +

ylab("token")+

xlab("quantity")+

theme\_minimal()+

theme(legend.position = "none")

```

> We see that, to most used word is "lbr". We could inspect the dataset way deeper, e. g. do a manual sentimentanalysis or do a lemmatization or stem the words. But we will have a look to these types in the different machine learning algorithm now.

# 5. Preparation

## 5.1 Define Recipe - rec1 - TF-IDF

```{r}

rec1 <-

recipe(c1 ~ ., data = select(d\_train, text, c1, id)) %>%

update\_role(id, new\_role = "id") %>%

step\_tokenize(text) %>%

step\_stopwords(text, language = "de", stopword\_source = "snowball") %>%

step\_stem(text) %>%

step\_tfidf(text) %>%

step\_normalize(all\_numeric\_predictors())

rec1

```

### Prep & Bake - rec1

```{r}

rec1\_prep <- rec1 %>%

prep() %>%

recipes::bake(new\_data = NULL)

```

## 5.2 Define Recipe - rec2 - word embedding

After fitting all the models I have decided to not fit the easy word embedding recipe (rec2)! The reason for this decision is, because of the analysis run-time! We have a large amount of data and extensive recipes, so when I raise the resamples for the prediction, R Studio quits the session.

- training: v = 1, folds = 2

- prediction: v = 2, folds = 5

```{r}

#rec2 <-

#recipe(c1 ~ ., data = select(d\_train, text, c1, id)) %>%

#update\_role(id, new\_role = "id") %>%

#step\_tokenize(text) %>%

#step\_stopwords(text, language = "de", stopword\_source = "snowball") %>%

#step\_word\_embeddings(text, embeddings = word\_embedding\_text)

```

### Insert The Predefined List

```{r}

out\_file\_model <- "C:/Users/sapi-/OneDrive - Hochschule für Angewandte Wissenschaften Ansbach/Desktop/AWM/angewandte Wirtschats- und Medienpsychologie/5. Semester/Word\_Embedding/de.300.bin"

```

```{r}

file.exists(out\_file\_model)

```

```{r}

fasttext\_model <- load\_model(out\_file\_model)

dictionary <- get\_dictionary(fasttext\_model)

get\_word\_vectors(fasttext\_model, c("menschen")) %>% `[`(1:10)

```

```{r}

print(head(dictionary, 10))

```

```{r}

word\_embedding\_text <- tibble(word = dictionary)

```

```{r}

options(mc.cores = parallel::detectCores())

words\_vecs <- get\_word\_vectors(fasttext\_model)

```

```{r output=FALSE}

word\_embedding\_text <-

word\_embedding\_text %>%

bind\_cols(words\_vecs)

```

```{r}

names(word\_embedding\_text) <- c("word", paste0("v", sprintf("%03d", 1:301)))

```

## 5.3 Define Recipe - rec3 - Word Embeddings

### Insert the Helperfunctions

We are using the package \[pradadata\] (https://github.com/sebastiansauer/pradadata) from @sebastian\_sauer\_2018\_1996614. The data are licensed according to General Public License 3 (GLP-3).

```{r}

data("schimpwoerter", package = "pradadata")

data("sentiws", package = "pradadata")

data("wild\_emojis", package = "pradadata")

source("C:/Users/sapi-/OneDrive/Studium/5. Semester/Data Science II/Data\_Science\_Blog/helper/helper\_funs.R")

```

### rec3

```{r}

rec3 <-

recipe(c1 ~., data = select(d\_train, text, c1, id)) %>%

update\_role(id, new\_role = "id") %>%

step\_text\_normalization(text) %>%

step\_mutate(emo\_count = map\_int(text, ~count\_lexicon(.x, sentiws$word))) %>%

step\_mutate(schimpf\_count = map\_int(text, ~count\_lexicon(.x, schimpfwoerter$word))) %>%

step\_mutate(wild\_emojis = map\_int(text, ~count\_lexicon(.x, wild\_emojis$emoji))) %>%

step\_mutate(text\_copy = text) %>%

step\_textfeature(text\_copy) %>%

step\_tokenize(text) %>%

step\_stopwords(text, language = "de", stopword\_source = "snowball") %>%

step\_stem(text) %>%

step\_word\_embeddings(text, embeddings = word\_embedding\_text)

```

```{r}

rec3\_prep <- rec3 %>%

prep() %>%

recipes::bake(new\_data = NULL)

```

## 5.4 Define Recipe - rec4 - TF-IDF

### rec4

