DAUPHINE PSL UNIVERSITÉ PARIS RESEARCH UNIVERSITY

ML/DM
2016 / 2017

Travaux Pratiques n° 2 : Approches supervisées / Analyse de sentiments

Objectifs : mise en pratique sous R des concepts de l'apprentissage supervisé vus en cours et leur application à une tâche réelle d'analyse de sentiments

Dans ce travail, nous nous servirons des librairies `RTextTools` et `ROCR`. La librairie `RTextTools` contient un ensemble d'outils de traitement de texte, et un grand nombre d'algorithmes d'apprentissage, en particulier supervisé (`Decision Tree, Naive Bayes, knn, Random Forests, SVM`). La librairie `ROCR` nous sera utile pour comparer les performances des algorithmes en calculant les courbes ROC.

1. On commence par charger les données, et spécifier les étiquettes (`happy, sad`).

```
#Chargement des librairies
library(RTextTools)
library(e1071)
library(ROCR)
#Chargement des données d'entrainement et de test

setwd("C:/Data/")
happy = readLines("./happy.txt")
sad = readLines("./sad.txt")
happy_test = readLines("./happy_test.txt")
sad_test = readLines("./sad_test.txt")

tweet = c(happy, sad)
tweet_test = c(happy_test, sad_test)

tweet_all = c(tweet, tweet_test)

sentiment = c(rep("happy", length(happy)), rep("sad", length(sad)))
sentiment_test = c(rep("happy", length(happy_test)),
          rep("sad", length(sad_test)))

sentiment_all = as.factor(c(sentiment, sentiment_test))
```

2. La première étape revient à calculer la matrice termes-documents.

```
#On utilise la fonction create_matrix qui encapsule des méthodes du
#  paquet tm pour la creation de la matrice termes documents.

mat = create_matrix(tweet_all, language = "english",
     removeStopwords = FALSE, removeNumbers = TRUE,
```

```
    stemWords = FALSE, tm::weightTfIdf)

mat = as.matrix(mat)
```

3. On commence par appliquer un classifieur simple : Naive Bayes.

```
classifier_bayes = naiveBayes(mat[1:160, ],
        as.factor(sentiment_all[1:160]))

#On applique le modèle aux données de test

predicted_bayes = predict(classifier_bayes, mat[161:180, ])

#On affiche le résultat
predicted_bayes
```

4. Calcul de la matrice de confusion

```
table(sentiment_test, predicted)
```

5. Calcul de l'accuracy

```
recall_accuracy(sentiment_test, predicted)
```

6. On applique d'autres méthodes d'apprentissage supervisé (MAXENT, SVM, SLDA, BAGGING, RF, TREE). Vous pouvez en essayer d'autres.

```
container = create_container(mat, as.numeric(sentiment_all), trainSize =
    testSize = 161:180, virgin = FALSE)

models = train_models(container, algorithms = c("MAXENT", "SVM", "SLDA",
    "RF", "TREE"))

results = classify_models(container, models)
analytics = create_analytics(container, results)
```

7. Visualisation des performances des différents méthodes. Prenez le temps d'analyser ces résultats.

```
summary(analytics)
```

8. Autres commandes pour comparer les modèles

```
head(analytics@algorithm_summary)

head(analytics@label_summary)


head(analytics@document_summary)


head(analytics@document_summary)

# Ensemble Agreement
```

```
ensemble = create_ensembleSummary(analytics@document_summary)
ensemble


# analytics@ensemble_summary
```

9. Pratique de la validation croisée, avec 5 folds sur les méthodes SVM, GLMNET et MAXENT. Expliquez !

```
N = 5
cross_SVM = cross_validate(container, N, "SVM")


cross_GLMNET = cross_validate(container, N, "GLMNET")


cross_MAXENT = cross_validate(container, N, "MAXENT")
```

10. Calcul des courbes ROC. Analyser les méthodes suivantes et leurs résultats. Quelle méthode choisiriez-vous pour conduire votre analyse ?

```
pred_tree<-prediction(results$TREE_PROB, sentiment_test)
perf_tree<-performance(pred_tree,"tpr","fpr")

plot(perf_tree,col="blue",spread.estimate="stderror")

pred_svm<-prediction(results$SVM_PROB, sentiment_test)
perf_svm<-performance(pred_svm,"tpr","fpr")

plot(perf_svm,col="red",spread.estimate="stderror",add=TRUE)

pred_ME<-prediction(results$MAXENTROPY_PROB, sentiment_test)
perf_ME<-performance(pred_ME,"tpr","fpr")

plot(perf_ME,col="green",spread.estimate="stderror",add=TRUE)

pred_SLDA<-prediction(results$SLDA_PROB, sentiment_test)
perf_SLDA<-performance(pred_SLDA,"tpr","fpr")
plot(perf_SLDA,col="yellow",spread.estimate="stderror",add=TRUE)

pred_Bagg<-prediction(results$BAGGING_PROB, sentiment_test)
perf_Bagg<-performance(pred_Bagg,"tpr","fpr")
plot(perf_Bagg,col="pink",spread.estimate="stderror",add=TRUE)
```