Evaluation of textmodels

Cornelius Erfort

8/5/2021

Contents

1 Summary

We obtain the highest accuracy of 77.1% using the pretrained Transformers model GBERT fine-tuned with our labeled dataset. It requires ~8 minutes of computing time using 4 training epochs (remote, Colab Pro).

The Superlearner ensemble of single supervised classifiers obtains 71.2% accuracy (w/o cross-validation!), using the tfidf-transformed dfm. It requires ~18 minutes (w/o cross-validation!) of computing time (remote, Colab Pro).

The best performing single classifier (after five-fold cross-validation) is Ridge (L2) with an accuracy of 68.1%, using the tfidf-transformed dfm. It requires ~9 seconds of computing time (locally).

Our supervised text classification works better for some issue areas and worse for others.

The size of the training dataset affects the accuracy of our model. Our analysis suggests that an enlargement of our training dataset would only lead to marginal performance improvements. On the contrary, the Transformers model already yields an accuracy of over 70% with only about 750 coded press releases.

2 Setting up

This script requires the files which are not included on GitHub.

2.1 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.

```
start_time <- Sys.time()

packages <- c("quanteda", "quanteda.textmodels", "dplyr", "caret", "randomForest",
    "tm", "rmarkdown", "plyr", "readr", "ggplot2", "stringr", "formatR", "readstata13",
    "lubridate", "reticulate", "doMC", "glmnet", "kableExtra", "stargazer", "extrafont",
    "tidyr", "ggrepel")

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

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

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

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

if (!dir.exists("supervised-files")) dir.create("supervised-files")

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

2.2 Loading document frequency matrix (dfm)

See other script for the generation of the dfm.

```
load("supervised-files/data/dfmat.RData")
load("supervised-files/data/dfmat_alt.RData")

issue_categories <- data.frame(issue_r1 = c(1:7, 9:10, 12, 15:17, 20, 99, 191:192),
    issue_r1_descr = c("Macroeconomics", "Civil Rights", "Health", "Agriculture",
        "Labor", "Education", "Environment and Energy", "Immigration", "Welfare",
        "Law and Crime", "Commerce", "Defense", "Technology", "Government Operations",
        "Other", "International Affairs", "EU"))

# Distribution with merged categories
table(dfmat$issue_r1) %>%
    as.data.frame() %>%
    dplyr::rename(issue = Var1, n = Freq) %>%
    t() %>%
    kbl(booktabs = T) %>%
    kable_styling(latex_options = "scale_down")
```

issue	1	2	3	4	5	6	7	9	10	12	15	16	17	20	99	191	192
n	169	175	119	99	166	134	185	123	201	188	191	121	65	88	108	342	138

3 Textmodels

Following Barberá et al. (2021) we estimated the following models:

1. Naive Bayes (multinomial)
2. Ridge regression (L2)
3. Lasso regression (L1)
4. Elastic Net
5. SVM
6. Random Forest

(Barberá, P., Boydstun, A., Linn, S., McMahon, R., & Nagler, J. (2021). Automated Text Classification of News Articles: A Practical Guide. Political Analysis, 29(1), 19-42. doi:10.1017/pan.2020.8)

Additionally, we run the following models:

7. Superlearner (ensemble of single classifiers)
8. Semi-supervised (Elastic net)
9. Transformers model (BERT)

Finally, we also test the performance of Readme2 for the estimation of proportions. (An evaluation of the classification of individual documents is not possible for this method.)

The individual models are run in previous scripts.

An overview of the results for each classifier can be found in this script.

4 Aggregating the results of the classifiers

```
# Create a dataframe for all textmodels and save first row
tm_eval <- data.frame()

# 1 Naive bayes
load("supervised-files/textmodels/naivebayes_eval.RData")
(naivebayes_eval_aggr <- aggregate(cbind(accuracy, precision, recall, f1_score, time,
    seed) ~ weight + distribution + smooth + model, naivebayes_eval[, -c(1)], mean) %>%
    arrange(desc(accuracy)))[1:5, ] %>%
    mutate_at(vars(accuracy, precision, recall, f1_score), function(x) round(x, 3)) %>%
    kbl(booktabs = T)
```

weight	distribution	smooth	model	accuracy	precision	recall	f1_score	time	seed
tfidf	multinomial	1	Naive bayes	0.670	0.661	0.640	0.639	0.230	1621447882
tfidf	multinomial	2	Naive bayes	0.668	0.681	0.624	0.628	0.248	1621447882
tfidf	multinomial	3	Naive bayes	0.660	0.681	0.613	0.618	0.352	1621447882
uniform	multinomial	1	Naive bayes	0.641	0.627	0.625	0.618	0.658	1621447882
termfreq	multinomial	1	Naive bayes	0.638	0.624	0.618	0.612	0.238	1621447882

```
naivebayes_eval_aggr$time <- naivebayes_eval_aggr$time/60
tm_eval <- naivebayes_eval_aggr %>%
    rbind.fill(tm_eval)

# 2 Ridge regression (L2)
```

```
load("supervised-files/textmodels/ridge_eval.RData")
(ridge_eval_aggr <- aggregate(cbind(accuracy, precision, recall, f1_score, time,
    seed) ~ weight + type + model, ridge_eval[, -c(1)], mean) %>%
    arrange(desc(accuracy)))[1:5, ] %>%
    mutate_at(vars(accuracy, precision, recall, f1_score), function(x) round(x, 3)) %>%
    kbl(booktabs = T)
```

weight	type	model	accuracy	precision	recall	f1_score	time	seed
tfidf	7	Ridge (L2)	0.681	0.684	0.655	0.661	8.9500	1621447882
tfidf	0	Ridge (L2)	0.680	0.682	0.654	0.659	10.7580	1621447882
uniform	0	Ridge (L2)	0.608	0.606	0.584	0.588	5.0660	1621447882
uniform	7	Ridge (L2)	0.606	0.606	0.582	0.586	3.4040	1621447882
termfreq	0	Ridge (L2)	0.587	0.634	0.542	0.550	1.7725	1621447882

```
ridge_eval_aggr$time <- ridge_eval_aggr$time/60
tm_eval <- ridge_eval_aggr %>%
    rbind.fill(tm_eval)

