Midterm Mini

Clare Cruz

10/9/2021

Methods Memo

In this short study, several text-analysis methods are utilized to determine the differences between student scores for a written test and identify potential areas to optimize a school's curriculum. Multiple methods are integrated into the analysis to answer the research questions since 'differences' between texts can be interpreted in many ways. Formally, the processes are keyword analysis, logistic regression model, and factor analysis. The intuition behind the chosen approaches was to evaluate the individual tokens or words, token categories, and clusters of token types to get a comprehensive overview of the disparity between scores.

First, the keyword analysis was used to compare individual tokens between the high and low scoring tests. This result would indicate if any specific words or clusters could describe some of the disparity in scores. To do this, separate document frequency matrices were created for high and low scoring tests. Then, the log-likelihood was calculated for each token between the two texts. The tokens with the largest log-likelihoods for high and low-scoring tests were identified, but the entire table was also evaluated to identify any token clusters.

Next, a logistic regression model is built with DocuScope tagged tokens to determine if particular categories of words could explain the different test scores. The significant variables in the final model can be helpful to the school's pedagogy by outlining specific types of terms that need more emphasis in the learning process. Before building the model, the variance inflation factors were calculated to determine if the data met the modeling assumptions. In the model building process, a multinominal regression model was also tested but the logistic regression model was favored since the binary variable made the interpretation simpler. Note that the Biber tag was also used to build the regression model but the results were inconclusive. After the model was built, the C-index was calculated to determine if the model had robust predictive power.

Lastly, factor analysis was considered with Biber's tagging to decide if any themes could distinguish the various tests and languages. Identifying themes can also provide constructive information to the curriculum since it can indicate the ties between writing techniques and test scores. For the factor analysis, a scree plot was created to determine an approximate number of factors and the ANOVA test was performed to see which factors resulted in a significant model. Then, the mean factor scores were calculated for every test score to see which test scores were different between the factors. The same method was also applied to the student's language to see if any patterns emerged based on the student's language.

Summary Report

Three different test-analysis methods are utilized determine the differences between student scores for a written test and identify potential areas to optimize a school's curriculum. The intention behind the chosen approaches was to evaluate the individual tokens or words, token categories, and clusters of token types to get a comprehensive overview of the disparity between scores.

First, the keyword analysis showed that the low-scoring students used personal pronouns significantly more than high-scoring students. For example, ‘I’ and ‘you’ had the greatest disparity between the scores and occurred more frequently in the lower scores (Table 2). In contrast, high-scoring students used more specialized nouns like ‘industry’ and model verbs such as ‘would’. In Table 2, the log-likelihood value is a standardized measurement of the differences between the absolute frequencies (AF) in the texts. This result indicates that students who use impersonal words tend to have higher scores.

Table 1: Tokens with the highest keyness values in the low-scoring exams when compared to the high-scoring exams.

Token	LL	AF_High_Scores	AF_Low_Scores
i	54.37	384	593
you	47.11	247	408
he	45.43	95	204
everything	35.68	8	50
ad	31.44	0	22

Similarly, the final logistic regression model also pointed to objective verbiage since the significant variables include academic terms and information-oriented phrases. However, other essential variables showed that students with higher scores provided a cohesive story with transitions, function words, and descriptions. The full list of significant variables with their coefficients can be viewed in Table 4. For this model, the exact interpretation for the coefficients is complex, but generally speaking the positive coefficients indicate that these variables increase the odds that the test score will be high. Then, the magnitude corresponds to how much that variable increases the odds for success. For example, students that used negative change words (informationnegativechange) increased their chances for a high score more than any other variable since their coefficient of 1.29 is the largest. The full description for these variables can be found at: https://huggingface.co/browndw/docusco-bert. Overall, the data met the underlying assumptions to build the model and has a high C-index value which indicates that the model has a lot of prediction power. Therefore, the model is reliable and we can expect that these variables are fairly accurate for predicting high and low test scores.

Table 2: Summary regression statistics for the significant coefficients in the logistic regression model that predicts high and low test scores.

