

DSP_tfidf_ds1

Melanie Weissenboeck

2022-10-16

Laden von Bibliotheken und Daten

```
library("xlsx")
library(word2vec)
library(udpipe)
library(SnowballC)
library(ggplot2)
library(tm)
library(wordcloud)
library(tidytext)
library(tidyr)
library(e1071)
library(caret)
library(mlbench)
```

Vorverarbeiten der Texte

Erstellen eines Korpus

Am Beginn der Vorverarbeitung wird zum `ds1` ein Wortkorpus erstellt. Dazu wird das `tm`-Package verwendet. Als Parameter wird angegeben, dass es sich um deutsche Texte handelt.

```
# CREATING CORPUS
# Corpus, VCorpus or SimpleCorpus -> SimpleCorpus
corp_ds1 = SimpleCorpus(VectorSource(ds1$ANF_BESCHREIBUNG), control = list(language = "de"))
corp_ds1
```

```
## <<SimpleCorpus>>
## Metadata: corpus specific: 1, document level (indexed): 0
## Content: documents: 378
```

Entfernen von störenden Zeichen

Es werden alle numerischen Zeichen aus den Texten entfernt. Zur Veranschaulichung wird ein Beispiel vor und nach dem Verarbeitungsschritt ausgegeben.

```
writeLines(as.character(corp_ds1[[300]]))
```

```
## Voraussetzung
## Die Strukturansicht ist geöffnet.
##
## Funktionselement
## Menü > DOKUMENT > Informationen anzeigen (Alt+I)
```

```
##
## Vorgehensweise
## 1. Selektieren Sie das gewünschte Dokument.
## 2. Betätigen Sie den Menüpunkt.
##
## Ergebnis
## Es wird ein Dialog mit den Aufgaben-, Dokument- und Seiteninformationen geöffnet.
##
## Tipp
## Zusätzlich zu den Dokumentinformationen wird im Dialog Informationen anzeigen für jede Seite, falls v
```

```
# REMOVING NUMBERS
corp_ds1 = tm_map(corp_ds1, removeNumbers)
#writeLines(as.character(corp_ds1[[1]]))
writeLines(as.character(corp_ds1[[300]]))
```

```
## Voraussetzung
## Die Strukturansicht ist geöffnet.
##
## Funktionselement
## Menü > DOKUMENT > Informationen anzeigen (Alt+I)
##
## Vorgehensweise
## . Selektieren Sie das gewünschte Dokument.
## . Betätigen Sie den Menüpunkt.
##
## Ergebnis
## Es wird ein Dialog mit den Aufgaben-, Dokument- und Seiteninformationen geöffnet.
##
## Tipp
## Zusätzlich zu den Dokumentinformationen wird im Dialog Informationen anzeigen für jede Seite, falls v
```

Nach den numerischen Zeichen werden auch die Satzzeichen entfernt.

```
# REMOVE PUNCTUATION
corp_ds1 = tm_map(corp_ds1, removePunctuation)
writeLines(as.character(corp_ds1[[300]]))
```

```
## Voraussetzung
## Die Strukturansicht ist geöffnet
##
## Funktionselement
## Menü DOKUMENT Informationen anzeigen AltI
##
## Vorgehensweise
## Selektieren Sie das gewünschte Dokument
## Betätigen Sie den Menüpunkt
##
## Ergebnis
## Es wird ein Dialog mit den Aufgaben Dokument und Seiteninformationen geöffnet
##
## Tipp
## Zusätzlich zu den Dokumentinformationen wird im Dialog Informationen anzeigen für jede Seite falls v
```

Leerzeichen und Abstände werden ebenfalls aus dem Datenset entfernt.

```

# STRIPWHITESPACE
corp_ds1 = tm_map(corp_ds1, stripWhitespace)
writeLines(as.character(corp_ds1[[300]]))

## Voraussetzung Die Strukturansicht ist geöffnet Funktionselement Menü DOKUMENT Informationen anzeigen
# wird ausgelassen - ohne diese Schritte besseres Ergebnis
# REMOVING STOPWORDS
#corp_ds1 <- tm_map(corp_ds1, removeWords, stopwords("german"))

# STEMMING
#corp_ds1 <- tm_map(corp_ds1, stemDocument)
#writeLines(as.character(corp_ds1[[300]]))