# 3 Lasso regression (L1)
load("supervised-files/textmodels/lasso_eval.RData")
(lasso_eval_aggr <- aggregate(cbind(accuracy, precision, recall, f1_score, time,
    seed) ~ weight + model, lasso_eval[, -c(1)], mean) %>%
    arrange(desc(accuracy)))[1:2, ] %>%
    mutate_at(vars(accuracy, precision, recall, f1_score), function(x) round(x, 3)) %>%
    kbl(booktabs = T)
```

weight	model	accuracy	precision	recall	f1_score	time	seed
tfidf	Lasso (L1)	0.626	0.624	0.600	0.604	12.728	1621447882
uniform	Lasso (L1)	0.583	0.572	0.555	0.557	3.278	1621447882

```
lasso_eval_aggr$time <- lasso_eval_aggr$time/60
tm_eval <- lasso_eval_aggr %>%
    rbind.fill(tm_eval)

# 4 Elastic net
load("supervised-files/textmodels/elasticnet_opt_alt.RData")
load("supervised-files/textmodels/elasticnet_mod.RData")
load("supervised-files/textmodels/elasticnet_pred.RData")
elastic_net_test <- dfm_subset(dfmat, dfmat$cv_sample == 1)

# If alternative model is better:
if (elasticnet_opt_alt) {
    load("supervised-files/textmodels/elasticnet_mod_alt.RData")
    elasticnet_mod <- elasticnet_mod_alt
    load("supervised-files/textmodels/elasticnet_pred_alt.RData")
    elasticnet_pred <- elasticnet_pred_alt
    elastic_net_test <- dfm_subset(dfmat_alt, dfmat_alt$cv_sample == 1)
}


tm_eval <- data.frame(accuracy = accuracy(elasticnet_pred, elastic_net_test$issue_r1),
    precision = precision(elasticnet_pred, elastic_net_test$issue_r1) %>%
```

```
        unlist() %>%
        mean(), recall = recall(elasticnet_pred, elastic_net_test$issue_r1) %>%
        unlist() %>%
        mean(), f1_score = f1_score(elasticnet_pred, elastic_net_test$issue_r1) %>%
        unlist() %>%
        mean(), time = elasticnet_mod$time, seed = elasticnet_mod$seed, weight = elasticnet_mod$weight,
    model = elasticnet_mod$model, alpha = 0.5, distribution = "multinomial") %>%
    rbind.fill(tm_eval)

# 5 SVM
load("supervised-files/textmodels/svm_eval.RData")
(svm_eval_aggr <- aggregate(cbind(accuracy, precision, recall, f1_score, time, seed) ~
    weight + model, svm_eval[, -c(1)], mean) %>%
    arrange(desc(accuracy)))[1:3, ] %>%
    mutate_at(vars(accuracy, precision, recall, f1_score), function(x) round(x, 3)) %>%
    kbl(booktabs = T)
```

weight	model	accuracy	precision	recall	f1_score	time	seed
tfidf	SVM	0.663	0.656	0.641	0.642	8.808	1621447882
termfreq	SVM	0.576	0.569	0.553	0.555	1.016	1621447882
docfreq	SVM	0.575	0.569	0.552	0.554	1.888	1621447882

```
svm_eval_aggr$time <- svm_eval_aggr$time/60
tm_eval <- svm_eval_aggr %>%
    rbind.fill(tm_eval)

# 6 Random Forest
load("supervised-files/textmodels/randomforest_opt_alt.RData")
load("supervised-files/textmodels/randomforest_eval.RData")
load("supervised-files/textmodels/randomforest_pred.RData")
randomforest_test <- dfm_subset(dfmat, dfmat$cv_sample == 1)

# If alternative model is better:
if (randomforest_opt_alt) {
    load("supervised-files/textmodels/randomforest_eval_alt.RData")
    randomforest_mod <- randomforest_eval_alt
    load("supervised-files/textmodels/randomforest_pred_alt.RData")
    randomforest_pred <- randomforest_pred_alt
    randomforest_test <- dfm_subset(dfmat_alt, dfmat_alt$cv_sample == 1)
}

tm_eval <- data.frame(accuracy = accuracy(randomforest_pred, randomforest_test$issue_r1),
    precision = precision(randomforest_pred, randomforest_test$issue_r1) %>%
        unlist() %>%
        mean(na.rm = T), recall = recall(randomforest_pred, randomforest_test$issue_r1) %>%
        unlist() %>%
        mean(na.rm = T), f1_score = f1_score(randomforest_pred, randomforest_test$issue_r1) %>%
        unlist() %>%
        mean(na.rm = T), time = randomforest_eval$time, seed = randomforest_eval$seed,
    model = randomforest_eval$model, weight = randomforest_eval$weight, alpha = 0.5,
    distribution = "multinomial") %>%
    rbind.fill(tm_eval)
```

```r
# 7 SuperLearner Load superlearner prediction and add row to evaluation table
# super_pred <- read_csv('superlearner-files/super-pred.csv', col_names = F)[[1]]
# %>% as.numeric() names(super_pred) <- c('prediction', 'issue_r1', 'cv_sample')
# supertime <- read_csv('superlearner-files/cv-time.txt', col_names = F,
# col_types = 'n')[1,1] %>% as.numeric() superlearner_test <-
# dfm_subset(dfmat_alt, dfmat$cv_sample == 1)$issue_r1 tm_eval <-
# data.frame(accuracy = accuracy(super_pred$prediction, super_pred$issue_r1),
# precision = precision(super_pred$prediction, super_pred$issue_r1) %>% unlist()
# %>% mean(), recall = recall(super_pred$prediction, super_pred$issue_r1) %>%
# unlist() %>% mean(), f1_score = f1_score(super_pred$prediction,
# super_pred$issue_r1) %>% unlist() %>% mean(), time = supertime, weight =
# 'tfidf', seed = 1621447882, model = 'SuperLearner ensemble') %>%
# rbind.fill(tm_eval)

tm_eval <- data.frame(accuracy = 0.712, time = 18, weight = "tfidf", seed = 1621447882,
    model = "SuperLearner ensemble") %>%
    rbind.fill(tm_eval)