	Estimate	Pvalue
(Intercept)	-30.46	0.00
academicterms	0.19	0.00
academicwritingmoves	0.97	0.02
facilitate	0.26	0.04
forcestressed	0.19	0.00
informationchange	0.31	0.04
informationchangenegative	1.29	0.04
informationexposition	0.13	0.01
informationstates	0.30	0.00
informationtopics	0.19	0.00
inquiry	0.22	0.01
metadiscourseinteractive	0.55	0.00
narrative	0.21	0.00
positive	0.23	0.00
syntacticcomplexity	0.30	0.00
uncertainty	0.29	0.05

Finally, the factor analysis resulted in two significant factors or themes. The first factor agrees with the other two methods by showing that low-scoring tests had more informal words like personal pronouns while high-scoring tests were more complex and formal. Additionally, the factors showed that Arabic-speaking students struggled with the test since they utilized informal language more than others. Then, the second significant factor showed a distinction between scores based on what tense they used. Specifically, low-scoring students used the present tense while the higher-scoring students used prepositions and modal verbs. Moreover, Korean students relied on the present tense verbs while other students applied the model verbs and prepositions. This result can be directly seen in Figure 1 which shows that Korean and Arabic students performed among the worst compared to the other students. Thus, the factor analysis provides a possible explanation behind the disparity.

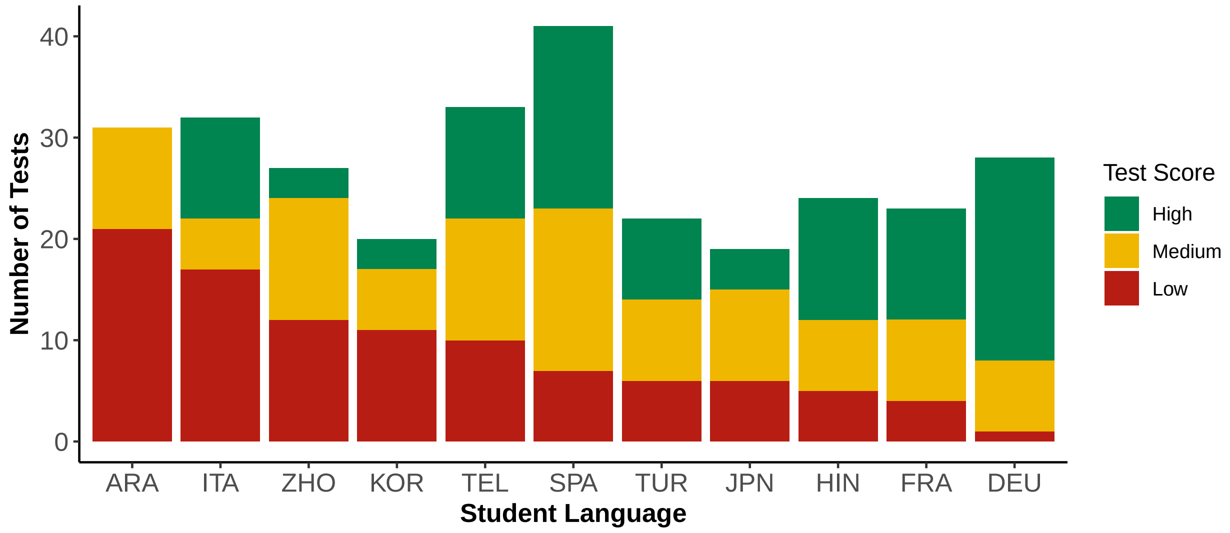

Figure 1: Student test performance per student native language.

The results of the analysis indicate that there are three potential opportunities for curriculum improvement. All the methods agree that informal writing with personal pronouns is not associated with high scores. Therefore, the curriculum should focus on formal writing techniques that emphasize objectivity. Then, the logistic regression model and factor analysis showed that cohesion and tense are associated with high scores. Thus, the school should consider implementing writing prompts that rely on these concepts, such as research papers. Finally, the factor analysis indicated that certain student groups struggle with the grammar associated with high scores. Further investigation should occur to see if targeted lessons are necessary to help these students perform well on the exam.