```{r}

rec4 <-

recipe(c1 ~., data = select(d\_train, text, c1, id)) %>%

update\_role(id, new\_role = "id") %>%

step\_text\_normalization(text) %>%

step\_mutate(emo\_count = map\_int(text, ~count\_lexicon(.x, sentiws$word))) %>%

step\_mutate(schimpf\_count = map\_int(text, ~count\_lexicon(.x, schimpfwoerter$word))) %>%

step\_mutate(wild\_emojis = map\_int(text, ~count\_lexicon(.x, wild\_emojis$emoji))) %>%

step\_mutate(text\_copy = text) %>%

step\_textfeature(text\_copy) %>%

step\_tokenize(text) %>%

step\_stopwords(text, language = "de", stopword\_source = "snowball") %>%

step\_stem(text) %>%

step\_tfidf(text)

```

```{r}

rec4\_prep <- rec4 %>%

prep() %>%

recipes::bake(new\_data = NULL)

```

# 6. Build Resamples

I have decided to go with the V-Fold-Cross-Validation, because of the time. We have a large amount of data with extremely extensive recipes, so the run-time would be enormous, if we would try another option of resampling, e. g. bootstrapping.

```{r}

folds <- vfold\_cv(data = d\_train,

v = 2,

repeats = 1,

strata = c1)

```

# 7. Build the Penalty-Grid

```{r}

lambda\_grid <- grid\_regular(penalty(), levels = 30)

```

# 8. Build the Models

## 8.1 Null Model

```{r}

mod0 <- null\_model() %>%

set\_engine("parsnip") %>%

set\_mode("classification")

```

### Define The Workflow

```{r}

wf0 <- workflow() %>%

add\_recipe(rec1) %>%

add\_model(mod0)

```

### Resampling & Model Quality

```{r}

options(mc.cores = parallel::detectCores())

tic()

fit0 <- fit\_resamples(

wf0,

folds,

control =control\_resamples(save\_pred = TRUE)

)

toc()

```

```{r}

performance0 <- collect\_metrics(fit0)

performance0

```

```{r}

preds0 <- collect\_predictions(fit0)

preds0 %>%

group\_by(id) %>%

roc\_curve(truth = c1, .pred\_OFFENSE) %>%

autoplot()

```

```{r}

conf\_mat\_resampled(fit0, tidy = FALSE) %>%

autoplot(type = "heatmap")

```

## 8.2 Lasso-L1 With TF-IDF

### L1-Model

```{r}

l1\_82\_mod <- logistic\_reg(penalty = tune(), mixture = 1) %>%

set\_engine("glmnet") %>%

set\_mode("classification")

l1\_82\_mod

```

### Define The Workflow

```{r}

l1\_82\_wf <-workflow() %>%

add\_recipe(rec1) %>%

add\_model(l1\_82\_mod)

l1\_82\_wf

```

### Resampling & Model Quality

```{r}

options(mc.cores = parallel::detectCores())

l1\_82\_wf\_fit <- tune\_grid(

l1\_82\_wf,

folds,

grid = lambda\_grid,

control = control\_resamples(save\_pred = TRUE)

)

```

### Model Performance

```{r}

l1\_82\_wf\_fit\_performance <- collect\_metrics(l1\_82\_wf\_fit)

l1\_82\_wf\_fit\_performance

```

```{r}

l1\_82\_wf\_fit\_preds <- collect\_predictions(l1\_82\_wf\_fit)

```

```{r}

l1\_82\_wf\_fit\_preds %>%

group\_by(id) %>%

roc\_curve(truth = c1, .pred\_OFFENSE) %>%

autoplot()

```

```{r}

autoplot(l1\_82\_wf\_fit)

```

### Select The Best

```{r}

chosen\_auc\_l1\_82\_wf\_fit <-

l1\_82\_wf\_fit %>%

select\_by\_one\_std\_err(metric = "roc\_auc", -penalty)

chosen\_auc\_l1\_82\_wf\_fit

```

## 8.3 Ridge-Regression-L2 With TF-IDF

### L2-Model

