# ReadMe

Cornelius Erfort

5/24/2021

## Contents



# 1 Setting up

This script requires the files "germany_textpress.RData" and "data_joint.RDS" which are not included on GitHub.

## 1.1 Loading packages

```
start_time <- Sys.time()

packages <- c("dplyr", "tm", "rmarkdown", "plyr", "readr", "ggplot2", "stringr",
    "formatR", "readstata13", "lubridate", "extrafont", "kableExtra", "stargazer",
    "word2vec", "tokenizers", "ggforce")

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

if (!("readme" %in% rownames(installed.packages()))) {
    remotes::install_github("iqss-research/readme-software/readme")
}

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

invisible(lapply(c(packages, "readme", "tensorflow"), require, character.only = T))

loadfonts()
loadfonts(device = "pdf")
theme_update(text = element_text(family = "LM Roman 10"))  # Set font family for ggplot
```

```
source("scripts/functions.R")
```

### 1.2 Loading the corpus

The sample data for Germany consists of 2,740 labeled press releases. The dataset is not uploaded on GitHub.

In order to improve the classification, similar topics were 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.

The corpus is generated in the script "Preparing the textual data". We create a text corpus based on the header and text of each press release.

```
load("supervised-files/germany_textpress.RData")
load("supervised-files/issue_categories.RData")
```

## 2 Readme2: Calculating proportions

In this section we use the package readme2 by Jerzak et al. (2019) to estimate the proportion of press releases regarding each topic. We do so by defining five folds for cross-validation. Our test dataset thus makes up 20% of documents.

In a first step, we load a word vector (embeddings trained on German Wikipedia, source: https://deepset.ai/german-word-embeddings). Second, we generate word vector summaries for all documents. Third, we define five folds of our training dataset. Fourth, we run the readme function to obtain predictions about the proportions in our test data.

(Jerzak, C. T., King, G., & Strezhnev, A. (forthcoming). An improved method of auto-mated nonparametric content analysis for social science. Political Analysis.)

```
# Run readme for each fold Load when file exists
if (!("readme_cv" %in% ls())) if (file.exists("readme-files/readme_cv.RData")) {

    load("readme-files/readme_cv.RData")

} else {
    # If not load word vector and calculate summaries

    if (!("wordVec_summaries_labeled" %in% ls()))
        if (file.exists("readme-files/wordvectors/wordVec_summaries_labeled.RData")) {

            load("readme-files/wordvectors/wordVec_summaries_labeled.RData")
        } else {

            # Read raw vector data and create vector matrix
            # (https://deepset.ai/german-word-embeddings)
            if (!("glove_ger" %in% ls()))
                if (file.exists("readme-files/wordvectors/glove_ger.RData")) {
                  load("readme-files/wordvectors/glove_ger.RData")
                } else {
                  glove_ger_raw <- scan(file = "readme-files/wordvectors/GloVe-german.txt",
                    what = "", sep = "\n")
                  glove_ger <- proc_pretrained_vec(glove_ger_raw)
                  glove_feat <- names(glove_ger)
                  glove_ger <- glove_ger %>%
```

```
                    t()
                  save(glove_ger, file = "readme-files/wordvectors/glove_ger.RData")

                }

            ## Generate a word vector summary for all documents
            wordVec_summaries_labeled <- undergrad(documentText = cleanme(germany_textpress$htext),
                wordVecs = glove_ger %>%
                  as.matrix)
            save(wordVec_summaries_labeled, file = "readme-files/wordvectors/wordVec_summaries_labeled.
        }

    readme_cv <- list()
    for (i in unique(germany_textpress$cv_sample) %>%
        sort) {
        # Estimate category proportions
        set.seed(2138)  # Set a seed if you choose
        readme_cv[[i]] <- readme(dfm = wordVec_summaries_labeled, labeledIndicator = ifelse(germany_tex
            i, 0, 1), categoryVec = germany_textpress$issue_r1, nCores = 2, nCores_OnJob = 2) %>%
            suppressWarnings()
    }
    save(readme_cv, file = "readme-files/readme_cv.RData")
}
```

# 3 Evaluation of Readme2

In order to evaluate the performance of readme2, we compare the predicted proportions in the test data with the true values. We present a table and plot to illustrate the link between predicted and true values.

```
# Prediction estimate and truth in %
agg_eval <- data.frame()
for (i in 1:length(readme_cv)) {
    readme.estimates <- readme_cv[[i]]
    readme.estimates$point_readme

    agg_eval <- data.frame(issue_r1 = attr(readme.estimates$point_readme, "names"),
        predicted = readme.estimates$point_readme %>%
            as.vector(), truth = (table(germany_textpress$issue_r1[germany_textpress$cv_sample ==
            i])/sum(table(germany_textpress$issue_r1[germany_textpress$cv_sample ==
            i]))) %>%
            as.vector(), cv_sample = i) %>%
        rbind.fill(agg_eval)

}

# Difference in percentage points (positive values indicate an inflated
# prediction, i.e. we estimate a higher share for the category compared to the
# truth)
agg_eval$difference <- agg_eval$predicted - agg_eval$truth

agg_eval <- dplyr::mutate(agg_eval, predicted = round(predicted, 3), truth = round(truth,
    3), difference = round(difference, 3))
```

