Supervised learning aggregated

Cornelius Erfort

5/10/2021

Contents

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", "SuperLearner")

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.

```r
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.

```r
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
                                                 )
# Category descriptions

issue_categories <- data.frame(issue_r1 = c(1:7, 9:10, 12, 15:17, 20, 99, 191:192), issue_r1_descr = c(

# 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 dfm
dfmat <- corpus_subset(corp_press) %>%
  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")

# Create training and test set
dfmat_training <- dfm_subset(dfmat, !(id %in% id_test))
dfmat_test <- dfm_subset(dfmat, id %in% id_test)
```

Textmodels

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.

```
# Five-fold cross-validation for every possible parameter combination
nb_eval <- textmodel_evaluate(dfmat, dfmat$issue_r1, k = 5, model = "textmodel_nb",
    fun = c("accuracy", "precision", "recall", "f1_score"), parameters = list(prior = c("uniform",
        "docfreq", "termfreq"), distribution = c("multinomial", "Bernoulli")))
head(nb_eval)
```

```
##   k  accuracy precision    recall  f1_score   prior distribution time
## 1 1 0.6265938 0.6080332 0.6180546 0.6049494 uniform  multinomial 0.18
## 2 2 0.6934307 0.6824506 0.6766623 0.6745682 uniform  multinomial 0.14
## 3 3 0.6478102 0.6518453 0.6373741 0.6352488 uniform  multinomial 0.14
## 4 4 0.6021898 0.5800243 0.5988456 0.5797627 uniform  multinomial 0.29
## 5 5 0.6265938 0.6372297 0.6202543 0.6205441 uniform  multinomial 0.14
##         seed
## 1 1620824575
## 2 1620824575
## 3 1620824575
## 4 1620824575
## 5 1620824575
```

```
test <- textmodel_evaluate(dfmat, dfmat$issue_r1, k = 5, model = "textmodel_nb",
    fun = c("accuracy", "precision", "recall", "f1_score"), parameters = list(prior = c("uniform"),
        distribution = c("multinomial")))

aggregate(cbind(accuracy, precision, recall, f1_score, time, seed) ~ prior + distribution,
    nb_eval[, -c(1)], mean) %>%
    arrange(desc(accuracy))
```

```
##      prior distribution  accuracy precision    recall  f1_score   time
## 1 termfreq  multinomial 0.6396899 0.6336533 0.6267681 0.6214223 0.1420
## 2  docfreq  multinomial 0.6393243 0.6335034 0.6257910 0.6207084 0.1420
## 3  uniform  multinomial 0.6393237 0.6319166 0.6302382 0.6230147 0.1780
## 4  uniform    Bernoulli 0.5496847 0.5803885 0.4983525 0.5049924 0.1500
## 5  docfreq    Bernoulli 0.5487723 0.5848722 0.4958818 0.5035218 0.1375
## 6 termfreq    Bernoulli 0.5483153 0.5848687 0.4956095 0.5034937 0.1575
##         seed
## 1 1620824575
## 2 1620824575
## 3 1620824575
## 4 1620824575
## 5 1620824575
## 6 1620824575
```

SVM

Linear predictive models estimation based on the LIBLINEAR C/C++ Library.

```
# Five-fold cross-validation for every possible parameter combination
svm_eval <- textmodel_evaluate(dfmat, dfmat$issue_r1, k = 5, model = "textmodel_svm",
    fun = c("accuracy", "precision", "recall", "f1_score"), parameters = list(weight = c("uniform",
        "docfreq", "termfreq"), type = c(0:7)))
head(svm_eval)
```

```
##   k  accuracy precision    recall  f1_score  weight type time       seed
## 1 1 0.6247723 0.6246101 0.6181015 0.6152794 uniform    0 2.13 1620824582
## 2 2 0.5821168 0.5847376 0.5521110 0.5610000 uniform    0 1.59 1620824582
## 3 3 0.6240876 0.6339587 0.5967090 0.6071835 uniform    0 1.48 1620824582
## 4 4 0.5912409 0.5975627 0.5630337 0.5687519 uniform    0 1.44 1620824582
## 5 5 0.6029144 0.5978814 0.5893072 0.5899750 uniform    0 1.43 1620824582
```

```
aggregate(cbind(accuracy, precision, recall, f1_score, time, seed) ~ weight + type,
    svm_eval[, -c(1)], mean) %>%
    arrange(desc(accuracy))
```

