oscar_task2

oscar

2025-11-15

```
getwd()
```

```
## [1] "/Users/oscarodonnell/Desktop/PSTAT 197/module2/module2claims/scripts"
```

```
source("preprocessing.R")
```

```
## Loading required package: rvest
```

```
##
## Attaching package: 'rvest'
```

```
## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
## Loading required package: qdapRegex
```

```
##
## Attaching package: 'qdapRegex'
```

```
## The following object is masked from 'package:dplyr':
##
##     explain
```

```
## The following object is masked from 'package:ggplot2':
##
##     %+%
```

```
load("/Users/oscarodonnell/Desktop/PSTAT 197/module2/module2claims/data/claims-raw.RData")
```

```
claims_clean <- claims_raw %>%
  parse_data()
```

```
set.seed(110122)
partitions <- claims_clean %>%
  initial_split(prop = 0.8)

claims_train <- training(partitions)
claims_test <- testing(partitions)
```

```
train_unigram <- nlp_fn(claims_train)
test_unigram <- nlp_fn(claims_test)
```

```
train_unigram <- nlp_fn(claims_train)
test_unigram <- nlp_fn(claims_test)

# make feature matrix for unigram
x_train_uni <- train_unigram %>% select(-.id, -bclass) %>% as.matrix()
```

```r
train_cols <- colnames(x_train_uni)

# store training labels
y_train <- train_unigram$bclass

# do the same with testing
x_test_uni_raw <- test_unigram %>% select(-.id, -bclass) %>% as.matrix()

y_test <- test_unigram$bclass
```

```r
# add missing training columns to test
missing_cols <- setdiff(train_cols, colnames(x_test_uni_raw))
if (length(missing_cols) > 0) {
  zeros <- matrix(0,
                  nrow = nrow(x_test_uni_raw),
                  ncol = length(missing_cols))
  colnames(zeros) <- missing_cols
  x_test_uni_raw <- cbind(x_test_uni_raw, zeros)
}

#drop extra test columns
extra_cols <- setdiff(colnames(x_test_uni_raw), train_cols)
if (length(extra_cols) > 0) {
  x_test_uni_raw <- x_test_uni_raw[, !(colnames(x_test_uni_raw) %in% extra_cols), drop = FALSE]
}

# match training order

x_test_uni_good <- x_test_uni_raw[, train_cols, drop = FALSE]
```

Do PCA on the unigram features

```r
pca_uni <- prcomp(x_train_uni, center = TRUE, scale. = TRUE)

# Use 50 principal componests
k_uni <- 50

# make sure NAs are zeroes
z_train_uni <- pca_uni$x[, 1:k_uni, drop = FALSE]

z_test_uni <- scale(x_test_uni_good, center = pca_uni$center, scale = pca_uni$scale) %*% pca_uni$rotati
```

Fit logistic regression

```r
train_pcr_uni <- data.frame(bclass = y_train, z_train_uni)

fit_pcr_uni <- glm(bclass ~ ., data = train_pcr_uni, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Performance evaluation:

```r
p_hat_uni <- predict(fit_pcr_uni, newdata = data.frame(z_test_uni), type = "response")

# turn probs into predictions using > 0.5 ruling
```

```
class_hat_uni <- ifelse(p_hat_uni > 0.5,
                        levels(y_test)[2],
                        levels(y_test)[1]) %>%
  factor(levels = levels(y_test))

baseline_accuracy_uni <- mean(class_hat_uni == y_test)
```

Show baseline accuracy and AUC

```
library(yardstick)
library(tibble)

uni_results <- tibble(
  truth = y_test,
  prob  = p_hat_uni
)

auc_uni <- roc_auc(
  uni_results,
  truth,
  prob,
  event_level = "second"
)$.estimate

auc_uni
```

```
## [1] 0.6696772
```

We get a baseline accuracy of 0.6171285, and a AUC of0.6696772. These are decent levels, indicating that the model is better than randomly guessing, however, there's still room for improvement. Let's see if using secondary tokenization (bigrams) captures additional information about the claims status of a page.

Let's start with the bigram features and PCA

```
#function to tokenize bigrams with frerquency filtered
get_common_bigrams <- function(data, min_freq = 5) {
  data %>%
    unnest_tokens(bigram, text_clean, token = "ngrams", n = 2) %>%
    count(bigram) %>%
    filter(n >= min_freq) %>%
    pull(bigram)
}

common_bigrams <- get_common_bigrams(claims_train, min_freq = 5)
```

Create TF-IDF features for train and test

