---

title: "Text Mining"

author: "Sabrina Michl"

date: "2023-01-25"

categories: [code, analysis]

bibliography: ref.bib

image: image.jpg

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.

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)

```

# Load Dataset And Minor Changes

## 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())

```

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())
```

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 sentimentanalyse or do lemmatization or stem the words. But we will have a look to these types in the different machine learning algorithmen now.

Preparation

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)
```

Define Recipe - rec2 - word embedding

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)))
```

Recipe Definition rec2

```{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)

rec2
```

```{r}
rec2_prep <- rec2 %>%
  prep() %>%
  recipes::bake(new_data = NULL)
```

Define Recipe - rec3 - Word Embeddings

Define 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)
```

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)
```

Build Resamples

Because of the large amount of the data and the extreme extensive recipes, we only use the V-Fold-Cross-Validation.

```{r}
folds <- vfold_cv(data = d_train,
                  v = 2,
                  repeats = 1,
                  strata = c1)
```

Build the Penalty-Grid

```{r}
lambda_grid <- grid_regular(penalty(),
                            levels = 30)
```

Build the Models

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")
```

Lasso-L1 with TF-IDF

```{r}
l1_8.2_mod <- logistic_reg(penalty = tune(), mixture = 1) %>%
  set_engine("glmnet") %>%
  set_mode("classification")
l1_8.2_mod
```

Define the Workflow

```{r}
l1_8.2_wf <-workflow() %>%
  add_recipe(rec1) %>%
  add_model(l1_8.2_mod)
l1_8.2_wf
```

Resampling & Model Quality

```{r}
options(mc.cores = parallel::detectCores())
l1_8.2_wf_fit <- tune_grid(
  l1_8.2_wf,
  folds,
```

```
  grid = lambda_grid,
  control = control_resamples(save_pred = TRUE)
)
```

```{r}
l1_8.2_wf_fit_performance <- collect_metrics(l1_8.2_wf_fit)
l1_8.2_wf_fit_performance
```

```{r}
select_best(l1_8.2_wf_fit)
```

```{r}
chosen_auc_l1_8.2_wf_fit <-
  l1_8.2_wf_fit %>%
  select_by_one_std_err(metric = "roc_auc", -penalty)
chosen_auc_l1_8.2_wf_fit
```

Ridge-Regression-L2 with TF-IDF

```{r}
l2_8.3_mod <- logistic_reg(penalty = tune(), mixture = 0) %>%
  set_engine("glmnet") %>%
  set_mode("classification")
l2_8.3_mod
```

Define the Workflow

```{r}
```

```
l2_8.3_wf <-workflow() %>%
  add_recipe(rec1) %>%
  add_model(l2_8.3_mod)
l2_8.3_wf
```

Resampling & Model Quality

```{r}
options(mc.cores = parallel::detectCores())
l2_8.3_wf_fit <- tune_grid(
  l2_8.3_wf,
  folds,
  grid = lambda_grid,
  control = control_resamples(save_pred = TRUE)
)
```

```{r}
collect_metrics(l2_8.3_wf_fit)
```

```{r}
chosen_auc_l2_8.3_wf_fit <-
  l2_8.3_wf_fit %>%
  select_by_one_std_err(metric = "roc_auc", -penalty)
chosen_auc_l2_8.3_wf_fit
```

Lasso-L1 with Word Embeddings

```{r}
l1_8.4_mod <- logistic_reg(penalty = tune(), mixture = 1) %>%
  set_engine("glmnet") %>%
```

```
  set_mode("classification")
l1_8.4_mod
```

Define the Workflow

```{r}
l1_8.4_wf <- workflow() %>%
  add_recipe(rec2) %>%
  add_model(l1_8.4_mod)
l1_8.4_wf
```

Resampling & Model Quality

```{r}
options(mc.cores = parallel::detectCores())
l1_8.4_wf_fit <- tune_grid(
  l1_8.4_wf,
  folds,
  grid = lambda_grid,
  control = control_resamples(save_pred = TRUE)
)
```

```{r}
collect_metrics(l1_8.4_wf_fit)
```

```{r}
chosen_auc_l1_8.4_wf_fit <-
  l1_8.4_wf_fit %>%
  select_by_one_std_err(metric = "roc_auc", -penalty)
chosen_auc_l1_8.4_wf_fit
```

```

## Ridge-Regression-L2 with Word Embeddings

```{r}
l2_8.5_mod <- logistic_reg(penalty = tune(), mixture = 0) %>%
  set_engine("glmnet") %>%
  set_mode("classification")
l2_8.5_mod
```

Define the Workflow

```{r}
l2_8.5_wf <- workflow() %>%
  add_recipe(rec2) %>%
  add_model(l2_8.5_mod)
l2_8.5_wf
```

Resampling & Model Quality