Code Appendix

```
# Package Load

library(cmu.textstat)
library(tidyverse)
library(quanteda)
library(quanteda.textstats)
library(ggraph)
library(tidyverse)
library(nnet)
library(corrplot)
library(future.apply)
library(udpipe)
library(nFactors)
library(pseudobibeR)
library(magrittr)
# Preprocessing
# Load in the data files
files_list <- list.files("C:/Users/cbrig/OneDrive/CMU/Text Analysis/Mini Midterm/midterm_corpus", full.n
# head(files_list)
# Metadata
exam_scores <- read.csv("C:/Users/cbrig/OneDrive/CMU/Text Analysis/Mini Midterm/midterm_meta.csv", heade
# Reading in DocuScope Dictionary for Logistic Regression
ds_dict <- dictionary(file = "/Users/cbrig/Downloads/ds_dict.yml")
# Read in the text files
essay_files <- readtext::readtext(files_list)
essay_files_udpipe <- split(essay_files, seq(1, nrow(essay_files), by = 2))

#head(essay_files)
## Download udpipe for factor analysis
#udpipe_download_model(language = 'english-ewt')
## Annotating with UDPipe
ncores <- 4L
plan(multisession, workers = ncores)

annotate_splits <- function(corpus_text) {
  ud_model <- udpipe_load_model("english-ewt-ud-2.5-191206.udpipe")
  x <- data.table::as.data.table(udpipe_annotate(ud_model, x = corpus_text$text,
                                                  doc_id = corpus_text$doc_id))
  return(x)
}
## Tagging the corpus
annotation <- future_lapply(essay_files_udpipe, annotate_splits, future.seed = T)
## Convert to udpipe tokens
annotation1 <- data.table::rbindlist(annotation)

annotation3 <- structure(annotation1, class = c("spacyr_parsed", "data.frame"))

udpipe_exam_tokens <- as.tokens(annotation3, include_pos = "tag", concatenator = "_")
## UDPipe DFM
udpipe_exam_dfm <- udpipe_exam_tokens %>%
  tokens_select("^.*[a-zA-Z0-9]+.*_[a-z]", selection = "keep", valuetype = "regex", case_insensitive = T
  dfm()

# udpipe_exam_dfm
## Tokenize the words normally for keyness analysis
essay_files <- essay_files %>%
  mutate(doc_id = str_extract(essay_files$doc_id, "^[0-9]+")) %>%
  mutate(doc_id = as.numeric(doc_id)) %>%
  arrange(doc_id) %>%
  mutate(exam_scores = exam_scores$test_score)
```

```r
exam_tokens <- essay_files %>%
  mutate(text = preprocess_text(text)) %>%
  corpus() %>%
  tokens(what="fastestword", remove_numbers=TRUE) %>%
  tokens_compound(pattern = phrase(multiword_expressions))
## Normal DFM
exam_dfm <- exam_tokens %>%
  dfm()

# exam_dfm
## Token summary per score

corpus_comp <- ntoken(exam_dfm) %>%
  data.frame(Tokens = .) %>%
  mutate(exam_id = str_extract(docnames(exam_dfm), "^[0-9]+")) %>%
  mutate(exam_id = as.numeric(exam_id)) %>%
  arrange(exam_id) %>%
  mutate(test_scores = exam_scores$test_score) %>%
  group_by(test_scores) %>%
  dplyr::summarize(Texts = n(),
    Tokens = sum(Tokens)) %>%
  rename("Final_Scores" = test_scores) %>%
  mutate(Final_Scores = c("High", "Medium", "Low")) %>%
  janitor::adorn_totals()

# corpus_comp
kableExtra::kbl(corpus_comp, caption = "Composition of the student test corpus.", booktabs = T, linesep
  kableExtra::kable_styling(latex_options = "HOLD_position") %>%
  kableExtra::kable_classic() %>%
  kableExtra::row_spec(3, hline_after = TRUE) %>%
  kableExtra::row_spec(4, bold=T)