```
create_bigram_features <- function(data, bigrams) {
  data %>%
    unnest_tokens(bigram, text_clean, token = "ngrams", n = 2) %>%
    filter(bigram %in% bigrams) %>%
    count(.id, bclass, bigram) %>%
    bind_tf_idf(bigram, .id, n) %>%
    pivot_wider(id_cols = c(".id", "bclass"),
                names_from = bigram,
                values_from = tf_idf,
                values_fill = 0)
```

```
}

train_bigram <- create_bigram_features(claims_train, common_bigrams)
test_bigram <- create_bigram_features(claims_test, common_bigrams)
```

```
x_train_bi <- train_bigram %>% select(-.id, -bclass) %>% as.matrix()
x_test_bi <- test_bigram %>%
  select(-.id, -bclass) %>%
  select(any_of(colnames(x_train_bi))) %>%
  as.matrix()

#cleaning data
for (col in setdiff(colnames(x_train_bi), colnames(x_test_bi))) {
  x_test_bi <- cbind(x_test_bi, 0)
  colnames(x_test_bi)[ncol(x_test_bi)] <- col
}
x_test_bi <- x_test_bi[, colnames(x_train_bi)]

x_train_bi[!is.finite(x_train_bi)] <- 0
x_test_bi[!is.finite(x_test_bi)] <- 0

col_var <- apply(x_train_bi, 2, var)
keep_cols <- col_var > 0
x_train_bi <- x_train_bi[, keep_cols, drop = FALSE]
x_test_bi <- x_test_bi[, keep_cols, drop = FALSE]
```

PCA on bigram features

```
pca_bi <- prcomp(x_train_bi, center = TRUE, scale. = FALSE)
z_train_bi <- pca_bi$x[, 1:25]
z_test_bi <- predict(pca_bi, x_test_bi)[, 1:25]
```

Combine unigram log odds stacked with bigram pricipal components

```
#Make sure datasets align because of filtering in bigram step
common_train_ids <- intersect(train_unigram$.id, train_bigram$.id)
common_test_ids <- intersect(test_unigram$.id, test_bigram$.id)

train_uni_idx <- which(train_unigram$.id %in% common_train_ids)
train_bi_idx <- which(train_bigram$.id %in% common_train_ids)
test_uni_idx <- which(test_unigram$.id %in% common_test_ids)
test_bi_idx <- which(test_bigram$.id %in% common_test_ids)

y_train_aligned <- y_train[train_uni_idx]
y_test_aligned <- y_test[test_uni_idx]
z_train_uni_aligned <- z_train_uni[train_uni_idx, , drop = FALSE]
z_test_uni_aligned <- z_test_uni[test_uni_idx, , drop = FALSE]
z_train_bi_aligned <- z_train_bi[train_bi_idx, , drop = FALSE]
z_test_bi_aligned <- z_test_bi[test_bi_idx, , drop = FALSE]

# refit baseline model on aligned data for fair comparison
train_pcr_uni_aligned <- data.frame(bclass = y_train_aligned, z_train_uni_aligned)
fit_pcr_uni_aligned <- glm(bclass ~ ., data = train_pcr_uni_aligned, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
#get log-odds from aligned baseline model
logit_train_uni <- predict(fit_pcr_uni_aligned, type = "link")
logit_test_uni <- predict(fit_pcr_uni_aligned,
                          newdata = data.frame(z_test_uni_aligned),
                          type = "link")

# combine unigram with bigram PCs
stack_train <- data.frame(bclass = y_train_aligned,
                          base_logit = logit_train_uni,
                          z_train_bi_aligned)
stack_test <- data.frame(bclass = y_test_aligned,
                         base_logit = logit_test_uni,
                         z_test_bi_aligned)

# fit
fit_stack <- glm(bclass ~ ., data = stack_train, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
# evaluate stacked model
p_hat_stack <- predict(fit_stack, newdata = stack_test, type = "response")
class_hat_stack <- ifelse(p_hat_stack > 0.5, levels(y_test_aligned)[2], levels(y_test_aligned)[1]) %>%
                   factor(levels = levels(y_test_aligned))

stack_accuracy <- mean(class_hat_stack == y_test_aligned)
stack_auc <- roc_auc(tibble(truth = y_test_aligned, prob = p_hat_stack),
                     truth, prob, event_level = "second")$.estimate
```

```r
stack_accuracy
```

```
## [1] 0.6472081
```

```r
stack_auc
```

```
## [1] 0.6950724
```

The stacked bigram model has a test accuracy of 0.6472081 and an AUC of 0.6950724 , these are higher than the unigram performance of 0.6171285 and 0.6696772. The overall better performance of the stacked bigram model shows that bigrams capture additional information about the claims status of a page.