# 8 Semi-supervised Load Semi-supervised prediction and add row to evaluation
# table
semi_pred <- read_csv("semi-files/semi-pred.csv", col_names = F)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double(),
##   X2 = col_double(),
##   X3 = col_double()
## )
```

```r
names(semi_pred) <- c("prediction", "issue_r1", "cv_sample")
semitime <- read_csv("semi-files/cv-time.txt", col_names = F, col_types = "n") %>%
    as.numeric()/60

tm_eval <- data.frame(accuracy = accuracy(semi_pred$prediction, semi_pred$issue_r1),
    precision = precision(semi_pred$prediction, semi_pred$issue_r1) %>%
        unlist() %>%
        mean(), recall = recall(semi_pred$prediction, semi_pred$issue_r1) %>%
        unlist() %>%
        mean(), f1_score = f1_score(semi_pred$prediction, semi_pred$issue_r1) %>%
        unlist() %>%
        mean(), time = semitime, weight = "tfidf", seed = 1621447882, model = "Semi-supervised") %>%
    rbind.fill(tm_eval)


# 9 Transformers Load transfer models prediction and add row to evaluation table
transfer_pred <- read_csv("transfer-files/bert-pred.csv", col_names = F)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double(),
##   X2 = col_double(),
##   X3 = col_double(),
```

```
##   X4 = col_double()
## )
```

```
names(transfer_pred) <- c("prediction", "issue_r1", "label", "cv_sample")
transfertime <- read_csv("transfer-files/cv-time.txt", col_names = F, col_types = "n") %>%
    as.numeric()/60

tm_eval <- data.frame(accuracy = accuracy(transfer_pred$prediction, transfer_pred$label),
    precision = precision(transfer_pred$prediction, transfer_pred$label) %>%
        unlist() %>%
        mean(), recall = recall(transfer_pred$prediction, transfer_pred$label) %>%
        unlist() %>%
        mean(), f1_score = f1_score(transfer_pred$prediction, transfer_pred$label) %>%
        unlist() %>%
        mean(), time = transfertime, seed = 1621447882, model = "GBERT (Transformers)") %>%
    rbind.fill(tm_eval)

# Add var for instance type
tm_eval <- mutate(tm_eval, instance = ifelse(str_detect(model, "(GBER)|(Super)|(Semi)"),
    "remote", "local"))
```

```
aggregate(accuracy ~ weight, tm_eval, mean)
```

```
##     weight  accuracy
## 1  docfreq 0.5977859
## 2 termfreq 0.5980868
## 3    tfidf 0.6697921
## 4  uniform 0.6066034
```

The tfidf-transformed dfm yields a higher performance for most classifiers and parameter settings.

5 Evaluation of textmodels

In this section, we present a table for comparison of our textmodels. We compare the model with the highest accuracy for each classifier.

```
# Only keep best model for each classifier
tm_eval <- tm_eval %>%
    dplyr::group_by(model) %>%
    dplyr::mutate(acc_rank = order(order(accuracy, decreasing = TRUE))) %>%
    filter(acc_rank == 1) %>%
    select(-c(acc_rank))

# Comparison of textmodels
tm_eval[, c("accuracy", "precision", "recall", "f1_score", "time")] <- apply(tm_eval[,
    c("accuracy", "precision", "recall", "f1_score", "time")], MARGIN = 2, function(x) round(x,
    3))

tm_eval[order(tm_eval$accuracy, decreasing = T), c("model", "weight", "accuracy",
    "precision", "recall", "f1_score", "time")] %>%
    kbl(booktabs = T, row.names = F)
```

model	weight	accuracy	precision	recall	f1_score	time
GBERT (Transformers)	NA	0.775	0.771	0.763	0.766	8.589
SuperLearner ensemble	tfidf	0.712	NA	NA	NA	18.000
Ridge (L2)	tfidf	0.681	0.684	0.655	0.661	0.149
Naive bayes	tfidf	0.670	0.661	0.640	0.639	0.004
Semi-supervised	tfidf	0.668	0.704	0.623	0.644	0.711
SVM	tfidf	0.663	0.656	0.641	0.642	0.147
Elastic net	uniform	0.634	0.650	0.590	0.596	21.772
Lasso (L1)	tfidf	0.626	0.624	0.600	0.604	0.212
Random forest	uniform	0.621	0.662	0.560	0.608	23.682

```
latex_out <- capture.output(tm_eval[order(tm_eval$accuracy, decreasing = T), c("model",
    "weight", "accuracy", "precision", "recall", "f1_score", "time")] %>%
    dplyr::rename(Model = model, DFM = weight, Accuracy = accuracy, Precision = precision,
        Recall = recall, `F1 score` = f1_score, `Time (min)` = time) %>%
    stargazer(out = "tables/tm-eval.tex", summary = F, rownames = F, title = "Summary of classifier peri
        label = "tab:tm-eval"))
```

We obtain the highest accuracy of 77.5% using the pretrained Transformers model GBERT fine-tuned with our labeled dataset. It requires ~8 minutes of computing time (remote, Colab Pro).

The Superlearner ensemble of single supervised classifiers obtains 71.2% accuracy, using the tfidf-transformed dfm. It requires ~18 minutes of computing time (remote, Colab Pro).

The best performing single classifier (after five-fold cross-validation) is Ridge (L2) with an accuracy of 68.1%, using the tfidf-transformed dfm. It requires ~9 seconds of computing time (locally).

The Naive bayes classifier obtains 67.0% accuracy in only 0.23 seconds (locally).

In the following, we therefore use this model to evaluate the performance of the supervised text classification.

6 Confusion matrix

Since some issue areas might be more suitable for an supervised classification than others, we analyze the accuracy for individual issues and present a confusion matrix.

To do so, we rely on the top-performing single classifier, Ridge (L2), as well as the fine-tuned Transformers model.

6.1 Ridge (L2)

```
# Baseline model (highest cross-validated accuracy)
ridge_pred <- data.frame()
for (i in 1:5) {
    print(i)
    ridge_pred <- data.frame(prediction = textmodel_svm(dfm_subset(dfmat_alt, dfmat_alt$cv_sample !=
        i), dfm_subset(dfmat_alt, dfmat_alt$cv_sample != i)$issue_r1, type = 7) %>%
        predict(., newdata = dfm_subset(dfmat_alt, dfmat_alt$cv_sample == i)), issue_r1 = dfm_subset(df
        dfmat_alt$cv_sample == i)$issue_r1) %>%
        rbind.fill(ridge_pred)
}
```

