Supervised learning aggregated

Cornelius Erfort

5/10/2021

Contents

1 Setting up

This script requires the files "sample_germany.dta" and "data_joint.RDS" in the parent directory. It also writes and reads two csv-files from the parent directory.

1.1 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", "readstata13", "lubridate", "reticulate")

# Using readstata13 because haven causes crash

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

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

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

1.2 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.dta13("../sample_germany.dta", convert.factors = F)

# 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
```

1.3 Merging categories

In order to improve the classification, similar topics are merged or subsumed under the "Other" category. 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
```

1.4 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") %>%
  suppressWarnings()

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

# Make order of documents random
dfmat <- dfmat[sample(1:ndoc(dfmat), ndoc(dfmat)), ]
```

2 Textmodels

2.1 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"), smooth = c(1,
```

```
        2, 3)))
head(nb_eval)
```

```
##   k  accuracy precision    recall  f1_score   prior distribution smooth time
## 1 1 0.6448087 0.6259133 0.6360242 0.6226354 uniform  multinomial      1 0.26
## 2 2 0.6149635 0.5916292 0.5905809 0.5890856 uniform  multinomial      1 0.21
## 3 3 0.6587591 0.6300781 0.6419045 0.6310083 uniform  multinomial      1 0.31
## 4 4 0.6277372 0.6205147 0.6225183 0.6114838 uniform  multinomial      1 0.21
## 5 5 0.6539162 0.6374436 0.6324565 0.6304221 uniform  multinomial      1 0.21
##         seed
## 1 1621271121
## 2 1621271121
## 3 1621271121
## 4 1621271121
## 5 1621271121
```

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

```
##       prior distribution smooth  accuracy precision    recall  f1_score
## 1   uniform  multinomial      1 0.6400370 0.6211158 0.6246969 0.6169270
## 2  termfreq  multinomial      1 0.6396727 0.6245469 0.6228926 0.6169522
## 3   docfreq  multinomial      1 0.6378478 0.6237741 0.6206248 0.6154525
## 4   docfreq  multinomial      2 0.6280025 0.6219481 0.5944700 0.5942840
## 5   uniform  multinomial      2 0.6272732 0.6215357 0.5985178 0.5974855
## 6  termfreq  multinomial      2 0.6269076 0.6197666 0.5952323 0.5942883
## 7   uniform  multinomial      3 0.6130483 0.6221279 0.5750871 0.5762528
## 8  termfreq  multinomial      3 0.6130450 0.6318800 0.5727763 0.5747389
## 9   docfreq  multinomial      3 0.6123170 0.6323590 0.5708494 0.5729715
## 10  uniform    Bernoulli      1 0.5471372 0.5687063 0.4871938 0.4936392
## 11 termfreq    Bernoulli      1 0.5404518 0.5623367 0.4763830 0.4811234
## 12  docfreq    Bernoulli      1 0.5398435 0.5614665 0.4751121 0.4801336
##         time       seed
## 1  0.2400000 1621271121
## 2  0.2040000 1621271121
## 3  0.2120000 1621271121
## 4  0.2060000 1621271121
## 5  0.2000000 1621271121
## 6  0.2060000 1621271121
## 7  0.2120000 1621271121
## 8  0.2080000 1621271121
## 9  0.2100000 1621271121
## 10 0.3233333 1621271121
## 11 0.2166667 1621271121
## 12 0.2033333 1621271121
```

Assuming a multinomial distribution of text features leads to a higher accuracy of the models compared to a Bernoulli distribution. The other benchmark parameters confirm this finding.

There is no clear pattern in regard to the effect of the priors on the quality of the models. A smoothing parameter of 1 for the feature counts seems optimal.

2.2 SVM

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

(Not running here to save time.)

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

None of the configurations lead to a higher accuracy compared to the benchmark NB model.

Regarding the type of linear models, type 0 (L2-regularized logistic regression, primal) and 7 (L2-regularized logistic regression, dual) seem to provide optimal results. There is no clear pattern regarding the weights.

The calculation of the models requires much more time/computing compared to the NB models.

2.3 Next algorithm...

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

3 Ensemble methods: SuperLearner

Prepare the datasets for the python code chunk.

```
# Get vector indicating training sample
training <- !(1:ndoc(corp_press) %in% id_test)

# Subset training sample and write
if (!file.exists("../train_data.csv")) as.data.frame(as.matrix(dfm_trim(dfmat, min_docfreq = 15,
    max_docfreq = 0.8 * nrow(dfmat), verbose = T))) %>%
    write.csv(train_data, "../train_data.csv")

# Get labels for training sample and write
if (!file.exists("../train_labels.csv")) write.csv(dfmat$issue_r1, "../train_labels.csv")
```