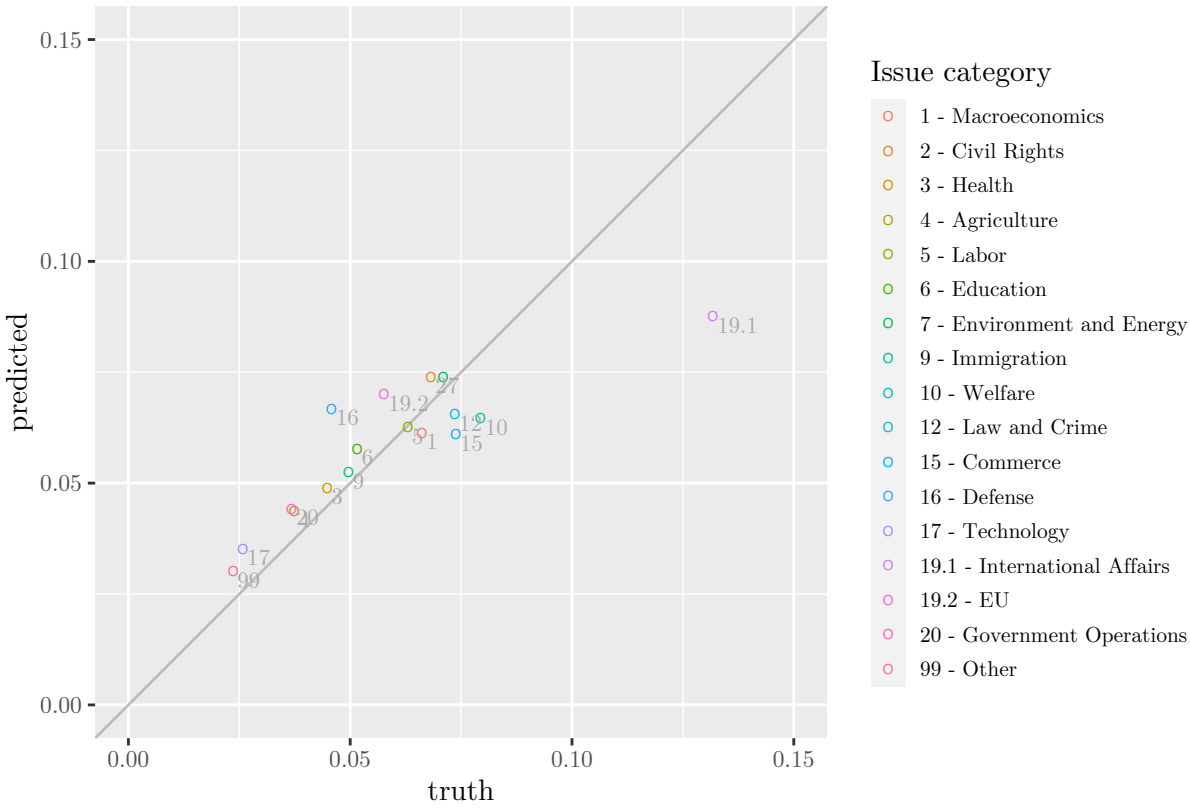
```r
# Change and order labels
agg_eval$issue_r1[agg_eval$issue_r1 == 191] <- 19.1
agg_eval$issue_r1[agg_eval$issue_r1 == 192] <- 19.2
agg_eval$issue_r1 <- as.factor(as.numeric(agg_eval$issue_r1))
levels(agg_eval$issue_r1) <- str_c(levels(agg_eval$issue_r1), " - ", issue_categories[c(1:13,
    16:17, 14:15), 2])

save(agg_eval, file = "readme-files/agg_eval.RData")

# Write latex table
if (!dir.exists("tables")) dir.create("tables")
latex_out <- capture.output(stargazer(agg_eval %>%
    dplyr::group_by(issue_r1) %>%
    dplyr::summarise(predicted = mean(predicted), truth = mean(truth)) %>%
    dplyr::rename(issue = issue_r1) %>%
    as.data.frame() %>%
    stargazer(summary = F, rownames = F, title = "Evaluation of aggregate values (readme2)",
        label = "tab:agg_eval_readme", notes = "Mean values from five-fold cross-validation."),
    out = "tables/agg_eval_readme.tex"))


# Plot aggregate evaluation
if (!dir.exists("plots")) dir.create("plots")

plot_agg_eval(agg_eval %>%
    dplyr::group_by(issue_r1) %>%
    dplyr::summarise(predicted = mean(predicted), truth = mean(truth)), "readme")
```




The plot shows the mean values from a five-fold cross-validation.

# 4 Readme2 applied

Now we turn to estimating proportions for the unlabeled documents. We do so by running the software for a subset of press releases from each quarter and party. All labeled documents are included in each estimation.