```
## [1] 1
```

```
## Warning: 3 features in newdata not used in prediction.
```

```
## [1] 2
```

```
## Warning: 2 features in newdata not used in prediction.
```

```
## [1] 3
## [1] 4
```

```
## Warning: 1 feature in newdata not used in prediction.
```

```
## [1] 5
```

```
## Warning: 2 features in newdata not used in prediction.
```

```
# Confusion matrix and overall statistics
ridge_pred$issue_r1 %>%
    table
```

```
## .
##   1   2   3   4   5   6   7   9  10  12  15  16  17  20  99 191 192
## 169 175 119  99 166 134 185 123 201 188 191 121  65  88 108 342 138
```

```
truth <- ridge_pred$issue_r1
truth[truth == 191] <- 19.1
truth[truth == 192] <- 19.2
truth <- factor(truth, levels = c(1:7, 9:10, 12, 15:17, 19.1, 19.2, 20, 99))

prediction <- ridge_pred$prediction
prediction[prediction == 191] <- 19.1
prediction[prediction == 192] <- 19.2
prediction <- factor(prediction, levels = c(1:7, 9:10, 12, 15:17, 19.1, 19.2, 20,
    99))

ridge_pred <- data.frame(truth, prediction, model = "Ridge (L2)")

(table(prediction, truth) %>%
    confusionMatrix(mode = "sens_spec", dnn = c("predicted", "truth")))$table
```

```
##           truth
## prediction   1   2   3   4   5   6   7   9  10  12  15  16  17 19.1 19.2  20
##       1    111   0   0   0  11   2   2   1   4   3   8   0   1    2    4  10
##       2      0  94  11   1   4   2   2   5   5  15   5   1   4    9    1   9
##       3      0   5  89   1   3   0   0   0   2   4   1   0   0    0    0   0
##       4      0   0   5  77   0   0   7   0   1   0   7   0   0    1    0   0
##       5     20   0   1   0 123   1   0   3  12   1   1   0   1    0    0   1
##       6      1   4   1   0   1 109   0   4   4   1   3   0   4    0    0   2
##       7      5   2   1   4   0   1 136   0  14   1   7   0   5    9    0   0
##       9      1   3   0   0   4   0   0  92   2   6   1   0   0    1    3   1
##       10     4  10   3   2   8   9  12   1 136   5  11   1   4    1    2  10
##       12     2  27   5   0   2   1   1   4   4 125   8   6   2    4    3  11
##       15    10   3   1  12   4   1  11   2   6   8 102   2  14    8   14   4
##       16     0   3   0   0   0   0   0   0   0   6   1  69   1   14    0   4
##       17     2   1   0   1   0   1   0   0   1   3   2   0  25    0    0   1
##       19.1   4  16   2   1   3   1  11   7   4   5  13  35   1  279   28   4
##       19.2   3   0   0   0   2   0   1   1   1   2  15   3   0    8   79   2
##       20     3   5   0   0   0   1   2   1   4   2   1   3   0    1    1  27
##       99     3   2   0   0   1   5   0   2   1   1   5   1   3    5    3   2
##           truth
## prediction  99
```

```
##       1      5
##       2      4
##       3      1
##       4      0
##       5      5
##       6      1
##       7      0
##       9      1
##       10     1
##       12     6
##       15     4
##       16     2
##       17     5
##       19.1   9
##       19.2   2
##       20     5
##       99    57
```

```
latex_out <- capture.output(as.data.frame(unclass((table(prediction, truth) %>%
    confusionMatrix(mode = "sens_spec", dnn = c("predicted", "truth")))$table)) %>%
    stargazer(summary = F, title = "Confusion matrix for the test data (Ridge L2)",
        label = "tab:confusion-mat-ridge"))

latex_out <- capture.output(latex_out %>%
    str_replace_all("tabular", "tabularx") %>%
    str_replace_all("\\@\\{\\\\extracolsep\\{5pt\\}\\} ccccccccccccccccc", "\\\\textwidth\\}\\{XXXXXXX
    cat(sep = "\n"), file = "tables/confusion-mat-ridge.tex")

issue_eval_ridge <- data.frame(truth = truth, prediction = prediction) %>%
    group_by(truth) %>%
    dplyr::mutate(accuracy = accuracy(truth, prediction)) %>%
    select(-c("prediction")) %>%
    unique() %>%
    merge(., data.frame(truth = levels(truth), precision = precision(truth, prediction),
        recall = recall(truth, prediction), f1_score = f1_score(truth, prediction)),
        by = "truth") %>%
    dplyr::rename(Accuracy = accuracy, Precision = precision, Recall = recall, `F1 score` = f1,
        Issue = truth)
issue_eval_ridge[, c("Accuracy", "Precision", "Recall", "F1 score")] <- issue_eval_ridge[,
    c("Accuracy", "Precision", "Recall", "F1 score")] %>%
    apply(., MARGIN = 2, FUN = function(x) round(as.numeric(x), 3))

# Change and order labels
issue_eval_ridge <- issue_eval_ridge[order(issue_eval_ridge$Issue), ]
issue_eval_ridge$Issue <- str_c(as.character(issue_eval_ridge$Issue), " - ", issue_categories[c(1:13,
    16:17, 14:15), 2])

# Accuracy
accuracy(truth, prediction)
```

```
## $accuracy
## [1] 0.6623277
```

6.2 Transformers (GBERT)

```
# Import results
transfer_pred <- read_csv("transfer-files/bert-pred.csv", col_names = F)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double(),
##   X2 = col_double(),
##   X3 = col_double(),
##   X4 = col_double()
## )
```

```
names(transfer_pred) <- c("prediction_label", "issue_r1", "label", "cv_sample")
labels <- select(transfer_pred, c(label, issue_r1)) %>%
    unique %>%
    dplyr::rename(prediction = issue_r1)
transfer_pred <- merge(transfer_pred, labels, by.x = "prediction_label", by.y = "label") %>%
    select(-c(prediction_label))