```{r}

l2\_83\_mod <- logistic\_reg(penalty = tune(), mixture = 0) %>%

set\_engine("glmnet") %>%

set\_mode("classification")

l2\_83\_mod

```

### Define The Workflow

```{r}

l2\_83\_wf <-workflow() %>%

add\_recipe(rec1) %>%

add\_model(l2\_83\_mod)

l2\_83\_wf

```

### Resampling & Model Quality

```{r}

options(mc.cores = parallel::detectCores())

l2\_83\_wf\_fit <- tune\_grid(

l2\_83\_wf,

folds,

grid = lambda\_grid,

control = control\_resamples(save\_pred = TRUE)

)

```

```{r}

collect\_metrics(l2\_83\_wf\_fit)

```

### Model Performance

```{r}

l1\_83\_wf\_fit\_performance <- collect\_metrics(l1\_83\_wf\_fit)

l1\_83\_wf\_fit\_performance

```

```{r}

l1\_83\_wf\_fit\_preds <- collect\_predictions(l1\_83\_wf\_fit)

```

```{r}

l1\_83\_wf\_fit\_preds %>%

group\_by(id) %>%

roc\_curve(truth = c1, .pred\_OFFENSE) %>%

autoplot()

```

### Select The Best

```{r}

chosen\_auc\_l2\_83\_wf\_fit <-

l2\_83\_wf\_fit %>%

select\_by\_one\_std\_err(metric = "roc\_auc", -penalty)

chosen\_auc\_l2\_83\_wf\_fit

```

## 8.4 Lasso-L1 With Word Embeddings

### L1-Model

```{r}

l1\_84\_mod <- logistic\_reg(penalty = tune(), mixture = 1) %>%

set\_engine("glmnet") %>%

set\_mode("classification")

l1\_84\_mod

```

### Define The Workflow

```{r}

l1\_84\_wf <- workflow() %>%

add\_recipe(rec3) %>%

add\_model(l1\_84\_mod)

l1\_84\_wf

```

### Resampling & Model Quality

```{r}

options(mc.cores = parallel::detectCores())

l1\_84\_wf\_fit <- tune\_grid(

l1\_84\_wf,

folds,

grid = lambda\_grid,

control = control\_resamples(save\_pred = TRUE)

)

```

```{r}

collect\_metrics(l1\_84\_wf\_fit)

```

### Model Performance

```{r}

l1\_84\_wf\_fit\_performance <- collect\_metrics(l1\_84\_wf\_fit)

l1\_84\_wf\_fit\_performance

```

```{r}

l1\_84\_wf\_fit\_preds <- collect\_predictions(l1\_84\_wf\_fit)

```

```{r}

l1\_84\_wf\_fit\_preds %>%

group\_by(id) %>%

roc\_curve(truth = c1, .pred\_OFFENSE) %>%

autoplot()

```

### Select The Best

```{r}

chosen\_auc\_l1\_84\_wf\_fit <-

l1\_84\_wf\_fit %>%

select\_by\_one\_std\_err(metric = "roc\_auc", -penalty)

chosen\_auc\_l1\_84\_wf\_fit

```

## 8.5 Ridge-Regression-L2 with TF-IDF

### L2-Model

```{r}

l2\_85\_mod <- logistic\_reg(penalty = tune(), mixture = 0) %>%

set\_engine("glmnet") %>%

set\_mode("classification")

l2\_85\_mod

```

### Define The Workflow

```{r}

l2\_85\_wf <-workflow() %>%

add\_recipe(rec3) %>%

add\_model(l2\_85\_mod)

l2\_85\_wf

```

### Resampling & Model Quality

```{r}

options(mc.cores = parallel::detectCores())

l2\_85\_wf\_fit <- tune\_grid(

l2\_85\_wf,

folds,

grid = lambda\_grid,

control = control\_resamples(save\_pred = TRUE)