```
##      weight type  accuracy precision    recall  f1_score  time       seed
## 1   uniform    7 0.6057550 0.6095070 0.5852324 0.5898926 2.694 1620824582
## 2   uniform    0 0.6050264 0.6077501 0.5838525 0.5884380 1.614 1620824582
## 3   docfreq    7 0.6039328 0.6475351 0.5650202 0.5750842 2.270 1620824582
## 4   docfreq    0 0.6035692 0.6446532 0.5641279 0.5737091 1.436 1620824582
## 5  termfreq    0 0.6017437 0.6428035 0.5614224 0.5704086 1.440 1620824582
## 6  termfreq    7 0.5984630 0.6438577 0.5568317 0.5645979 2.202 1620824582
## 7   docfreq    2 0.5948167 0.5968501 0.5761193 0.5766482 1.446 1620824582
## 8  termfreq    2 0.5929919 0.5935869 0.5730199 0.5730773 1.592 1620824582
## 9   uniform    6 0.5929872 0.5917936 0.5768207 0.5788305 1.690 1620824582
## 10 termfreq    1 0.5798639 0.5804470 0.5648348 0.5663264 2.130 1620824582
## 11  docfreq    1 0.5795002 0.5806852 0.5650311 0.5669516 2.162 1620824582
## 12  uniform    2 0.5776741 0.5772543 0.5641388 0.5643924 1.274 1620824582
## 13 termfreq    4 0.5754863 0.5728639 0.5609942 0.5614729 1.232 1620824582
## 14  docfreq    5 0.5751240 0.5653028 0.5359957 0.5371802 2.674 1620824582
## 15 termfreq    5 0.5736688 0.5714961 0.5355178 0.5331050 2.874 1620824582
## 16  docfreq    4 0.5736635 0.5709949 0.5589198 0.5594514 1.156 1620824582
## 17 termfreq    3 0.5736608 0.5716781 0.5573660 0.5578820 2.070 1620824582
## 18  docfreq    3 0.5725686 0.5720500 0.5588629 0.5594580 1.972 1620824582
## 19  uniform    1 0.5692892 0.5700381 0.5535308 0.5552475 2.072 1620824582
## 20  uniform    4 0.5656402 0.5604428 0.5484679 0.5483226 1.206 1620824582
## 21  uniform    3 0.5645493 0.5640355 0.5495501 0.5508299 2.074 1620824582
## 22  uniform    5 0.5554266 0.5477262 0.5459543 0.5422274 8.252 1620824582
## 23  docfreq    6 0.4767577 0.4737274 0.4102561 0.4127446 1.535 1620824582
## 24 termfreq    6 0.4507299 0.4519468 0.3845728 0.3898422 1.560 1620824582
```

?

```
svm_eval <- textmodel_evaluate(dfmat, dfmat$issue_r1, k = 5, model = "textmodel_svm",
    fun = c("accuracy", "precision", "recall", "f1_score"), parameters = list(weight = c("uniform",
        "docfreq", "termfreq")))
head(svm_eval)
```

```
##   k  accuracy precision    recall  f1_score  weight time       seed
## 1 1 0.5591985 0.5560708 0.5662563 0.5532027 uniform 2.21 1620824841
## 2 2 0.5492701 0.5550465 0.5500492 0.5469206 uniform 2.04 1620824841
## 3 3 0.5656934 0.5567543 0.5523037 0.5489956 uniform 1.90 1620824841
## 4 4 0.5474453 0.5459126 0.5247964 0.5291947 uniform 1.89 1620824841
## 5 5 0.6120219 0.5949361 0.5912846 0.5877243 uniform 1.85 1620824841
```

```
aggregate(cbind(accuracy, time, seed) ~ weight, svm_eval[, -c(1)], mean) %>%
    arrange(desc(accuracy))
```

```
##     weight  accuracy  time       seed
## 1 termfreq 0.5809501 1.820 1620824841
## 2  docfreq 0.5794916 1.916 1620824841
## 3  uniform 0.5667258 1.978 1620824841
```

Ensemble methods: SuperLearner?