Run the SuperLearner in Python.

```
# Load packages
import pandas as pd
import numpy as np

# Load sklearn tools
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load classifiers
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import AdaBoostClassifier
```

```
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB

# Load SuperLearner
from mlens.ensemble import SuperLearner

# create a list of base-models
def get_models():
    models = list()
    models.append(MultinomialNB())
    models.append(BernoulliNB())
    models.append(LogisticRegression(solver = 'liblinear', max_iter = 200))
    models.append(DecisionTreeClassifier())
    models.append(SVC(gamma = 'scale', probability = True))
    models.append(GaussianNB())
    models.append(KNeighborsClassifier())
    models.append(AdaBoostClassifier())
    models.append(BaggingClassifier())
    models.append(RandomForestClassifier())
    models.append(ExtraTreesClassifier())
    return models

# create the super learner
def get_super_learner(X):
    ensemble = SuperLearner(scorer = None, folds = 5, shuffle = True, random_state = np.random.seed(302

    # add base models
    models = get_models()
    ensemble.add(models, proba = True)

    # add the meta model
    ensemble.add_meta(LogisticRegression(solver = "lbfgs", max_iter = 200), proba = False)
    return ensemble

# load press release data
train_data = pd.read_csv("../train_data.csv", index_col = 0)
train_labels = pd.read_csv("../train_labels.csv", index_col = 0)

X = train_data.values # Documents are already in random order
y = np.asarray([int(i) for i in train_labels.values])

# create the inputs and outputs

# split data
train, test, train_val, test_val = train_test_split(X, y, test_size = 0.20) # test_size = 0.50
print('Train', train.shape, train_val, 'Test', test.shape, test_val.shape)

# create the super learner
ensemble = get_super_learner(train)
```

```
# fit the super learner
ensemble.fit(train, train_val)

# summarize base learners
print(ensemble.data)

# make predictions on hold out set
test_hat = ensemble.predict(test)

print('Super Learner: %.3f' % (accuracy_score(test_val, test_hat) * 100))
```

```
## Train (2193, 3386) [7 2 6 ... 6 4 9] Test (549, 3386) (549,)
##
## Fitting 2 layers
## Fit complete                        | 00:19:32
##                                      ft-m    ft-s  pt-m  pt-s
## layer-1  adaboostclassifier         36.20   0.88  0.94  0.05
## layer-1  baggingclassifier          70.48   4.68  0.60  0.35
## layer-1  bernoullinb                 7.12   1.48  0.15  0.14
## layer-1  decisiontreeclassifier     22.39   4.67  0.02  0.01
## layer-1  extratreesclassifier       68.64   1.02  0.44  0.12
## layer-1  gaussiannb                  0.63   0.16  0.72  0.08
## layer-1  kneighborsclassifier        0.00   0.00  0.86  0.21
## layer-1  logisticregression          4.69   0.27  0.03  0.01
## layer-1  multinomialnb              10.37   0.75  0.04  0.01
## layer-1  randomforestclassifier     28.30   1.18  0.34  0.05
## layer-1  svc                       393.87  85.50  9.92  4.59
##
##
## Predicting 2 layers
## Predict complete                    | 00:00:07
## [MLENS] backend: threading
## Super Learner: 62.842
```

The SuperLearner does not significantly increase the accuracy.

4 Classification of unlabelled data

4.1 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
```

```r
# 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) %>% suppressWarnings()

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

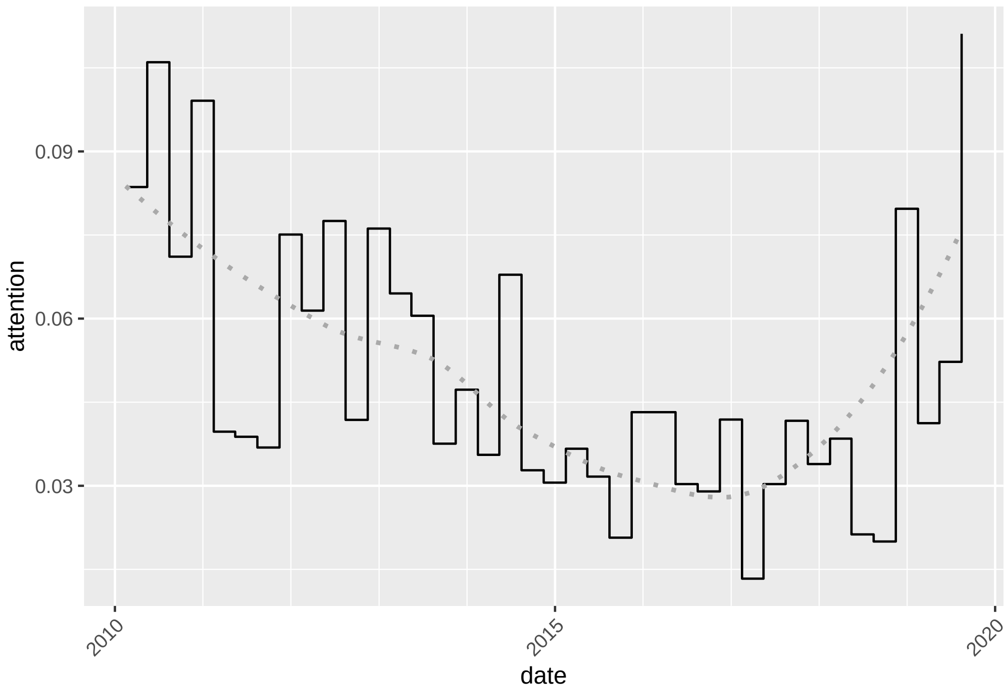
4.2 Aggregation of the issues categories over time and party