```

Erstellen von Matrizen

Mit den vorverarbeiteten Texten kann nun eine Matrix erstellt werden. Zunächst wird eine **term-document**-Matrix erstellt und danach eine **document-term**-Matrix. Bei letzterer muss angegeben werden, welche Gewichtungsfunktion verwendet wird, in diesem Fall wird anhand **tf-idf** gewichtet.

```

# term document matrix
tdm_ds1 <- TermDocumentMatrix(corp_ds1, control = list(removeSparseTerms = TRUE,
                                                         removePunctuation = TRUE,
                                                         removeNumbers = TRUE,
                                                         stopwords = TRUE,
                                                         stemming = FALSE))

# document term matrix
dtm_ds1 <- DocumentTermMatrix(corp_ds1,
                              control = list(weighting = function(x)
                                              weightTfIdf(x, normalize = FALSE), stopwords = TRUE))

```

```
tdm_ds1
```

```

## <<TermDocumentMatrix (terms: 1757, documents: 378)>>
## Non-/sparse entries: 10110/654036
## Sparsity           : 98%
## Maximal term length: 51
## Weighting          : term frequency (tf)

```

```
dtm_ds1
```

```

## <<DocumentTermMatrix (documents: 378, terms: 1757)>>
## Non-/sparse entries: 10110/654036
## Sparsity           : 98%
## Maximal term length: 51
## Weighting          : term frequency - inverse document frequency (tf-idf)

```

Die **document-term**-Matrix ist nur sehr dünn besetzt. Um den Speicherbedarf zu reduzieren, werden einige Terme weggelassen und somit die Dimension reduziert.

```

# A term-document matrix where those terms from x are removed
# which have at least a sparse percentage of empty
# (i.e., terms occurring 0 times in a document) elements.
# Resulting matrix contains only terms with a sparse factor less than sparse
dtm_ds1 <- removeSparseTerms(dtm_ds1, 0.80)
dtm_ds1

```

```
## <<DocumentTermMatrix (documents: 378, terms: 15)>>
## Non-/sparse entries: 1655/4015
## Sparsity          : 71%
## Maximal term length: 13
## Weighting          : term frequency - inverse document frequency (tf-idf)
```

```
inspect(tdm_ds1)
```

```
## <<TermDocumentMatrix (terms: 1757, documents: 378)>>
## Non-/sparse entries: 10110/654036
## Sparsity          : 98%
## Maximal term length: 51
## Weighting          : term frequency (tf)
## Sample            :
```

```
##              Docs
## Terms      139 159 160 185 186 217 218 228 250 258
## anfrage      0  0  0  0  0  0  0  0  0  0
## anwendung    3  3  0  3  3  0  0  0  0  0
## aufgabe     12 12  1 12 12  2  1  0  0  0
## betätigen    2  2  0  2  2  1  1  1  0  2
## dokument     0  0  0  0  0  4  5  5  2  3
## erfassen     3  3  0  3  3  0  0  0  1  0
## feld         0  0  0  0  0  0  0  0  0  0
## ticket       5  5  1  5  5  0  0  0  0  0
## trennblatt   0  0  0  0  0  0  0  0  0  0
## wurde        1  1  1  1  1  0  0  0  0  0
```

```
inspect(dtm_ds1)
```

```
## <<DocumentTermMatrix (documents: 378, terms: 15)>>
## Non-/sparse entries: 1655/4015
## Sparsity          : 71%
## Maximal term length: 13
## Weighting          : term frequency - inverse document frequency (tf-idf)
## Sample            :
```

```
##              Terms
## Docs  aufgabe auswählen bereits betätigen button dokument erstellen
## 138 16.06123 0.000000 2.314315 0.000000 5.075633 0 8.092334
## 139 24.09184 6.022961 4.628630 2.666847 1.691878 0 6.069251
## 143 10.03827 2.007654 4.628630 1.333424 1.691878 0 2.023084
## 144 10.03827 2.007654 4.628630 1.333424 1.691878 0 2.023084
## 145 10.03827 2.007654 4.628630 1.333424 1.691878 0 2.023084
## 146 10.03827 2.007654 4.628630 1.333424 1.691878 0 2.023084
## 159 24.09184 6.022961 4.628630 2.666847 1.691878 0 6.069251
## 184 16.06123 0.000000 2.314315 0.000000 5.075633 0 8.092334
## 185 24.09184 6.022961 4.628630 2.666847 1.691878 0 6.069251
## 186 24.09184 6.022961 4.628630 2.666847 1.691878 0 6.069251
```