# Keyness Analysis

low_dfm <- dfm_subset(exam_dfm, exam_scores == "low") %>% dfm_trim(min_termfreq = 1)
medium_dfm <- dfm_subset(exam_dfm, exam_scores == "medium") %>% dfm_trim(min_termfreq = 1)
high_dfm <- dfm_subset(exam_dfm, exam_scores == "high") %>% dfm_trim(min_termfreq = 1)
exam_kw <- keyness_table(high_dfm, low_dfm)
# kw_high <- exam_kw %>%
#   dplyr::select('Token', 'LL', 'AF_Tar', 'AF_Ref') %>%
#   arrange(desc(LL)) %>%
#   top_n(5, LL) %>%
#   rename(AF_High_Scores = AF_Tar) %>%
#   rename(AF_Low_Scores = AF_Ref)
#
# kableExtra::kbl(kw_high, caption = "Tokens with the highest keyness values in the high scoring exams 
#   kableExtra::kable_styling(latex_options = "HOLD_position") %>%
#   kableExtra::kable_classic()

kw_low <- exam_kw %>%
  dplyr::select('Token', 'LL','AF_Tar', 'AF_Ref') %>%
  arrange(LL) %>%
  top_n(5, -LL) %>%
  mutate(LL = abs(LL)) %>%
  rename(AF_High_Scores = AF_Tar) %>%
  rename(AF_Low_Scores = AF_Ref)

kableExtra::kbl(kw_low, caption = "Tokens with the highest keyness values in the low-scoring exams when
  kableExtra::kable_styling(latex_options = "HOLD_position") %>%
  kableExtra::kable_classic()

# Logistic Regression

## Categorize the tokens based on docuscop
ds_counts <- exam_tokens %>%
  tokens_lookup(dictionary = ds_dict, levels = 1, valuetype = "fixed") %>%
  dfm() %>%
  convert(to = "data.frame") %>%
  as_tibble() %>%
  mutate(exam_id = str_extract(doc_id, "^[0-9]+"))
ds_counts$exam_id <- as.numeric(ds_counts$exam_id)
ds_counts <- merge(ds_counts, exam_scores)
ds_counts <- ds_counts[ , -which(names(ds_counts) %in% c("quest_num","taker_lang"))] # delete the langu
names(ds_counts)[names(ds_counts) == 'test_score'] <- 'exam_score'
ds_counts$exam_score <- as.factor(ds_counts$exam_score)
#ds_counts
# Add a binary variable for pass/fail to do a binary logistic regression
ds_binary <- ds_counts
ds_binary$binary_score <-  with(ds_binary, ifelse(exam_score == "low", 0, ifelse(exam_score == "high", 
ds_binary_num <- ds_binary %>% dplyr::select(-c('exam_score')) %>%
  filter(binary_score != '')

ds_binary_num$binary_score <- as.factor(ds_binary_num$binary_score)
ds_binary_final <- ds_binary_num
log_data <- ds_binary_final[ , -which(names(ds_binary_final) %in% c("exam_id",'doc_id'))]
#mr_data <- ds_counts[ , -which(names(ds_counts) %in% c("doc_id",'exam_id'))]

# mr_fit <- multinom(exam_score~., data = mr_data)
glm_fit <- glm(binary_score~., data = log_data, family = "binomial")
logistic_sum <- summary(glm_fit)
logistic_sum_table <- logistic_sum$coefficients
logistic_sum_df <- data.frame(logistic_sum_table)
colnames(logistic_sum_df) <- c('Estimate','Std_error','z-value','Pvalue')
logistic_final <- logistic_sum_df %>%
  filter(Pvalue <= 0.05) %>%
  dplyr::select(Estimate, Pvalue)

kableExtra::kbl(logistic_final, caption = "Summary regression statistics for the significant coefficien
  kableExtra::kable_styling(latex_options = "HOLD_position") %>%
  kableExtra::kable_classic()
# VIFs are all below five, so no multicolinearity issues here
# car::vif(glm_fit)
# jtools::export_summs(glm_fit, exp = TRUE)
# DescTools::Cstat(glm_fit)
lang_comp <- ntoken(exam_dfm) %>%
  data.frame(Tokens = .) %>%
  mutate(exam_id = str_extract(docnames(exam_dfm), "^[0-9]+")) %>%
