碩管一甲 M09218001 周彥廷

數據分析與應用 作業3：文件情緒分類器實作

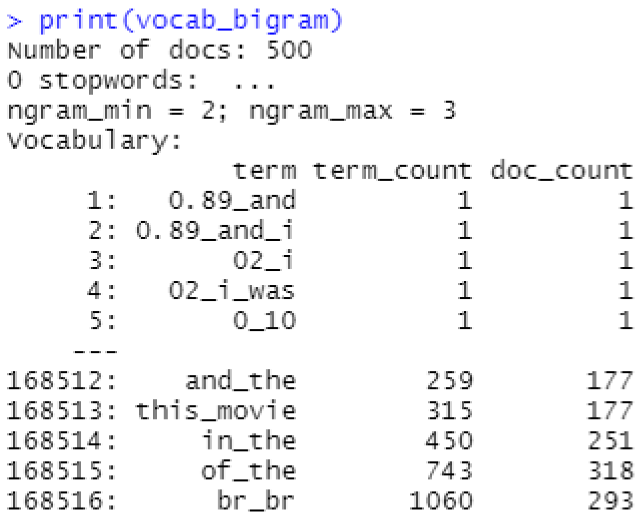
題目：

1.Please take the DTM on the based of bi-grams or tri-grams and run the classifier.

2.Please take the TF-IDF adjusted DTM and run the classifier.

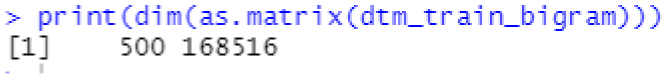
1.Please take the DTM on the based of bi-grams or tri-grams and run the classifier.

```
#安裝與載入套件
install.packages(c("text2vec","caret","data.table"))
library(caret)
library(text2vec)
library(data.table)
# load data 讀取資料
data("movie_review")
setDT(movie_review)
setkey(movie_review, id)
set.seed(2016L)
all_ids = movie_review$id
# data partition 切分資料
train_ids = sample(all_ids, 500)
test_ids = setdiff(all_ids, train_ids)
train = movie_review[J(train_ids)]
test = movie_review[J(test_ids)]
#文字處理
prep_fun = tolower
tok_fun = word_tokenizer
# Create an iterator to pass to the create_vocabulary function
it_train = itoken(train$review, preprocessor = prep_fun, tokenizer = word_tokenizer,
ids = train$id, progressbar = FALSE)
# Now create a vocabulary for training data
vocab_bigram = create_vocabulary(it_train, ngram = c(2, 3))
print(vocab_bigram)
```



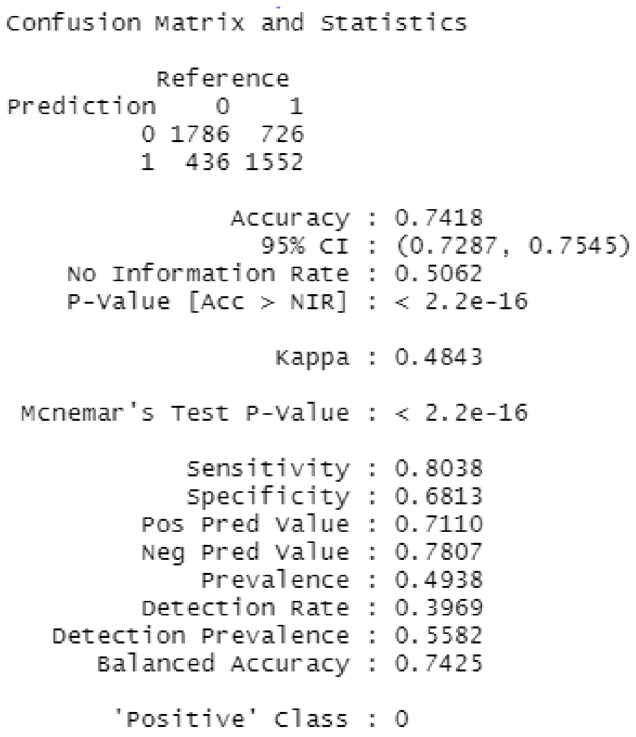
```
> print(vocab_bigram)
Number of docs: 500
0 stopwords:  ...
ngram_min = 2; ngram_max = 3
Vocabulary:
                  term term_count doc_count
      1:    0.89_and          1         1
      2: 0.89_and_i          1         1
      3:        02_i          1         1
      4:    02_i_was          1         1
      5:        0_10          1         1
    ---
168512:     and_the        259       177
168513: this_movie        315       177
168514:      in_the        450       251
168515:      of_the        743       318
168516:       br_br       1060       293
```

```
# vectorize the vocabulary
vectorizer_bigram = vocab_vectorizer(vocab_bigram)
# create a dtm
dtm_train_bigram = create_dtm(it_train, vectorizer_bigram)
print(dim(as.matrix(dtm_train_bigram)))
```



```
> print(dim(as.matrix(dtm_train_bigram)))
[1]    500 168516
```