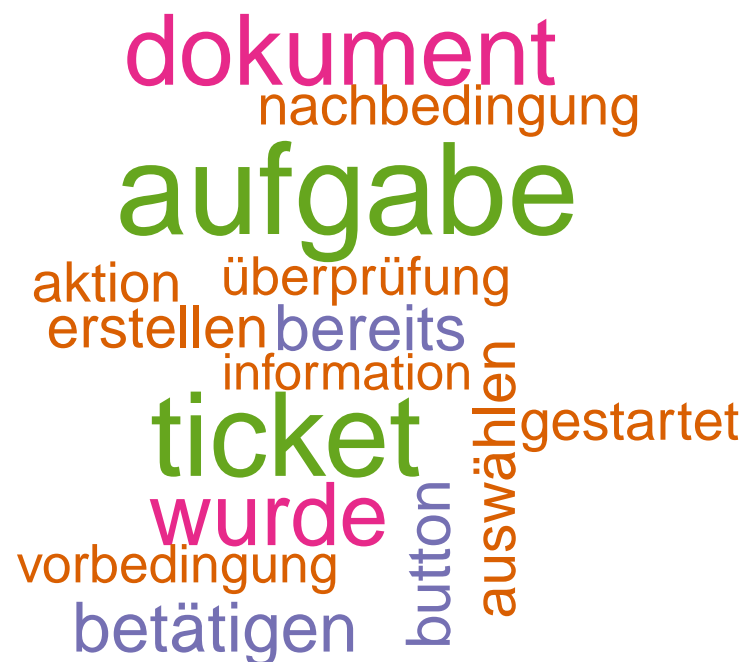
```
##              Terms
## Docs nachbedingung ticket wurde
## 138 2.992306 2.295456 2.647675
## 139 1.496153 11.477279 1.323838
## 143 1.496153 11.477279 3.971513
## 144 1.496153 11.477279 3.971513
## 145 1.496153 11.477279 3.971513
## 146 1.496153 11.477279 3.971513
```

```
## 159      1.496153 11.477279 1.323838
## 184      2.992306  2.295456 2.647675
## 185      1.496153 11.477279 1.323838
## 186      1.496153 11.477279 1.323838
```

Darstellen einer Wortwolke

Zur Veranschaulichung der reduzierten `document-term`-Matrix wird eine Wortwolke erstellt, in der die wichtigsten Begriffe des Datensets gezeigt werden.

```
# show a wordcloud
freq = data.frame(sort(colSums(as.matrix(dtm_ds1)), decreasing=TRUE))
wordcloud(rownames(freq), freq[,1], max.words=100, colors=brewer.pal(5, "Dark2"))
```



