Supervised learning aggregated

Cornelius Erfort

5/10/2021

This script requires the files “sample_germany.dta” and “data_joint.RDS” in the parent directory.

Loading packages

This script is based mainly on the functions of the quanteda package. For the cross-validation of the textmodels, quanteda.classifiers has to be loaded from GitHub.

```
packages <- c("quanteda", "quanteda.textmodels", "dplyr", "caret", "randomForest",
    "tm", "beepr", "rmarkdown", "e1071", "penalized", "plyr", "readr", "repr", "ggplot2",
    "rsample", "remotes", "stringr", "formatR", "haven", "lubridate")

lapply(packages[!(packages %in% rownames(installed.packages()))], install.packages)

if (!("quanteda.classifiers" %in% rownames(installed.packages()))) {
    remotes::install_github("quanteda/quanteda.classifiers")
}

lapply(c(packages, "quanteda.classifiers"), require, character.only = T)
```

Loading data

The sample data for Germany consists of 2,742 labelled press releases. The dataset is not on GitHub and is loaded from the parent directory here.

```
sample_germany <- read_dta("../sample_germany.dta")

# Correcting classification for three documents
sample_germany$issue[sample_germany$id == 229] <- 191
sample_germany$issue[sample_germany$id == 731] <- 7
sample_germany$issue[sample_germany$id == 902] <- 10

# Subset to relevant vars
germany_textpress <- sample_germany %>%
    select("header", "text", "issue", "position", "id")

# Distribution of issues in the hand-coded sample
table(germany_textpress$issue)
```

```
##
##   1   2   3   4   5   6   7   8   9  10  12  13  14  15  16  17  18  20  23  98
## 175 181 119  99 167 137  84 105 131  74 195 104  32 168 121  68  27  97  19  91
##  99 191 192
##  46 350 152
```

Merging categories

To improve the classification similar topics are merged. In practice, press releases regarding, for instance, Environment and Energy are often not distinguishable. Furthermore, small categories with very few observations are not suitable for automated classification.

```
germany_textpress$issue_r1 <- as.numeric(germany_textpress$issue)

germany_textpress <- germany_textpress %>% mutate(issue_r1 = recode(issue_r1,
                           `8`  = 7,   # Environment & Energy
                           `13` = 10, # Transportation & Welfare
                           `14` = 10, # Housing & Welfare
                           `18` = 15, # Foreign Trade and Domestic Commerce
                           `98` = 99, # Non-thematic & Other
                           `23` = 99) # Culture: Too few observations
                                                  )
# Distribution with merged categories
table(germany_textpress$issue_r1)
```

```
##
##   1   2   3   4   5   6   7   9  10  12  15  16  17  20  99 191 192
## 175 181 119  99 167 137 189 131 210 195 195 121  68  97 156 350 152
```

Creating the document frequency matrix (dfm)

We create a text corpus based on the header and text of each press release. We draw a random sample from the corpus to create a training and a test dataset. The test dataset consists of approx. one fifth of the documents.

Subsequently, we follow standard procedures for the preparation of the document frequency matrix. First, we remove stopwords and stem the words in order to better capture the similarities across documents. Second, we remove all punctuation, numbers, symbols and URLs. In a last step, we remove all words occurring in less than 0.5% or more than 90% of documents.

```
corp_press <- str_c(germany_textpress$header, " ", germany_textpress$text) %>% corpus()
```

```
## Warning: NA is replaced by empty string
```

```
# Add id var to corpus
docvars(corp_press, "id") <- germany_textpress$id
docvars(corp_press, "issue_r1") <- germany_textpress$issue_r1

# Create random sample for test dataset (size: 1/5 of all classified documents)
set.seed(300)
id_test <- sample(docvars(corp_press, "id"),
                  round(length(docvars(corp_press, "id"))/5, 0), replace = FALSE)

# Create training and test set
dfmat_training <- corpus_subset(corp_press, !(id %in% id_test)) %>%
  dfm(remove = stopwords("de"), # Stem and remove stopwords, punctuation etc.
      stem = T,
      remove_punct = T,
      remove_number = T,
      remove_symbols = T,
      remove_url = T) %>%
  dfm_trim(min_docfreq = 0.005, # Remove words occurring <.5% or > 80% of docs
           max_docfreq = .9,
          docfreq_type = "prop")

dfmat_test <- corpus_subset(corp_press, id %in% id_test) %>%
   dfm(remove = stopwords("de"), # Stem and remove stopwords, punctuation etc.
      stem = T,
      remove_punct = T,
      remove_number = T,
      remove_symbols = T,
      remove_url = T) %>%
  dfm_trim(min_docfreq = 0.005, # Remove words occurring <.5% or > 80% of docs
          max_docfreq = .9,
          docfreq_type = "prop")
```

Multinomial Naive Bayes classification model

We calculate a Multinomial Naive Bayes text classification model. Multinomial NB models take into account the number of times a word occurs in a document, whereas Bernoulli NB models use the presence or absence of words only.

```
tmod_nb_r1 <- textmodel_nb(dfmat_training, dfmat_training$issue_r1, distribution = "multinomial")
```

Evaluation

To evaluate the quality of the textmodel, we compare the actual labels of the test dataset with those predicted by the model.