```
# training <- !(1:ndoc(corp_press) %in% id_test) Xtrain <-
# as.data.frame(as.matrix(dfm_trim(dfmat, min_docfreq = 15,
# max_docfreq=0.80*nrow(dfmat), verbose = T))) features <- names(Xtrain)
# names(Xtrain) <- paste0('V', 1:ncol(Xtrain)) set.seed(777) sl <- SuperLearner(Y
# = dfmat$issue_r1[training], X = Xtrain[training,], family = binomial(),
# SL.library = c('SL.glmnet'), cvControl=list(V=2))
```

Classification of unlabelled data

Using the NB textmodel

We trained the models using a set of 2,742 labelled documents. In order to obtain aggregated measures of issue attention, we predict the issue categories of all 47,111 labelled and unlabelled press releases in our sample.

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

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   24  0  0  0  4  1  4  0  1  1  2  0  1  0  0   0   1
##          2    0 20  2  1  0  1  0  2  0  7  0  2  0  2  1   3   0
##          3    1  1 16  0  0  1  1  0  0  0  0  0  0  0  0   1   0
##          4    0  0  0 16  0  0  1  0  0  0  0  0  0  0  0   0   0
##          5    4  0  2  0 23  0  0  1  0  1  2  0  0  0  0   0   1
##          6    0  1  0  0  2 20  0  0  3  0  0  0  0  0  1   0   0
##          7    0  0  0  4  0  0 29  0  3  1  1  0  1  1  0   1   1
##          9    0  0  0  0  1  1  0 13  0  1  0  0  0  2  1   1   0
##          10   1  1  0  1  3  1  5  1 17  1  6  0  0  1  1   1   0
##          12   0  2  0  1  0  1  2  3  0 21  2  1  1  1  1   2   2
##          15   1  3  1  0  0  1  3  0  2  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  0  0  4  0  2   0   0
##          20   3  0  0  0  0  2  1  0  0  1  0  1  1  3  0   0   0
##          99   2  0  0  0  0  4  0  0  0  0  0  2  1  2 20   2   1
##          191  0  1  0  1  0  1  3  0  0  1  0  5  0  3  0  55   4
##          192  2  0  0  0  0  0  0  1  0  1  2  0  0  0  2   3  20
```

```
# 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
```

```r
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
## 2714 2672 1703 1925 2925 3306 3361 2411 2997 3574 3086 1988 1122 1619 2652 6083
##  192
## 2973
```

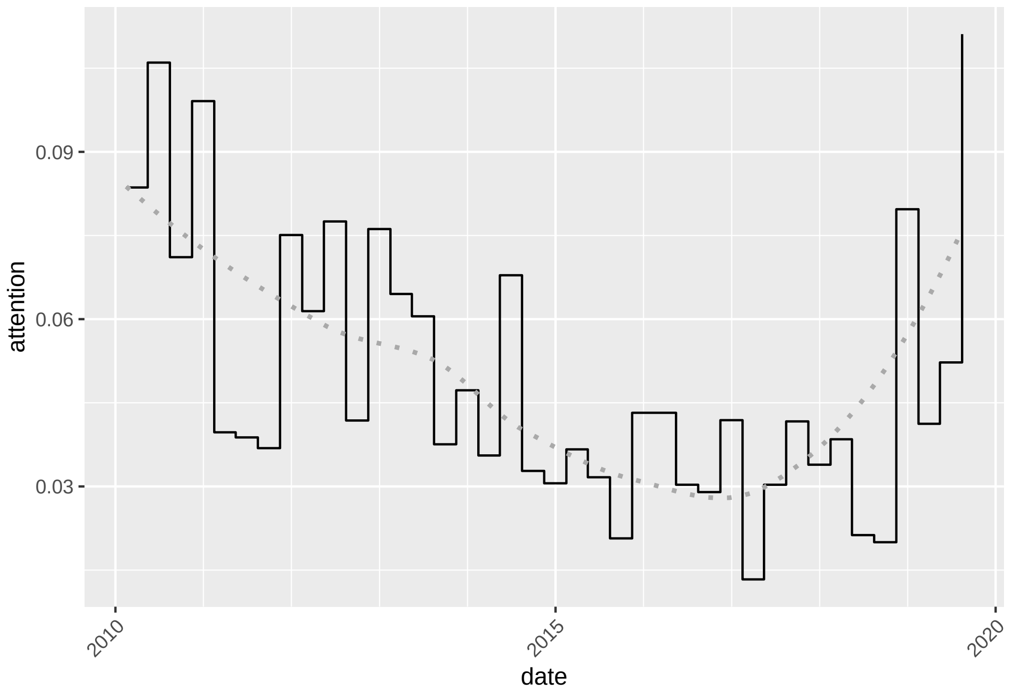
Aggregation of the issues categories over time and party