Konvertieren der Dokument Term Matrix zu Dataframe

Um die konstruierte Matrix mit den anderen Prädiktoren zusammengefügt werden kann, muss diese zunächst in einen Dataframe transformiert werden.

```
tmp_df_ds1 <- tidy(dtm_ds1)
head(tmp_df_ds1)
```

```
## # A tibble: 6 x 3
##   document term      count
##   <chr>    <chr>    <dbl>
## 1 6      information  2.10
## 2 7      information  2.10
## 3 9      button     1.69
```

```
## 4 9      dokument      2.24
## 5 10     button       3.38
## 6 11     aufgabe       2.01

tmp_df_ds1 <- tmp_df_ds1 |> pivot_wider(names_from = term, values_from = count,
                                       names_repair = "unique", values_fill = 0)

colnames(tmp_df_ds1)[1] <- "doc_id"
tmp_df_ds1$doc_id <- as.integer(tmp_df_ds1$doc_id)
tmp_df_ds1$row_sum <- rowSums(tmp_df_ds1)

rbind(tmp_df_ds1, sum(tmp_df_ds1[, 1:length(tmp_df_ds1)]))

## # A tibble: 263 x 17
##   doc_id information button dokument aufgabe auswählen aktion betätigen
##   <dbl>      <dbl> <dbl>    <dbl>    <dbl>    <dbl> <dbl>    <dbl>
## 1      6      2.10  0      0      0      0      0      0
## 2      7      2.10  0      0      0      0      0      0
## 3      9      0      1.69  2.24  0      0      0      0
## 4     10      0      3.38  0      0      0      0      0
## 5     11      0      0      0      2.01  0      0      0
## 6     12      0      0      2.24  2.01  0      0      0
## 7     13      0      1.69  0      2.01  0      0      0
## 8     21      0      0      0      0      2.01  0      0
## 9     23      0      0      4.48  0      0      0      0
## 10    24      0      1.69  0      0      2.01  1.46  2.67
## # ... with 253 more rows, and 9 more variables: erstellen <dbl>,
## #   gestartet <dbl>, nachbedingung <dbl>, vorbedingung <dbl>, wurde <dbl>,
## #   überprüfung <dbl>, bereits <dbl>, ticket <dbl>, row_sum <dbl>
```

Zusammenführen mit anderen Prädiktoren

Im Folgenden werden alle Prädiktoren in einem Dataframe zusammengefasst. Dieser stellt die Ausgangslage für die Klassifikation dar.

```
final_ds1 <- cbind(ds1$ANF_RISIKO, ds1$TF_ABDECKUNG,
                  ds1$AKT_RES_RELEASE, ds1$AKT_RES_STATUS, tmp_df_ds1[1:378, 2:17])

final_ds1 <- final_ds1[1:262, 1:17]
summary(final_ds1)
```

```
## ds1$ANF_RISIKO      ds1$TF_ABDECKUNG ds1$AKT_RES_RELEASE ds1$AKT_RES_STATUS
## Length:262         Min.   : 0.00   Length:262         Length:262
## Class :character   1st Qu.: 50.00   Class :character   Class :character
## Mode  :character   Median :100.00   Mode  :character   Mode  :character
##                   Mean    : 73.09
##                   3rd Qu.:100.00
##                   Max.    :100.00
##   information      button      dokument      aufgabe
## Min.   :0.0000    Min.   :0.0000    Min.   : 0.000    Min.   : 0.000
## 1st Qu.:0.0000    1st Qu.:0.0000    1st Qu.: 0.000    1st Qu.: 0.000
## Median :0.0000    Median :0.0000    Median : 0.000    Median : 0.000
## Mean   :0.7223    Mean   :0.9299    Mean   : 1.667    Mean   : 2.276
## 3rd Qu.:2.1028    3rd Qu.:1.6919    3rd Qu.: 2.240    3rd Qu.: 2.008
## Max.   :4.2056    Max.   :8.4594    Max.   :20.163    Max.   :24.092
##   auswählen      aktion      betätigen      erstellen
```

```
## Min. :0.0000 Min. :0.0000 Min. :0.000 Min. :0.0000
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.000 1st Qu.:0.0000
## Median :0.0000 Median :1.4642 Median :1.333 Median :0.0000
## Mean :0.8506 Mean :0.8327 Mean :1.059 Mean :0.8417
## 3rd Qu.:2.0077 3rd Qu.:1.4642 3rd Qu.:1.333 3rd Qu.:2.0231
## Max. :6.0230 Max. :2.9284 Max. :2.667 Max. :8.0923
## gestartet nachbedingung vorbedingung wurde
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.000
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.000
## Median :0.0000 Median :1.4962 Median :1.4642 Median :1.324
## Mean :0.8109 Mean :0.8337 Mean :0.7656 Mean :1.450
## 3rd Qu.:1.5622 3rd Qu.:1.4962 3rd Qu.:1.4642 3rd Qu.:2.648
## Max. :3.1245 Max. :2.9923 Max. :1.4642 Max. :5.295
## überprüfung
## Min. :0.0000
## 1st Qu.:0.0000
## Median :0.0000
## Mean :0.7673
## 3rd Qu.:1.9329
## Max. :3.8658
```

```
df <- final_ds1
names(df)[names(df)=="ds1$ANF_RISIKO"] <- "ANF_RISIKO"
names(df)[names(df)=="ds1$TF_ABDECKUNG"] <- "TF_ABDECKUNG"
names(df)[names(df)=="ds1$AKT_RES_RELEASE"] <- "AKT_RES_RELEASE"
names(df)[names(df)=="ds1$AKT_RES_STATUS"] <- "AKT_RES_STATUS"
```