```{r}
options(mc.cores = parallel::detectCores())
l2_8.5_wf_fit <- tune_grid(
  l2_8.5_wf,
  folds,
  grid = lambda_grid,
  control = control_resamples(save_pred = TRUE)
)
```

```{r}
collect_metrics(l2_8.5_wf_fit)
```
```

```

```{r}
chosen_auc_l2_8.5_wf_fit <-
  l2_8.5_wf_fit %>%
  select_by_one_std_err(metric = "roc_auc", -penalty)
chosen_auc_l2_8.5_wf_fit
```

Lasso-L1 with Word Embeddings
```{r}
l1_8.6_mod <- logistic_reg(penalty = tune(), mixture = 1) %>%
  set_engine("glmnet") %>%
  set_mode("classification")
l1_8.6_mod
```

Define the Workflow
```{r}
l1_8.6_wf <- workflow() %>%
  add_recipe(rec3) %>%
  add_model(l1_8.6_mod)
l1_8.6_wf
```

Resampling & Model Quality
```{r}
options(mc.cores = parallel::detectCores())
l1_8.6_wf_fit <- tune_grid(
  l1_8.6_wf,
  folds,
  grid = lambda_grid,

```
  control = control_resamples(save_pred = TRUE)
)
```

Performance

```{r}
collect_metrics(l1_8.6_wf_fit)
```

Best Model

```{r}
chosen_auc_l1_8.6_wf_fit <-
  l1_8.6_wf_fit %>%
  select_by_one_std_err(metric = "roc_auc", -penalty)
chosen_auc_l1_8.6_wf_fit
```

Ridge-Regression-L2 with TF-IDF

```{r}
l2_8.7_mod <- logistic_reg(penalty = tune(), mixture = 0) %>%
  set_engine("glmnet") %>%
  set_mode("classification")
l2_8.7_mod
```

Define the Workflow

```{r}
l2_8.7_wf <-workflow() %>%
  add_recipe(rec3) %>%
  add_model(l2_8.7_mod)
l2_8.7_wf
```

Resampling & Model Quality

```{r}
options(mc.cores = parallel::detectCores())
l2_8.7_wf_fit <- tune_grid(
  l2_8.7_wf,
  folds,
  grid = lambda_grid,
  control = control_resamples(save_pred = TRUE)
)
```

```{r}
l2_8.7_wf_performance <- collect_metrics(l2_8.7_wf_fit)
l2_8.7_wf_performance
```

```{r}
select_best(l2_8.7_wf_fit)
```

```{r}
chosen_auc_l2_8.7_wf_fit <-
  l2_8.7_wf_fit %>%
  select_by_one_std_err(metric = "roc_auc", -penalty)
chosen_auc_l2_8.7_wf_fit
```

Lasso-L1 with TF-IDF

```{r}
l1_8.8_mod <- logistic_reg(penalty = tune(), mixture = 1) %>%
```

```
  set_engine("glmnet") %>%
  set_mode("classification")
l1_8.8_mod
```

Define the Workflow
```{r}
l1_8.8_wf <-workflow() %>%
  add_recipe(rec4) %>%
  add_model(l1_8.8_mod)
l1_8.8_wf
```

Resampling & Model Quality
```{r}
options(mc.cores = parallel::detectCores())
l1_8.8_wf_fit <- tune_grid(
  l1_8.8_wf,
  folds,
  grid = lambda_grid,
  control = control_resamples(save_pred = TRUE)
)
```

```{r}
l1_8.8_wf_performance <- collect_metrics(l1_8.8_wf_fit)
l1_8.8_wf_performance
```

```{r}
select_best(l1_8.8_wf_fit)
```

```{r}
chosen_auc_l1_8.8_wf_fit <-
  l1_8.8_wf_fit %>%
  select_by_one_std_err(metric = "roc_auc", -penalty)
chosen_auc_l1_8.8_wf_fit
```

Ridge-Regression-L2 with TF-IDF

```{r}
l2_8.9_mod <- logistic_reg(penalty = tune(), mixture = 0) %>%
  set_engine("glmnet") %>%
  set_mode("classification")
l2_8.9_mod
```

Define the Workflow

```{r}
l2_8.9_wf <-workflow() %>%
  add_recipe(rec4) %>%
  add_model(l2_8.9_mod)
l2_8.9_wf
```

Resampling & Model Quality

```{r}
options(mc.cores = parallel::detectCores())
l2_8.9_wf_fit <- tune_grid(
  l2_8.9_wf,
  folds,
  grid = lambda_grid,
  control = control_resamples(save_pred = TRUE)
```

)
```

```{r}
l2_8.9_wf_performance <- collect_metrics(l2_8.9_wf_fit)
l2_8.9_wf_performance
```

```{r}
select_best(l2_8.9_wf_fit)
```

```{r}
chosen_auc_l2_8.9_wf_fit <-
  l2_8.9_wf_fit %>%
  select_by_one_std_err(metric = "roc_auc", -penalty)
chosen_auc_l2_8.9_wf_fit
```
```