We perform a five-fold cross-validation of the fitted textmodel in order to ensure the robustness

```
dfmat_matched <- dfm_match(dfmat_test, features = featnames(dfmat_training))

actual_class <- docvars(dfmat_matched, "issue_r1")
predicted_class <- predict(tmod_nb_r1, newdata = dfmat_matched)
tab_class <- table(actual_class, predicted_class)
tab_class
```

```
##               predicted_class
## actual_class  1  2  3  4  5  6  7  9 10 12 15 16 17 20 99 191 192
##          1   23  0  0  0  4  1  4  0  1  1  2  0  1  0  0   0   2
##          2    0 18  2  1  1  1  0  2  0  8  0  1  0  2  1   4   0
##          3    1  1 16  0  0  1  1  0  0  0  0  0  0  0  0   1   0
##          4    0  0  0 14  0  0  1  0  0  0  1  0  0  0  0   1   0
##          5    4  0  2  0 23  0  1  1  0  1  1  0  0  0  0   0   1
##          6    0  0  0  0  2 21  0  0  3  0  0  0  0  0  1   0   0
##          7    0  0  0  4  0  0 28  0  3  1  2  0  1  1  0   1   1
##          9    0  0  0  0  1  0  0 14  0  2  0  0  0  1  1   1   0
##          10   1  1  0  0  3  1  5  1 17  1  7  0  0  1  1   1   0
##          12   1  3  0  1  0  1  1  3  0 19  2  2  1  0  2   2   2
##          15   1  3  1  0  0  1  2  0  3  2 13  0  1  0  1   1   9
##          16   0  2  1  0  0  0  0  0  0  2  1 15  0  2  0   3   0
##          17   0  2  0  0  0  3  0  0  0  1  1  0  3  0  2   0   0
##          20   2  0  0  0  0  2  1  0  0  1  0  1  1  4  0   0   0
##          99   2  0  0  0  0  4  0  0  0  0  0  2  0  4 20   2   0
##          191  0  1  0  1  0  1  3  1  0  1  0  4  0  3  0  55   4
##          192  2  0  0  0  0  0  0  1  0  1  3  0  0  0  2   3  19
```

```
# Five-fold cross-validation
crossval(tmod_nb_r1, k = 5)
```

```
##         precision            recall                f1          accuracy
##         0.6241461         0.6399636         0.6244665         0.6207797
## balanced_accuracy
##         0.6035375
```