Normalisieren numerischer Spalten

Mittels min-max-Normalisierung werden die numerischen Spalten auf eine gemeinsame Skalierung gebracht. Zur besseren Übersicht wird am Ende nochmal eine Zusammenfassung ausgegeben.

```
set.seed(1234)
# definiere normalisierungsfunktion
min_max_norm <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}
# alle spalten normalisieren
df[, 5:17] <- as.data.frame(lapply(df[, 5:17], min_max_norm))
df[2] <- as.data.frame(lapply(df[2], min_max_norm))
df$ANF_RISIKO <- as.factor(df$ANF_RISIKO)
df$AKT_RES_STATUS <- as.factor(df$AKT_RES_STATUS)
df$AKT_RES_RELEASE <- as.factor(df$AKT_RES_RELEASE)
summary(df)
```

```
## ANF_RISIKO TF_ABDECKUNG AKT_RES_RELEASE AKT_RES_STATUS information
## gering: 73 Min. :0.0000 21x :26 FAILED: 9 Min. :0.0000
## hoch : 83 1st Qu.:0.5000 22.10 :59 OK :247 1st Qu.:0.0000
## mittel:106 Median :1.0000 22.20 :91 OPEN : 6 Median :0.0000
## Mean :0.7309 22.30 : 2 Mean :0.1718
## 3rd Qu.:1.0000 OLDERT21:84 3rd Qu.:0.5000
## Max. :1.0000 Max. :1.0000
## button dokument aufgabe auswählen
## Min. :0.0000 Min. :0.0000 Min. :0.00000 Min. :0.0000
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.00000 1st Qu.:0.0000
```

```
## Median :0.0000 Median :0.0000 Median :0.00000 Median :0.0000
## Mean :0.1099 Mean :0.0827 Mean :0.09447 Mean :0.1412
## 3rd Qu.:0.2000 3rd Qu.:0.1111 3rd Qu.:0.08333 3rd Qu.:0.3333
## Max. :1.0000 Max. :1.0000 Max. :1.00000 Max. :1.0000
## aktion betätigen erstellen gestartet
## Min. :0.0000 Min. :0.0000 Min. :0.000 Min. :0.0000
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.000 1st Qu.:0.0000
## Median :0.5000 Median :0.5000 Median :0.000 Median :0.0000
## Mean :0.2844 Mean :0.3969 Mean :0.104 Mean :0.2595
## 3rd Qu.:0.5000 3rd Qu.:0.5000 3rd Qu.:0.250 3rd Qu.:0.5000
## Max. :1.0000 Max. :1.0000 Max. :1.000 Max. :1.0000
## nachbedingung vorbedingung wurde überprüfung
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000
## Median :0.5000 Median :1.0000 Median :0.2500 Median :0.0000
## Mean :0.2786 Mean :0.5229 Mean :0.2739 Mean :0.1985
## 3rd Qu.:0.5000 3rd Qu.:1.0000 3rd Qu.:0.5000 3rd Qu.:0.5000
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000
```