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

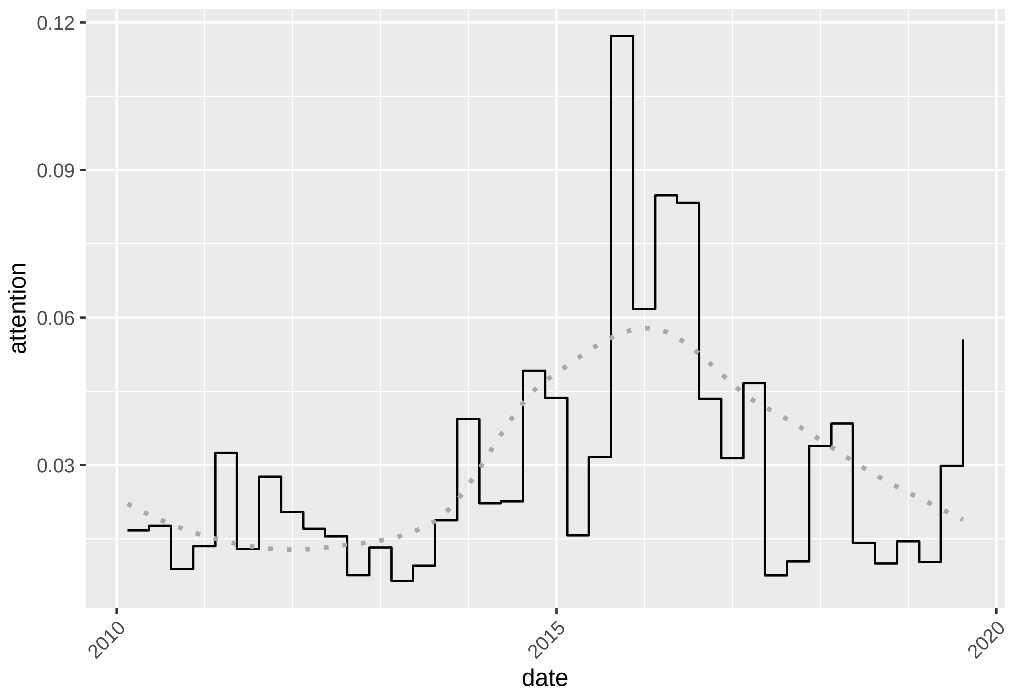
```r
# Create dataframe from dfm
issue_agendas <- data.frame(date = docvars(dfmat_all, "date"), party = docvars(dfmat_all,
    "party"), issue_r1 = docvars(dfmat_all, "issue_r1"))

# 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

# Add issue descriptions
issue_agendas <- merge(issue_agendas, issue_categories, by = "issue_r1")
```

Plotting issue attention: Examples

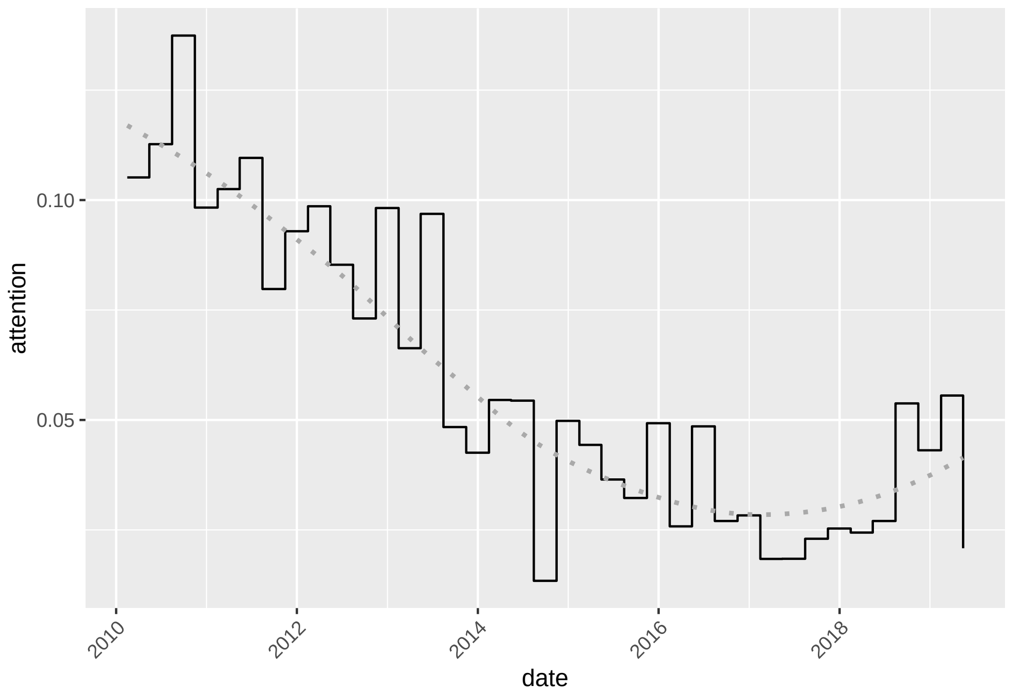
```r
if (!dir.exists("plots")) dir.create("plots")

# Function for plotting parties' issue attention over time
plot_issue_party <- function(plot_issue, plot_party) 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", color = "dark grey",
    lty = 3, se = F) + theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    scale_x_date(date_minor_breaks = "1 year") + # ggtitle('Share of press releases for issue per quart
# issue_categories$issue_r1_descr[issue_categories$issue_r1 == plot_issue], ' -
# ', plot_party, ')')) +
ggsave(str_c("plots/", plot_issue, " - ", issue_categories$issue_r1_descr[issue_categories$issue_r1 ==
    plot_issue], "_", plot_party, ".pdf"), device = cairo_pdf, width = 5 * 2^0.5,
    height = 5) + ggsave(str_c("plots/", plot_issue, " - ", issue_categories$issue_r1_descr[issue_catego
    plot_issue], "_", plot_party, ".png"), width = 5 * 2^0.5, height = 5)


# Plot quarterly issue attention for category '7 Environment & Energy' for
# 'union_fraktion'
plot_issue_party(7, "union_fraktion")  # There seems to be a decline since Fukushima in 2011.
```