Classifying unlabelled data

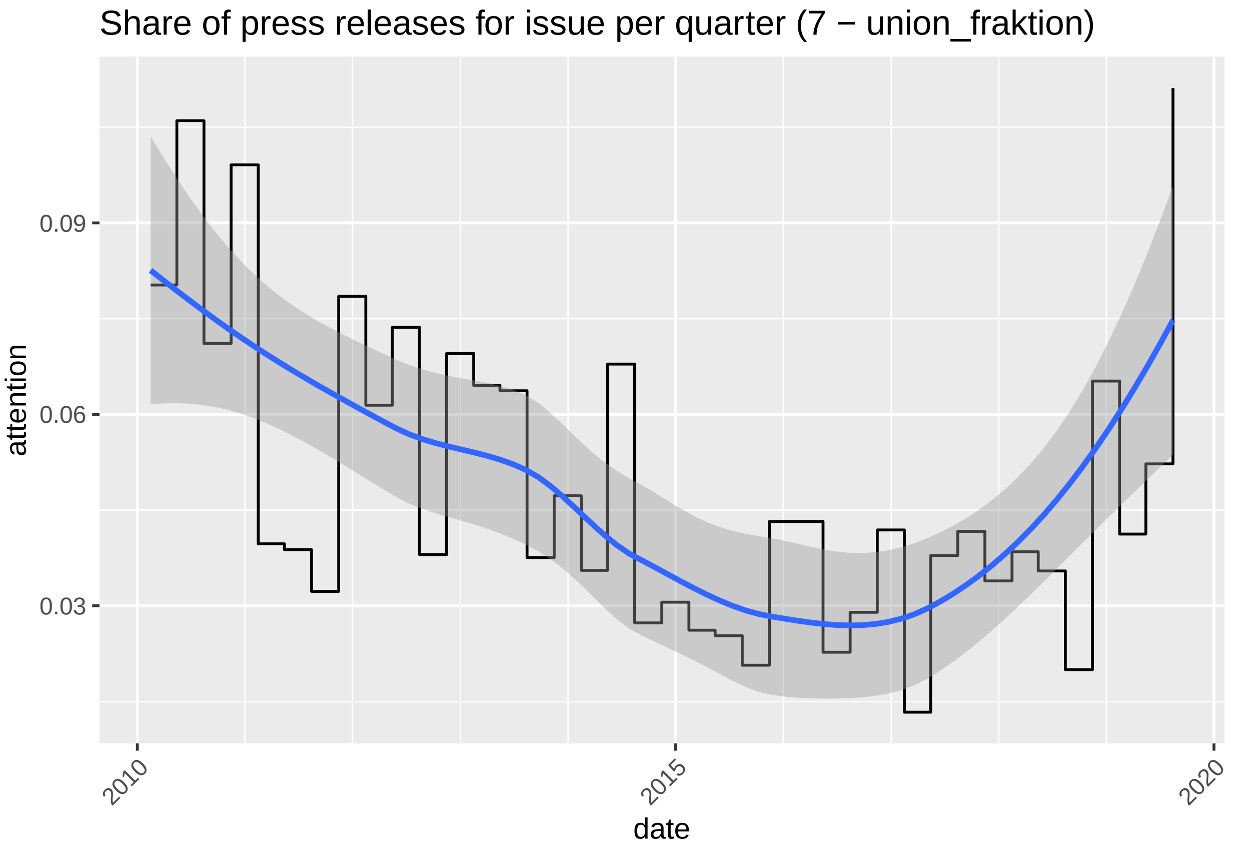
```
# Loading full dataset from parent dir
all_germany <- read_rds("../data_joint.RDS") %>% select(c(header, text.x, date.x, issue, party.x, id))

# Constructing the document frequency matrix
dfmat_all <- corpus(str_c(all_germany$header, " ", all_germany$text.x)) %>%
  dfm(remove = stopwords("de"), # Stem and remove stopwords, punctuation etc.
      stem = T,
      remove_punct = T,
      remove_number = T,
      remove_symbols = T,
      remove_url = T)

# Adding docvars
docvars(dfmat_all, "party") <- all_germany$party.x
docvars(dfmat_all, "date") <- all_germany$date.x
docvars(dfmat_all, "id") <- all_germany$id

# Subsetting to features in the training data
dfmat_all <- dfm_match(dfmat_all, features = featnames(dfmat_training))

# Predicting the issue category for all documents
dfmat_all$issue_r1 <- predict(tmod_nb_r1, newdata = dfmat_all)

table(dfmat_all$issue_r1)
```

```
##
##    1    2    3    4    5    6    7    9   10   12   15   16   17   20   99  191
## 2734 2633 1701 1947 2958 3340 3330 2420 3033 3510 3057 1978 1093 1591 2678 6115
##  192
## 2993
```

Aggregation of the issues categories over time and party

To measure parties' evolving issue agendas, we aggregate the category counts over time.

```
# Create dataframe from dfm
issue_agendas <- data.frame(date = docvars(dfmat_all, "date"), party = docvars(dfmat_all,
    "party"), issue_r1 = docvars(dfmat_all, "issue_r1"))

# Make date quarterly
issue_agendas$date <- as.character(issue_agendas$date) %>%
    substr(1, 8) %>%
    str_c("15") %>%
    str_replace_all(c(`-01-` = "-02-", `-03-` = "-02-", `-04-` = "-05-", `-06-` = "-05-",
        `-07-` = "-08-", `-09-` = "-08-", `-10-` = "-11-", `-12-` = "-11-")) %>%
    ymd()

# Add variable for counting
issue_agendas$freq <- 1

# Aggregate by party, date and issue
issue_agendas <- aggregate(freq ~ party + date + issue_r1, issue_agendas, sum)

# Add var for total press releases per party and month
issue_agendas$party_sum <- ave(issue_agendas$freq, issue_agendas$date, issue_agendas$party,
    FUN = sum)

issue_agendas$attention <- issue_agendas$freq/issue_agendas$party_sum

if (!dir.exists("plots")) dir.create("plots")

# Plot quarterly issue attention for category '7 Environment & Energy' for
# 'union_fraktion'
plot_issue <- 7
plot_party <- "union_fraktion"
ggplot(issue_agendas %>%
    filter(issue_r1 == plot_issue & party == plot_party), aes(x = date, y = attention)) +
    geom_step() + geom_smooth(method = "loess", formula = "y ~ x") + theme(axis.text.x = element_text(an
    hjust = 1)) + scale_x_date(date_minor_breaks = "1 year") + ggtitle("Share of press releases for issu
    str_c(" (", plot_issue, " - ", plot_party, ")")) + ggsave("plots/7_union_fraktion.pdf",
    device = cairo_pdf, width = 6 * 2^0.5, height = 6)
```


Share of press releases for issue per quarter (7 − union_fraktion)
attention
0.09
0.06
0.03
date
2010
2015
2020