```

```r
  mutate(exam_id = as.numeric(exam_id)) %>%
  arrange(exam_id) %>%
  mutate(test_scores = exam_scores$test_score) %>%
  mutate(quest_num = exam_scores$quest_num) %>%
  mutate(test_lang = exam_scores$taker_lang) %>%
  group_by(test_lang, test_scores) %>%
  dplyr::summarize(Texts = n(),
    Tokens = sum(Tokens))

#lang_comp
plot_order <- lang_comp %>% filter(test_scores == 'low') %>% arrange(Texts) %>% select(test_lang)

lang_comp$test_scores <- factor(lang_comp$test_scores, levels=c("high", "medium", "low"))
lang_comp$test_lang <- factor(lang_comp$test_lang, levels=c("ARA", "ITA", "ZHO", "KOR", "TEL", "SPA", "


ggplot(lang_comp, aes(x = test_lang, y = Texts, fill= test_scores)) +
  geom_bar(stat = 'identity') +
  theme_classic() +
  scale_fill_manual(labels = c("High", "Medium", "Low"), values = c("#008450", "#EFB700", "#B81D13")) +
  labs(fill = "Test Score") +
  xlab('Student Language') +
  ylab('Number of Tests') +
 theme(plot.title = element_text(hjust = 0.5)) +
 theme(axis.text.x = element_text(size=12)) +
 theme(axis.text.y = element_text(size=12)) +
 theme(axis.title=element_text(size=12,face="bold"))
# Factor Analysis

exam_biber <- biber_udpipe(annotation1) %>%
  mutate(exam_id = str_extract(doc_id, "^[0-9]+")) %>%
  column_to_rownames("doc_id") %>%
  dplyr::select(exam_id, everything()) %>%
  mutate(exam_id = as.numeric(exam_id))

# head(exam_biber)
# Merge the metadata
biber_counts <- merge(exam_biber, exam_scores)

# Looking at factors based on each metadata
biber_score <- biber_counts[ , -which(names(biber_counts) %in% c("taker_lang","quest_num"))]
biber_score$exam_score <- as.factor(biber_score$test_score)

# Language
biber_lang <- biber_counts[ , -which(names(biber_counts) %in% c("exam_score","quest_num"))]
biber_lang$taker_lang <- as.factor(biber_lang$taker_lang)

# head(biber_counts,10)
biber_cor <- cor(exam_biber[-1], method = "pearson")
# corrplot::corrplot(biber_cor, type = "upper", order = "hclust", tl.col = "black", tl.srt = 45, diag =
#par(mfrow = c(1,2))
#screeplot_mda(biber_score)
#screeplot_mda(biber_lang)
```

```r
biber_score_mda <- mda_loadings(biber_score, n_factors = 5)
biber_lang_mda <- mda_loadings(biber_lang, n_factors = 5)
# knitr::kable(attr(biber_score_mda, 'loadings'), caption = "Factor loadings for exam data.", booktabs 
par(mfrow = c(1,2))
# mda.biber::heatmap_mda(biber_score_mda, n_factor = 1)
# mda.biber::heatmap_mda(biber_lang_mda, n_factor = 1)
# mda.biber::heatmap_mda(biber_score_mda, n_factor = 4)
# mda.biber::heatmap_mda(biber_lang_mda, n_factor = 4)
f_aov <- aov(Factor1 ~ group, data = biber_score_mda)
#broom::tidy(f_aov)

f_aov <- aov(Factor4 ~ group, data = biber_score_mda)
#broom::tidy(f_aov)
f1_lm <- lm(Factor1 ~ group, data = biber_score_mda)
names(f1_lm$coefficients) <- names(coef(f1_lm)) %>% str_remove("group")

f4_lm <- lm(Factor4 ~ group, data = biber_score_mda)
names(f1_lm$coefficients) <- names(coef(f4_lm)) %>% str_remove("group")
#jtools::export_summs(f4_lm, statistics = c(DF = "df.residual", R2 = "r.squared", "F statistic" = "stat

#jtools::export_summs(f1_lm, statistics = c(DF = "df.residual", R2 = "r.squared", "F statistic" = "stat
```