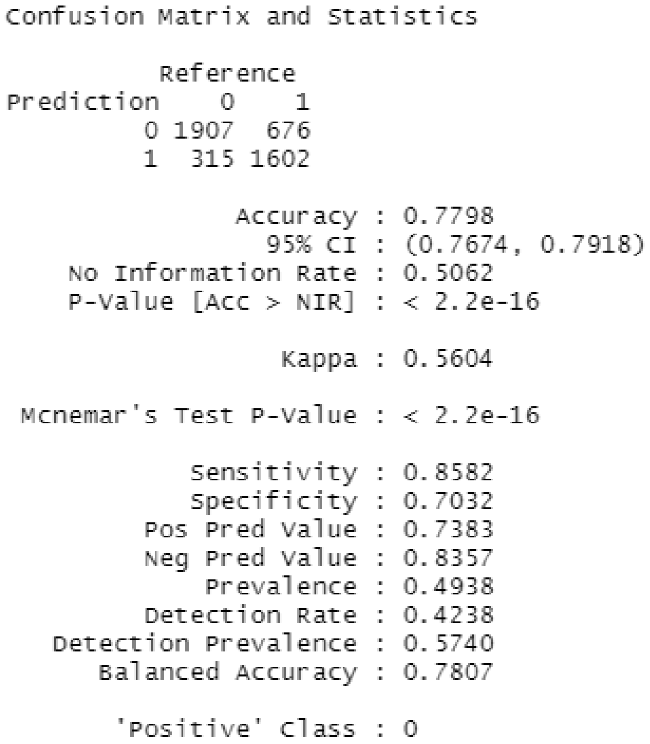
```
# vectorize vocabulary
# training the model
control <- trainControl(method="repeatedcv", number=5, repeats=3,
summaryFunction = multiClassSummary, selectionFunction = "best", classProbs = F,
search = "random", verboseIter = FALSE)
fit.model <- caret::train(x = dtm_train_bigram, y = as.factor(train[['sentiment']]),
method='glmnet', metric='Balanced_Accuracy', tuneLength = 5, trControl=control)
# create and vextorize    vocabulary for testing data
it_test = itoken(test$review, preprocessor = prep_fun, tokenizer = word_tokenizer, ids = test$id, progressbar = FALSE)
dtm_test = create_dtm(it_test, vectorizer_bigram)
# testing the model  混淆矩陣與測試樣本集正確率
preds = predict(fit.model, dtm_test)
confusionMatrix(preds, as.factor(test[['sentiment']]))
```



```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1786  726
         1  436 1552
                                          
               Accuracy : 0.7418          
                 95% CI : (0.7287, 0.7545)
    No Information Rate : 0.5062          
    P-Value [Acc > NIR] : < 2.2e-16       
                                          
                  Kappa : 0.4843          
                                          
 Mcnemar's Test P-Value : < 2.2e-16       
                                          
            Sensitivity : 0.8038          
            Specificity : 0.6813          
         Pos Pred Value : 0.7110          
         Neg Pred Value : 0.7807          
             Prevalence : 0.4938          
         Detection Rate : 0.3969          
   Detection Prevalence : 0.5582          
      Balanced Accuracy : 0.7425          
                                          
       'Positive' Class : 0               
```

2.Please take the TF-IDF adjusted DTM and run the classifier.