This allows us, to compare the values with those obtained from the "classify and count" method using supervised text classification.

```
if (!dir.exists("readme-files/readme-quarters")) dir.create("readme-files/readme-quarters")

# Loading full dataset (not on GitHub)
all_germany <- read_rds("data/data_joint.RDS") %>%
    select(c(header, text.x, date.x, issue, party.x, id)) %>%
    filter(!is.na(date.x) & !is.na(text.x) & (issue != "98 Non-thematic" | is.na(issue))) %>%
    dplyr::rename(party = party.x, date = date.x, text = text.x)
nrow(all_germany)

# Unite parties
all_germany$party <- all_germany$party %>%
    str_replace_all(c(union_fraktion = "CDU/CSU", spd_fraktion = "SPD", `90gruene_fraktion` = "B'90/Die
        fdp_bundesverband = "FDP", fdp_fraktion = "FDP", linke_fraktion = "DIE LINKE",
        afd_bundesverband = "AfD", afd_fraktion = "AfD"))

# Merge categories
all_germany$issue_r1 <- all_germany$issue %>%
    as.character %>%
    str_replace_all(c(a = "1", b = "2")) %>%
```

```
    str_extract("[:digit:]*") %>%
    as.numeric

# Make quarterly date
all_germany$date <- as.character(all_germany$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()

# Go through parties
issue_agendas_readme <- data.frame()
for (party in unique(all_germany$party)) {
    print(party)

    # Go through quarters
    for (date in unique(all_germany$date)) {
        print(date %>%
            as.Date(origin = "1970-01-01"))

        if (sum((all_germany$party == party & all_germany$date == date)) == 0)
            next  # Only   continue if there are documents for party/time

        if (file.exists(str_c("readme-files/readme-quarters/", party %>%
            str_replace_all("/", "-"), "_", date %>%
            as.Date(origin = "1970-01-01"), ".RData"))) {
            # Load if file exists
            load(str_c("readme-files/readme-quarters/", party %>%
                str_replace_all("/", "-"), "_", date %>%
                as.Date(origin = "1970-01-01"), ".RData"))
        } else {

            if (!("wordVec_summaries_all" %in% ls()))
                if (file.exists("readme-files/wordvectors/wordVec_summaries_all.RData")) {

                  load("readme-files/wordvectors/wordVec_summaries_all.RData")
                } else {

                  # Read raw vector data and create vector matrix
                  # (https://deepset.ai/german-word-embeddings)
                  if (!("glove_ger" %in% ls()))
                    if (file.exists("readme-files/wordvectors/glove_ger.RData")) {
                      load("readme-files/wordvectors/glove_ger.RData")
                    } else {
                      glove_ger_raw <- scan(file = "readme-files/wordvectors/GloVe-german.txt",
                        what = "", sep = "\n")
                      glove_ger <- proc_pretrained_vec(glove_ger_raw)
                      glove_feat <- names(glove_ger)
                      glove_ger <- glove_ger %>%
                        t()
                      save(glove_ger, file = "readme-files/wordvectors/glove_ger.RData")
```

```
                    }

                    ## Generate a word vector summary for all documents
                    wordVec_summaries_all <- undergrad(documentText = cleanme(str_c(all_germany$header,
                      " - ", all_germany$text)), wordVecs = glove_ger %>%
                      as.matrix)
                    save(wordVec_summaries, file = "readme-files/wordvectors/wordVec_summaries_all.RData")
                }

            if (!("wordVec_summaries_all" %in% ls())) {
                load("readme-files/wordvectors/wordVec_summaries_all.RData")
            }

            # Subset corpus to party/time and run readme
            readme_storage <- readme(dfm = wordVec_summaries_all[(all_germany$party ==
                party & all_germany$date == date) | !is.na(all_germany$issue_r1),
                ], labeledIndicator = ifelse(is.na(all_germany$issue_r1)[(all_germany$party ==
                party & all_germany$date == date) | !is.na(all_germany$issue_r1)],
                0, 1), categoryVec = all_germany$issue_r1[(all_germany$party == party &
                all_germany$date == date) | !is.na(all_germany$issue_r1)], nCores = 2,
                nCores_OnJob = 2) %>%
                suppressWarnings()

            # Save output to file
            save(readme_storage, file = str_c("readme-files/readme-quarters/", party %>%
                str_replace_all("/", "-"), "_", date %>%
                as.Date(origin = "1970-01-01"), ".RData"))
        }

        issue_agendas_readme <- data.frame(issue_r1 = attr(readme_storage$point_readme,
            "names"), attention = readme_storage$point_readme %>%
            as.vector(), date = date %>%
            as.Date(origin = "1970-01-01"), party = party) %>%
            rbind.fill(issue_agendas_readme)

    }
}

issue_agendas_readme <- merge(issue_agendas_readme, issue_categories, by = "issue_r1")
save(issue_agendas_readme, file = "readme-files/issue_agendas_readme.RData")
```

```
# Time needed to run script (much shorter when results are just loaded from a
# file)
print(Sys.time() - start_time)
```

```
## Time difference of 51.08608 secs
```

```
# Running readme for all parties/quarters took about 24h
```

```
```