Klassifikation

Erstellen von Train- / Test-Split

Die vorliegenden Daten werden in Trainings- und Testdaten aufgeteilt im Verhältnis 80:20.

```
# partition erstellen
part <- createDataPartition(df$ANF_RISIKO, times = 1, p = 0.80)
# extract training set
X_train <- df[part$Resample1, ]
# extract testing set
X_test <- df[-part$Resample1, ]
# extract target
y_train <- df[part$Resample1, 1]
y_test <- df[-part$Resample1, 1]
```

NaiveBayes Klassifikation

Ein Naive-Bayes Klassifikator wird erstellt und mit den Trainingsdaten trainiert. Anhand der Testdaten wird das Modell evaluiert. Die Ergebnisse werden in einer Confusionmatrix angegeben.

```
model_nb = naiveBayes(ANF_RISIKO ~ ., data = X_train)

pred_nb <- predict(model_nb, X_test)
mat.nb <- confusionMatrix(pred_nb, X_test$ANF_RISIKO, mode = "prec_recall")
mat.nb
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction gering hoch mittel
##      gering      9      1      7
##      hoch       2     10      6
##      mittel      3      5      8
##
## Overall Statistics
```



```
##
##           Accuracy : 0.5294
##           95% CI   : (0.3846, 0.6707)
##    No Information Rate : 0.4118
##    P-Value [Acc > NIR] : 0.05977
##
##           Kappa : 0.2961
##
##    McNemar's Test P-Value : 0.56739
##
## Statistics by Class:
##
##           Class: gering Class: hoch Class: mittel
## Precision           0.5294      0.5556      0.5000
## Recall              0.6429      0.6250      0.3810
## F1                  0.5806      0.5882      0.4324
## Prevalence          0.2745      0.3137      0.4118
## Detection Rate      0.1765      0.1961      0.1569
## Detection Prevalence 0.3333      0.3529      0.3137
## Balanced Accuracy    0.7133      0.6982      0.5571
```

KNN Klassifikation

Analog zum Naive-Bayes Klassifikator wird auch ein KNN Modell trainiert. Auch hier wird das Ergebnis anhand einer Confusionmatrix gezeigt.

```
model_knn <- train(ANF_RISIKO ~ ., data = X_train, "knn",
trControl = trainControl(method = "cv", number = 5))
model_knn
```

```
## k-Nearest Neighbors
##
## 211 samples
## 16 predictor
## 3 classes: 'gering', 'hoch', 'mittel'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 169, 169, 168, 169, 169
## Resampling results across tuning parameters:
##
##  k  Accuracy  Kappa
##  5  0.5829457  0.3706194
##  7  0.5737542  0.3553897
##  9  0.5356589  0.3031042
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
pred_knn <- predict(model_knn, X_test, type = "raw")
mat.knn <- confusionMatrix(pred_knn, X_test$ANF_RISIKO, mode = "prec_recall")
mat.knn

## Confusion Matrix and Statistics
##
##           Reference
```

```

## Prediction gering hoch mittel
##      gering      9      2      2
##      hoch       1     11      3
##      mittel      4      3     16
##
## Overall Statistics
##
##              Accuracy : 0.7059
##              95% CI : (0.5617, 0.8251)
##      No Information Rate : 0.4118
##      P-Value [Acc > NIR] : 2.059e-05
##
##              Kappa : 0.5489
##
## McNemar's Test P-Value : 0.8013
##
## Statistics by Class:
##
##              Class: gering Class: hoch Class: mittel
## Precision              0.6923      0.7333      0.6957
## Recall                  0.6429      0.6875      0.7619
## F1                      0.6667      0.7097      0.7273
## Prevalence              0.2745      0.3137      0.4118
## Detection Rate          0.1765      0.2157      0.3137
## Detection Prevalence    0.2549      0.2941      0.4510
## Balanced Accuracy       0.7674      0.7866      0.7643

```