# Confusion matrix and overall statistics
transfer_pred$issue_r1 %>%
    table
```

```
## .
##   1   2   3   4   5   6   7   9  10  12  15  16  17  20  99 191 192
## 169 175 119  99 166 134 185 123 201 188 191 121  65  88 108 342 138
```

```
truth <- transfer_pred$issue_r1
truth[truth == 191] <- 19.1
truth[truth == 192] <- 19.2
truth <- factor(truth, levels = c(1:7, 9:10, 12, 15:17, 19.1, 19.2, 20, 99))

prediction <- transfer_pred$prediction
prediction[prediction == 191] <- 19.1
prediction[prediction == 192] <- 19.2
prediction <- factor(prediction, levels = c(1:7, 9:10, 12, 15:17, 19.1, 19.2, 20,
    99))

transfer_pred <- data.frame(truth, prediction, model = "Transformers")

(table(prediction, truth) %>%
    confusionMatrix(mode = "sens_spec", dnn = c("predicted", "truth")))$table
```

```
##           truth
## prediction   1   2   3   4   5   6   7   9  10  12  15  16  17 19.1 19.2  20
##       1    131   0   1   0  10   1   1   0   2   2  11   0   1    1    3   8
##       2      0 112   2   0   1   3   1   2   5  14   4   2   1    8    0   4
##       3      2   6  98   3   1   0   0   0   2   1   0   0   0    1    0   0
##       4      0   0   5  90   0   0   4   0   1   0   3   0   0    1    0   0
##       5      8   1   4   0 137   4   0   1   9   0   1   0   0    0    1   1
##       6      0   2   0   0   1 110   0   0   6   0   1   0   2    0    0   0
##       7      3   0   1   2   0   0 164   0   8   0   1   0   1   10    0   0
##       9      0   6   0   0   0   0   0 114   1   3   1   1   0    1    3   1
##       10     2   4   2   0  10   7   8   0 159   4   4   0   0    0    2   7
```

```
##       12     0  18   4   0   1   0   0   2   2 146   5   6   0    4    1  10
##       15     6   5   0   2   0   0   2   0   3   4 135   0   8    7    9   4
##       16     0   1   0   0   0   1   0   0   0   6   1  93   1   16    0   5
##       17     3   2   0   0   0   3   0   0   0   0   4   0  46    0    0   0
##       19.1   2   9   2   2   2   0   5   2   0   3   9  14   2  282   18   1
##       19.2   6   1   0   0   1   0   0   1   0   2   9   1   0    7   99   2
##       20     2   4   0   0   0   0   0   0   3   3   0   4   0    0    1  37
##       99     4   4   0   0   2   5   0   1   0   0   2   0   3    4    1   8
##           truth
## prediction  99
##       1      2
##       2      1
##       3      1
##       4      0
##       5      2
##       6      1
##       7      0
##       9      0
##       10     1
##       12     3
##       15     3
##       16     1
##       17     2
##       19.1   9
##       19.2   5
##       20     7
##       99    70
```

```
latex_out <- capture.output(as.data.frame(unclass((table(prediction, truth) %>%
    confusionMatrix(mode = "sens_spec", dnn = c("predicted", "truth")))$table)) %>%
    stargazer(summary = F, title = "Confusion matrix for the test data (Transformers)",
        label = "tab:confusion-mat-transfer"))

latex_out <- capture.output(latex_out %>%
    str_replace_all("tabular", "tabularx") %>%
    str_replace_all("\\@\\{\\\\extracolsep\\{5pt\\}\\} ccccccccccccccccc", "\\\\textwidth\\}\\{XXXXXXX
    cat(sep = "\n"), file = "tables/confusion-mat-transfer")

issue_eval_transfer <- data.frame(truth = truth, prediction = prediction) %>%
    group_by(truth) %>%
    dplyr::mutate(accuracy = accuracy(truth, prediction)) %>%
    select(-c("prediction")) %>%
    unique() %>%
    merge(., data.frame(truth = levels(truth), precision = precision(truth, prediction),
        recall = recall(truth, prediction), f1_score = f1_score(truth, prediction)),
        by = "truth") %>%
    dplyr::rename(Accuracy = accuracy, Precision = precision, Recall = recall, `F1 score` = f1,
        Issue = truth)

issue_eval_transfer[, c("Accuracy", "Precision", "Recall", "F1 score")] <- issue_eval_transfer[,
    c("Accuracy", "Precision", "Recall", "F1 score")] %>%
    apply(., MARGIN = 2, FUN = function(x) round(as.numeric(x), 3))

# Order and add names
```

```
issue_eval_transfer <- issue_eval_transfer[order(issue_eval_transfer$Issue), ]
issue_eval_transfer$Issue <- str_c(as.character(issue_eval_transfer$Issue), " - ",
    issue_categories[c(1:13, 16:17, 14:15), 2])

# Write latex table
if (!dir.exists("tables")) dir.create("tables")

# Accuracy
accuracy(truth, prediction)
```

```
## $accuracy
## [1] 0.7745023
```

```
# Merge with issue_eval_ridge
issue_eval_ridge_transfer <- merge(select(issue_eval_ridge, c(Issue, Accuracy)) %>%
    dplyr::rename(Ridge = Accuracy), select(issue_eval_transfer, c(Issue, Accuracy)) %>%
    dplyr::rename(Transformers = Accuracy), by = "Issue")
issue_eval_ridge_transfer <- issue_eval_ridge_transfer[order(issue_eval_ridge_transfer$Issue %>%
    str_extract("^[:digit:]\\.?[:digit:]?") %>%
    as.numeric()), ]

latex_out <- capture.output(issue_eval_ridge_transfer %>%
    stargazer(out = "tables/issue-eval-ridge-transfer.tex", summary = F, rownames = F,
        title = "Performance statistics by issue for different models", label = "tab:issue_eval_transfe

accuracy(truth[!(truth %in% c(2, 20, 99))], prediction[!(truth %in% c(2, 20, 99))])
```