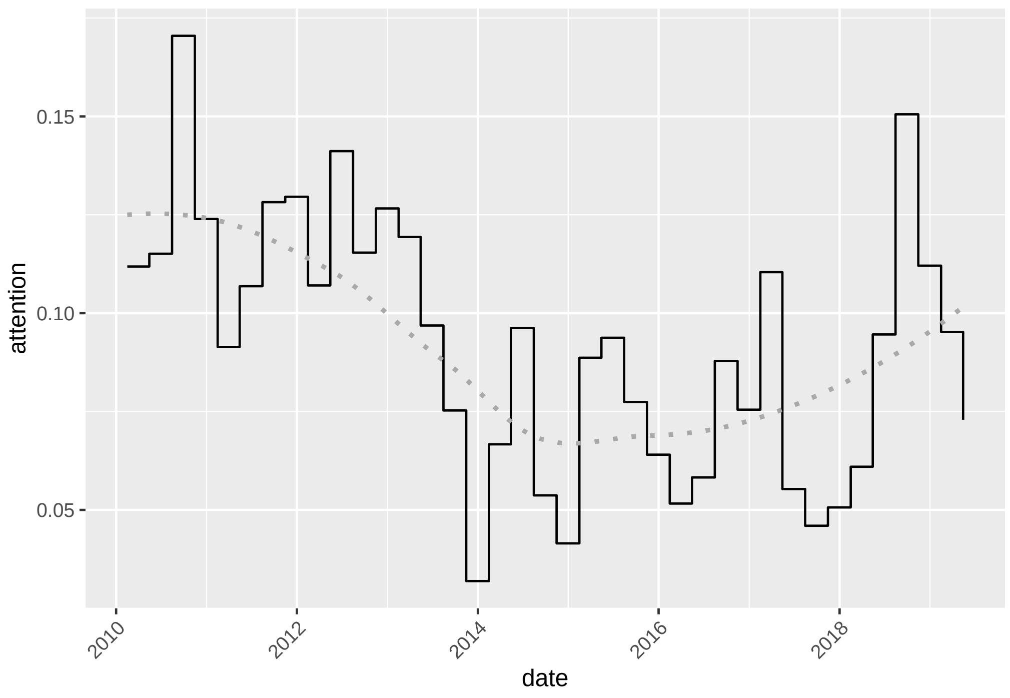
```
# Plot quarterly issue attention for category '9 Immigration' for
# 'union_fraktion'
plot_issue_party(9, "union_fraktion")  # There seems to be a peak around the so-called refugee crisis i
```



```
# Plot quarterly issue attention for category '7 Environment & Energy' for
# 'spd_fraktion'
plot_issue_party(7, "spd_fraktion")  # There seems to be a decline since Fukushima in 2011.
```



```
# Plot quarterly issue attention for category '10 Welfare' for 'spd_fraktion'
plot_issue_party(10, "spd_fraktion")  # There seems to be a lower emphasis on welfare after the entry i
```


0.15
0.10
0.05
attention
2010
2012
2014
2016
2018
date