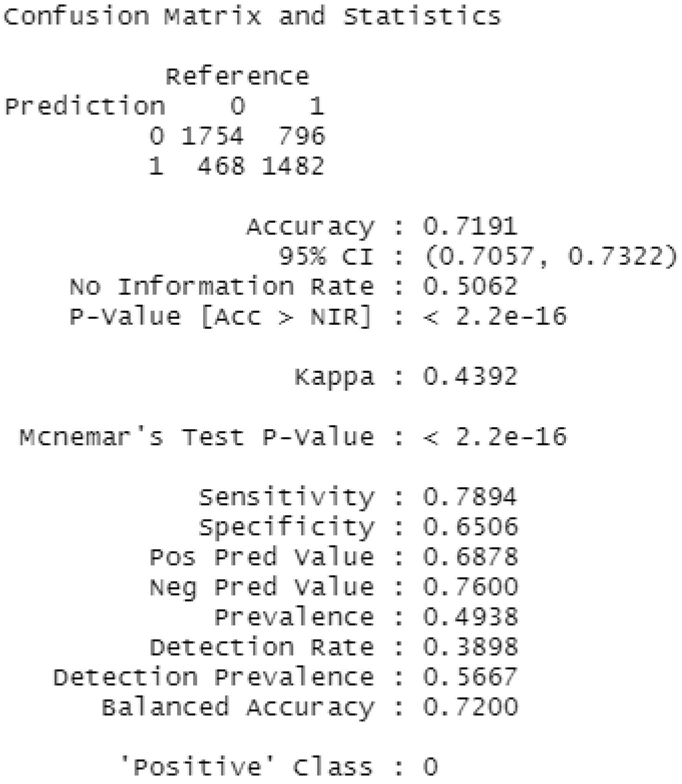
```
#安裝與載入套件
install.packages(c("text2vec","caret","data.table"))
library(caret)
library(text2vec)
library(data.table)
# load data
data("movie_review")
setDT(movie_review)
setkey(movie_review, id)
set.seed(2016L)
all_ids = movie_review$id
# data partition
train_ids = sample(all_ids, 500)
test_ids = setdiff(all_ids, train_ids)
train = movie_review[J(train_ids)]
test = movie_review[J(test_ids)]
#文字處理
prep_fun = tolower
tok_fun = word_tokenizer
# Create an iterator to pass to the create_vocabulary function
it_train = itoken(train$review, preprocessor = prep_fun, tokenizer = word_tokenizer,
ids = train$id, progressbar = FALSE)
# Now create a vocabulary for training data
# We see how to implement it using the text2vec package.
vocab = create_vocabulary(it_train)
vectorizer = vocab_vectorizer(vocab)
dtm_train = create_dtm(it_train, vectorizer) # create a dtm
tfidf = TfIdf$new()
dtm_train_tfidf = fit_transform(dtm_train, tfidf)
# training the model
control <- trainControl(method="repeatedcv", number=5, repeats=3,
summaryFunction = multiClassSummary, selectionFunction = "best", classProbs = F,
search = "random", verboseIter = FALSE)
fit.model <- caret::train(x = dtm_train_tfidf, y = as.factor(train[['sentiment']]),
method='glmnet', metric='Balanced_Accuracy', tuneLength = 5, trControl=control)
# create and vextorize   vocabulary for testing data
it_test = itoken(test$review, preprocessor = prep_fun, tokenizer = word_tokenizer, ids
= test$id, progressbar = FALSE)
dtm_test = create_dtm(it_test, vectorizer)
dtm_test_tfidf = fit_transform(dtm_test, tfidf)
# testing the model
preds = predict(fit.model, dtm_test_tfidf)
confusionMatrix(preds, as.factor(test[['sentiment']]))
```



```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1907  676
         1  315 1602

               Accuracy : 0.7798
                 95% CI : (0.7674, 0.7918)
    No Information Rate : 0.5062
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5604

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8582
            Specificity : 0.7032
         Pos Pred Value : 0.7383
         Neg Pred Value : 0.8357
             Prevalence : 0.4938
         Detection Rate : 0.4238
   Detection Prevalence : 0.5740
      Balanced Accuracy : 0.7807

       'Positive' Class : 0
```

```
#3.Please take the DTM on the based of (bi-grams or tri-grams) & TF-IDF and run the classifier.
library(caret)
library(text2vec)
library(data.table)
# load data
data("movie_review")
setDT(movie_review)
setkey(movie_review, id)
set.seed(2016L)
all_ids = movie_review$id
# data partition
train_ids = sample(all_ids, 500)
test_ids = setdiff(all_ids, train_ids)
train = movie_review[J(train_ids)]
test = movie_review[J(test_ids)]
prep_fun = tolower
tok_fun = word_tokenizer
# Create an iterator to pass to the create_vocabulary function
it_train = itoken(train$review, preprocessor = prep_fun, tokenizer = word_tokenizer,
ids = train$id, progressbar = FALSE)
# Now create a vocabulary for training data
vocab_bigram = create_vocabulary(it_train, ngram = c(2, 3))
print(vocab_bigram)
# Create an iterator to pass to the create_vocabulary function
it_train = itoken(train$review, preprocessor = prep_fun, tokenizer = word_tokenizer,
ids = train$id, progressbar = FALSE)
# Now create a vocabulary for training data
# We see how to implement it using the text2vec package.
vectorizer = vocab_vectorizer(vocab_bigram)
# create a dtm
dtm_train = create_dtm(it_train, vectorizer)
tfidf = TfIdf$new()
dtm_train_tfidf = fit_transform(dtm_train, tfidf)
# training the model
control <- trainControl(method="repeatedcv", number=5, repeats=3,
summaryFunction = multiClassSummary, selectionFunction = "best", classProbs = F,
search = "random", verboseIter = FALSE)
```

```
fit.model <- caret::train(x = dtm_train_tfidf, y = as.factor(train[['sentiment']]), method='glmnet', metric='Balanced_Accuracy', tuneLength = 5, trControl=control)
# create and vextorize    vocabulary for testing data
it_test = itoken(test$review, preprocessor = prep_fun, tokenizer = word_tokenizer, ids = test$id, progressbar = FALSE)
dtm_test = create_dtm(it_test, vectorizer)
dtm_test_tfidf = fit_transform(dtm_test, tfidf)
# testing the model
preds = predict(fit.model, dtm_test_tfidf)
confusionMatrix(preds, as.factor(test[['sentiment']]))
```



```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1754  796
         1  468 1482

               Accuracy : 0.7191
                 95% CI : (0.7057, 0.7322)
    No Information Rate : 0.5062
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4392

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.7894
            Specificity : 0.6506
         Pos Pred Value : 0.6878
         Neg Pred Value : 0.7600
             Prevalence : 0.4938
         Detection Rate : 0.3898
   Detection Prevalence : 0.5667
      Balanced Accuracy : 0.7200

       'Positive' Class : 0
```