```
## $accuracy
## [1] 0.8049978
```

Regarding the issues, the classifiers works better for specific issue categories.

While some have a higher than average sensitivity (e.g. 4 - Agriculture, 7 - Env. & Energy, 9 - Immigration, 191 - Int. Affairs), others fare worse than average (e.g. 2- Civil Rights, 20 - Gov. Ops., 99 - Other). Unsurprisingly, these rather unspecific categories are difficult to predict. Without them, the accuracy of the Transformers model rises to above 80%.

The better sensitivity for other categories is likely the result of a more specific use of words. Category 17 - Technology may feature a worse accuracy (especially for Ridge) because it is underrepresented in the labeled data. Category 15 - Commerce is often misclassified as 191 - Int. Affairs and 192 - EU: Categories where a press release may contain similar words.

16 - Defense is often misclassified as 191 - International Affairs.

7 Accuracy of predicted proportions

Since we are most interested in the prediction of proportions (i.e. how many press releases were dedicated to issue X in a given time frame), we analyze the accuracy of predicted proportions.

To do so, we rely on the top-performing single classifier, Ridge (L2), and perform a five-fold cross-validation.

We find the greatest average difference in predicted and actual proportions for 19.1 - International Affairs with 4.8%. And the lowest for 6 - Education wit 0.4%.

The MSE is 1.33%.

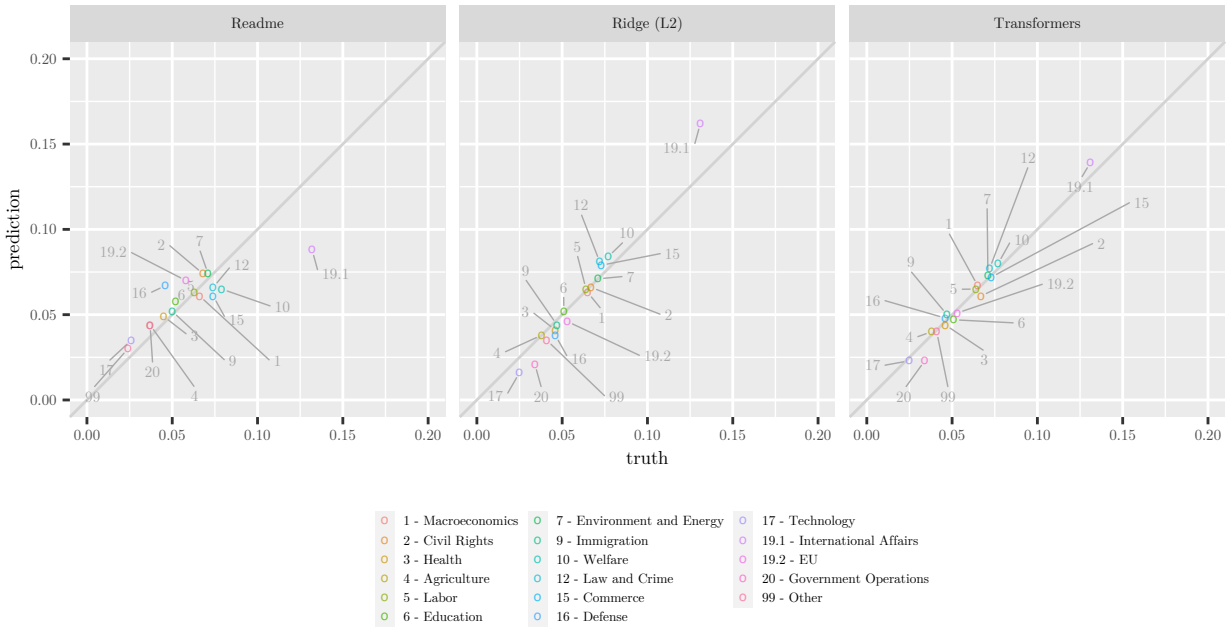
```
# Load
load("readme-files/agg_eval.RData")
```

```r
readme_agg <- agg_eval

# Aggregate Accuracy of predicted proportions (five-fold cross-validation)

# Ridge
ridge_agg <- merge(table(ridge_pred$truth)/nrow(ridge_pred), table(ridge_pred$prediction)/nrow(ridge_pr
    by = "Var1") %>%
    dplyr::rename(Issue = Var1, truth = Freq.x, prediction = Freq.y) %>%
    dplyr::mutate(model = "Ridge (L2)")
ridge_agg <- ridge_agg[order(ridge_agg$Issue), ]
ridge_agg$Issue <- str_c(as.character(ridge_agg$Issue), " - ", issue_categories[c(1:13,
    16:17, 14:15), 2]) %>%
    factor()

# Transformers
transfer_agg <- merge(table(transfer_pred$truth)/nrow(transfer_pred), table(transfer_pred$prediction)/n
    by = "Var1") %>%
    dplyr::rename(Issue = Var1, truth = Freq.x, prediction = Freq.y) %>%
    dplyr::mutate(model = "Transformers")
transfer_agg <- transfer_agg[order(transfer_agg$Issue), ]
transfer_agg$Issue <- str_c(as.character(transfer_agg$Issue), " - ", issue_categories[c(1:13,
    16:17, 14:15), 2]) %>%
    factor()

# Readme (update!)
readme_agg <- readme_agg %>%
    dplyr::rename(Issue = issue_r1, prediction = predicted) %>%
    select(-c(cv_sample, difference)) %>%
    dplyr::group_by(Issue) %>%
    dplyr::mutate(prediction = mean(prediction), truth = mean(truth), model = "Readme") %>%
    unique()

three_agg <- rbind.fill(readme_agg, transfer_agg, ridge_agg)

# Difference in percentage points (positive values indicate an inflated
# prediction, i.e. we estimate a higher share for the category compared to the
# truth)
three_agg$difference <- three_agg$prediction - three_agg$truth
three_agg[, c("truth", "prediction", "difference")] <- apply(three_agg[, c("truth",
    "prediction", "difference")], MARGIN = 2, function(x) round(x, 3))  # Round