)

```

### Model Performance

```{r}

l2\_85\_wf\_performance <- collect\_metrics(l2\_85\_wf\_fit)

l2\_85\_wf\_performance

```

```{r}

l1\_85\_wf\_fit\_preds <- collect\_predictions(l1\_85\_wf\_fit)

```

```{r}

l1\_85\_wf\_fit\_preds %>%

group\_by(id) %>%

roc\_curve(truth = c1, .pred\_OFFENSE) %>%

autoplot()

```

### Show The Best

```{r}

show\_best(l2\_85\_wf\_fit)

```

### Select The Best

```{r}

chosen\_auc\_l2\_85\_wf\_fit <-

l2\_85\_wf\_fit %>%

select\_by\_one\_std\_err(metric = "roc\_auc", -penalty)

chosen\_auc\_l2\_85\_wf\_fit

```

## 8.6 Lasso-L1 With TF-IDF

### L1-Model

```{r}

l1\_86\_mod <- logistic\_reg(penalty = tune(), mixture = 1) %>%

set\_engine("glmnet") %>%

set\_mode("classification")

l1\_86\_mod

```

### Define The Workflow

```{r}

l1\_86\_wf <-workflow() %>%

add\_recipe(rec4) %>%

add\_model(l1\_86\_mod)

l1\_86\_wf

```

### Resampling & Model Quality

```{r}

options(mc.cores = parallel::detectCores())

l1\_86\_wf\_fit <- tune\_grid(

l1\_86\_wf,

folds,

grid = lambda\_grid,

control = control\_resamples(save\_pred = TRUE)

)

```

### Model Performance

```{r}

l1\_86\_wf\_performance <- collect\_metrics(l1\_86\_wf\_fit)

l1\_86\_wf\_performance

```

```{r}

l1\_86\_wf\_fit\_preds <- collect\_predictions(l1\_86\_wf\_fit)

```

```{r}

l1\_86\_wf\_fit\_preds %>%

group\_by(id) %>%

roc\_curve(truth = c1, .pred\_OFFENSE) %>%

autoplot()

```

### Show The Best

```{r}

select\_best(l1\_86\_wf\_fit)

```

### Select The Best

```{r}

chosen\_auc\_l1\_86\_wf\_fit <-

l1\_86\_wf\_fit %>%

select\_by\_one\_std\_err(metric = "roc\_auc", -penalty)

chosen\_auc\_l1\_86\_wf\_fit

```

## 8.7 Ridge-Regression-L2 With TF-IDF

### L2-Model

```{r}

l2\_87\_mod <- logistic\_reg(penalty = tune(), mixture = 0) %>%

set\_engine("glmnet") %>%

set\_mode("classification")

l2\_87\_mod

```

### Define The Workflow

```{r}

l2\_87\_wf <-workflow() %>%

add\_recipe(rec4) %>%

add\_model(l2\_87\_mod)

l2\_87\_wf

```

### Resampling & Model Quality

```{r}

options(mc.cores = parallel::detectCores())

l2\_87\_wf\_fit <- tune\_grid(

l2\_87\_wf,

folds,

grid = lambda\_grid,

control = control\_resamples(save\_pred = TRUE)

)

```

### Model Performance

```{r}

l2\_87\_wf\_performance <- collect\_metrics(l2\_87\_wf\_fit)

l2\_87\_wf\_performance

```

```{r}

l1\_87\_wf\_fit\_preds <- collect\_predictions(l1\_87\_wf\_fit)

```

```{r}

l1\_87\_wf\_fit\_preds %>%

group\_by(id) %>%

roc\_curve(truth = c1, .pred\_OFFENSE) %>%

autoplot()

```

### Select The Best

```{r}

chosen\_auc\_l2\_87\_wf\_fit <-

l2\_87\_wf\_fit %>%

select\_by\_one\_std\_err(metric = "roc\_auc", -penalty)

chosen\_auc\_l2\_87\_wf\_fit

```

# 9. Predictions