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

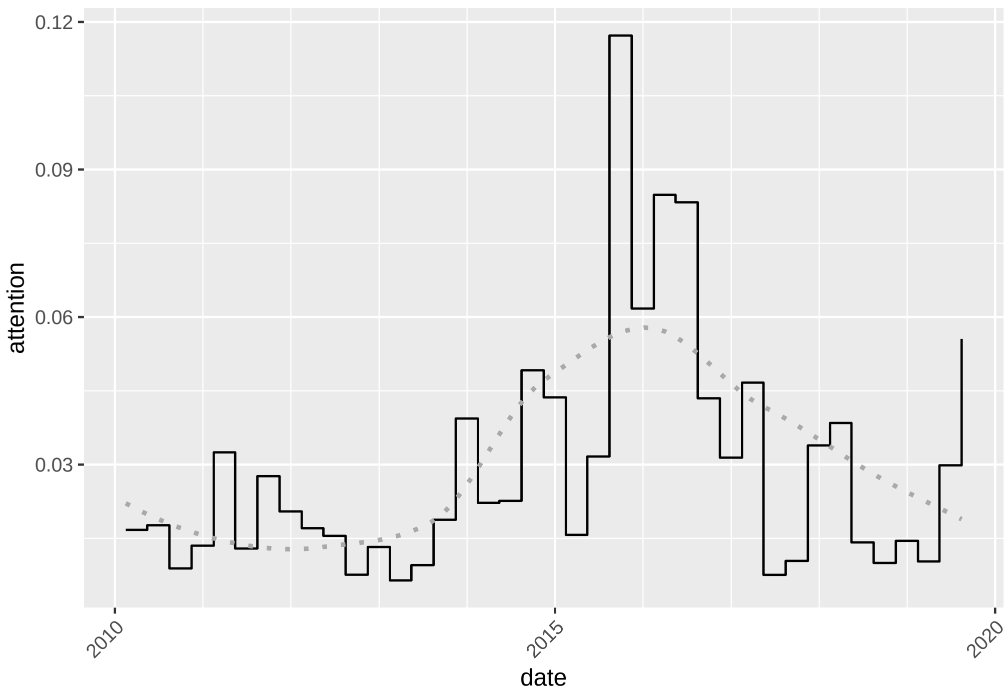
```r
# Create dataframe from dfm
issue_agendas <- data.frame(date = docvars(dfmat_all, "date"), party = docvars(dfmat_all,
    "party"), issue_r1 = docvars(dfmat_all, "issue_r1"))

# 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")
```

4.3 Plotting issue attention: Examples

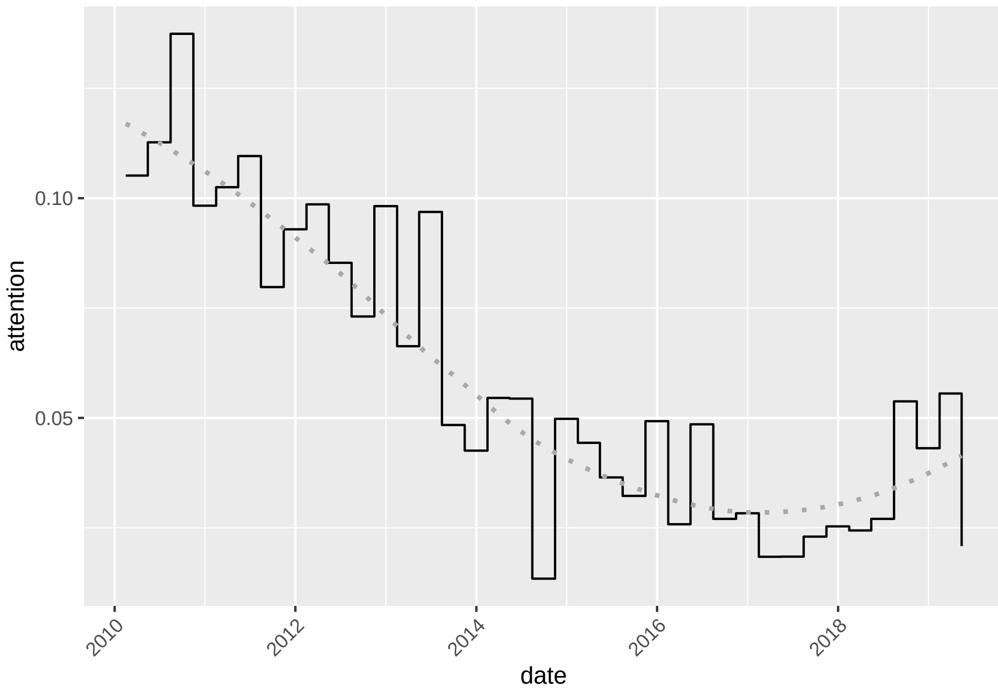
```r
if (!dir.exists("plots")) dir.create("plots")

# 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
```