# MSE
print(sum((three_agg$difference * 100)^2)/nrow(three_agg))  # 0.899
```

```
## [1] 1.014118
```

```r
# Plot Plotting aggregate evaluation in one plot
ggplot(three_agg, aes(x = truth, y = prediction)) + geom_abline(slope = 1, color = "light grey") +
    geom_text_repel(label = three_agg$Issue %>%
        str_extract("[:digit:]{1,2}(\\.[:digit:])?"), box.padding = 0.4, color = "dark grey",
        size = 2, family = "LM Roman 10", segment.size = 0.25, min.segment.length = 0.1,
        point.padding = 0.15, max.overlaps = 100) + geom_point(shape = "o", aes(color = Issue),
    alpha = 0.75) + ylim(c(0, 0.2)) + xlim(c(0, 0.2)) + guides(color = guide_legend(ncol = 3)) +
    labs(color = "Issue category") + # caption = 'The plot shows predicted proportions from five folds.
```

```
theme(legend.position = "bottom", legend.title = element_blank(), aspect.ratio = 1,
    text = element_text(size = 7), legend.key.size = unit(0.5, "line"), legend.text = element_text(size
    facet_wrap(~model) + ggsave("plots/agg_eval_compare_facet.pdf", device = cairo_pdf,
    width = 3 * 2^0.5, height = 3) + ggsave("plots/agg_eval_compare_facet.png", width = 3 *
    2^0.5, height = 3)
```




```
# Write table
three_agg <- three_agg %>%
    select(c(Issue, difference, model)) %>%
    pivot_wider(id_cols = "Issue", names_from = "model", values_from = "difference")

three_agg %>%
    kbl(booktabs = T, row.names = F)
```

Issue	Readme	Transformers	Ridge (L2)
1 - Macroeconomics	-0.005	0.002	-0.002
2 - Civil Rights	0.006	-0.006	-0.001
3 - Health	0.004	-0.002	-0.005
4 - Agriculture	0.006	0.002	0.000
5 - Labor	0.000	0.001	0.001
6 - Education	0.006	-0.004	0.000
7 - Environment and Energy	0.003	0.002	0.000
9 - Immigration	0.003	0.003	-0.003
10 - Welfare	-0.015	0.003	0.007
12 - Law and Crime	-0.008	0.005	0.009
15 - Commerce	-0.013	-0.001	0.006
16 - Defense	0.021	0.002	-0.008
17 - Technology	0.009	-0.002	-0.009
20 - Government Operations	0.007	-0.010	-0.012
99 - Other	0.007	-0.002	-0.007
19.1 - International Affairs	-0.044	0.008	0.031
19.2 - EU	0.012	-0.002	-0.007

```
latex_out <- capture.output(three_agg %>%
    stargazer(out = "tables/aggregated-eval.tex", summary = F, rownames = F, title = "Difference betwee
        label = "tab:aggregated-eval"))
```

8 Key features for the categories

```
# Ridge (L2) with full labeled data
tm_ridge <- textmodel_svm(dfmat_alt, dfmat_alt$issue_r1, type = 7)

# Key words/features for classified topics
key_feats <- data.frame()
for (i in rownames(tm_ridge$weights)) key_feats <- data.frame(issue_r1 = i) %>%
    cbind(sort(tm_ridge$weights[which(rownames(tm_ridge$weights) == i), ], decreasing = T)[1:5] %>%
        attr("names") %>%
        str_replace_all("\\_", " ") %>%
        t) %>%
    rbind.fill(key_feats)

key_feats <- key_feats[order(key_feats$issue_r1 %>%
    str_replace_all(c(`191` = "19.1", `192` = "19.2")) %>%
    as.numeric), ]

key_feats$issue_r1 <- levels(readme_agg$issue_r1)
```

```
## Warning: Unknown or uninitialised column: `issue_r1`.
```

```
key_feats %>%
    kbl(booktabs = T, row.names = F)
```

1	2	3	4	5
investitionen	haushalt	steuern	einnahmen	vermögen
frauen	pegida	menschen mit	datenschutz	gleichstellung
pflege	patienten	prävention	versicherten	rösler
landwirtschaft	tierschutz	aigner	glyphosat	agrarpolitik
mindestlohn	rent	beschäftigten	beschäftigt	beschäftigung
hochschulen	bildung	schulen	bafö	studierenden
energiewend	klimaschutz	energien	gorleben	umweltausschuss
flüchtling	integr	zuwanderung	migranten	nach deutschland
betreuungsgeld	hartz	verkehrspolitisch	familien	elterngeld
verfassungsschutz	straftaten	vorratsdatenspeicherung	kriminalität	täter
bank	tourismus	ttip	banken	finanztransaktionssteu
bundeswehr	waffen	nato	soldaten	rüstungsexport
digital	digitalisierung	internet	sender	rundfunk
russland	iran	israel	wahlen	menschenrecht
ezb	griechenland	ungarn	mitgliedstaaten	zypern
wowereit	kommunen	steinbach	öffentlichkeit	berlin
kultur	tod	bundesvorstand	wahl	trauern

```
latex_out <- capture.output(stargazer(key_feats, summary = F, title = "Key features for each issue (Ridg
    label = "tab:key_feats", out = "tables/key_feats.tex"))
```

9 Training data size and accuracy

To test the influence of the size of the training dataset on the accuracy, we estimate models with smaller sizes.

To do so, we rely on the top-performing single classifier, Ridge (L2), as well as the fine-tuned Transformers model.

The size of the training dataset affects the accuracy of our model. Our analysis suggests that an enlargement of our training dataset would only lead to marginal performance improvements. On the contrary, the Transformers model already yields an accuracy of over 70% with only about 750 coded press releases.

```
trainsize_ridge <- c(5, 25, 50, 100, 200, 300, 400, 500, 600, 700, 1000, 1500, 1750,
    2000, 2095)
trainsize_ridge <- data.frame(Model = "Ridge (L2)", train_size = trainsize_ridge)

trainsize_ridge$accuracy <- sapply(trainsize_ridge$train_size, FUN = function(x) {
    sub_train <- sample(1:ndoc(dfm_subset(dfmat_alt, dfmat_alt$cv_sample != 5)),
        x)
    ridge_opt_mod <- textmodel_svm(dfm_subset(dfmat_alt, dfmat_alt$cv_sample != 5)[sub_train,
        ], dfm_subset(dfmat_alt, dfmat_alt$cv_sample != 5)$issue_r1[sub_train], type = 7)
    accuracy(dfm_subset(dfmat_alt, dfmat_alt$cv_sample == 5)$issue_r1, predict(ridge_opt_mod,
        newdata = dfm_subset(dfmat_alt, dfmat_alt$cv_sample == 5)))
}) %>%
    unlist
```

```
## Warning: 23986 features in newdata not used in prediction.

## Warning: 20888 features in newdata not used in prediction.

## Warning: 18481 features in newdata not used in prediction.

## Warning: 15180 features in newdata not used in prediction.
```

```
## Warning: 11116 features in newdata not used in prediction.
## Warning: 7547 features in newdata not used in prediction.
## Warning: 5298 features in newdata not used in prediction.
## Warning: 4128 features in newdata not used in prediction.
## Warning: 2789 features in newdata not used in prediction.
## Warning: 1993 features in newdata not used in prediction.
## Warning: 796 features in newdata not used in prediction.
## Warning: 91 features in newdata not used in prediction.
## Warning: 23 features in newdata not used in prediction.
## Warning: 5 features in newdata not used in prediction.
## Warning: 2 features in newdata not used in prediction.
```

```
# Load the results from GBERT transformer models
trainsize_transfer <- read_csv("transfer-files/results-training-size.csv", col_names = F)
```

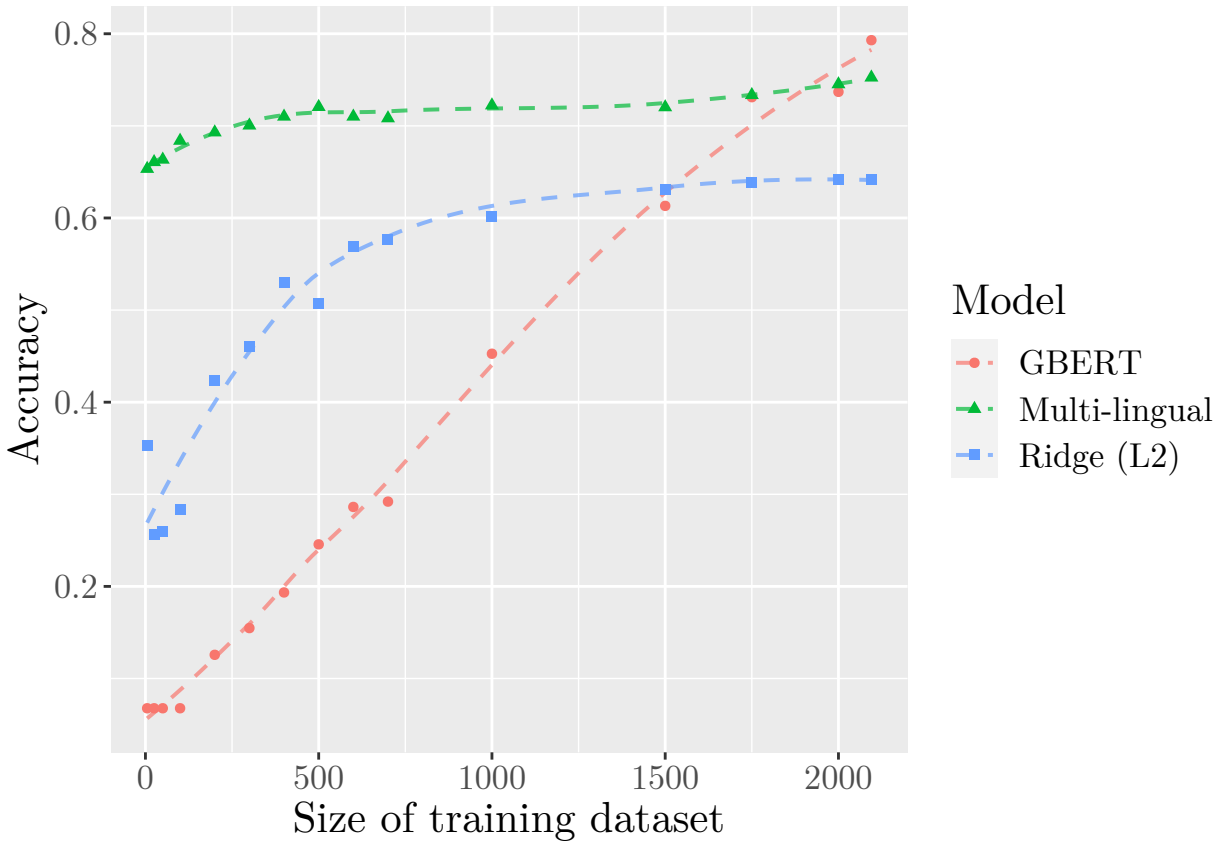
```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double(),
##   X2 = col_double()
## )
```

```
names(trainsize_transfer) <- c("accuracy", "train_size")
trainsize_transfer$Model <- "GBERT"
train_size <- rbind(trainsize_transfer, trainsize_ridge)

# Load the results from multilingual transformer models
trainsize_multi <- read_csv("other-countries/results-training-size.csv", col_names = F)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double(),
##   X2 = col_double()
## )
```

```
names(trainsize_multi) <- c("accuracy", "train_size")
trainsize_multi$Model <- "Multi-lingual"
train_size <- rbind(train_size, trainsize_multi)
```

```
ggplot(train_size, aes(x = train_size, y = accuracy)) + theme(text = element_text(size = 16)) +
    # geom_smooth(aes(color = Model, lty = Model), method = 'loess', formula = 'y ~
# x', se = F) +
geom_line(aes(group = Model, color = Model, lty = Model), stat = "smooth", method = "loess",
    formula = "y ~ x", size = 0.7, linetype = 2, alpha = 0.7, se = F) + geom_point(aes(color = Model,
    shape = Model)) + xlab("Size of training dataset") + ylab("Accuracy") + ggsave(str_c("plots/training
    device = cairo_pdf, width = 5 * 2^0.5, height = 5) + ggsave(str_c("plots/training-size-simulation.pn
    width = 5 * 2^0.5, height = 5)
```




```
# Time needed to run script (much shorter when textmodels are just loaded from a
# file) The estimation time for the single textmodels can found in the table
# above.

print(Sys.time() - start_time)
```

```
## Time difference of 2.918991 mins
```

```
# In total, the script needs about 2-3h to run.
```