Data mining and analysis

Individuele opdracht

Marc Veens (500758673)

Data mining and analysis
17-6-2018

Inhoudsopgave

Dataset

Voor het trainen en testen van de film recensies heb ik een tweetal datasets gebruikt. Allereest de toegereikte dataset, afkomstig van https://www.kaggle.com/c/word2vec-nlp-tutorial/data. De set bestaat uit een CSV van 25000 records waarbij per regel een review is geplaatst, en het sentiment van de review. Een 0 voor IMDB ratings lager dan een 5, en een 1 voor reviews groter of gelijk aan een 7.

De tweede dataset is afkomstig van https://www.kaggle.com/nltkdata/movie-review. Hetzelfde principe van sentiment is daar van toepassing, maar de structuur van de dataset is anders opgebouwd. Er wordt namelijk gebruik gemaakt van twee mappen met daarin positieve of negatieve recensies. De recensies zijn ook nog los verdeeld in verschillende bestanden.

Om alle recensies samen te brengen, heb ik ervoor gekozen om een R script ontwikkeld wat beide sets in eenzelfde vorm giet, namelijk een dataframe, met 3 kolommen; sentiment, tekst en document id.

```
getValuesFromTxt <- function(files, sentiment) {
  ds <- readtext::readtext(files)
  ds$doc_id <- NULL;
  ds$sentiment <- sentiment;
  colnames(ds)[1] <- "text"
  ds
}

setwd("C:\Users\MarcVe\OneDrive\Documenten\HvA\2017\Data analysis\Individual")

data <- read.csv2("dataset/1/labeledTrainData.csv", quote="", sep="\t")
data$id <- NULL
colnames(data)[2] <- "text"

dataset2neg <- getValuesFromTxt("dataset/2/neg/*.txt", 0)
dataset2pos <- getValuesFromTxt("dataset/2/pos/*.txt", 1)

data <- bind_rows(data, dataset2neg)
data <- bind_rows(data, dataset2pos)

## Randomize all data rows so there aren't humps of successive positive and negatives
data <- data[sample(nrow(data), nrow(data)), ]

data$doc_id <- seq.int(nrow(data))
```

Figuur 1 - Formateren van datasets

Om dit script niet elke keer te hoeven draaien wanneer ik de data nodig heb, koos ik ervoor om het vervolgens in MongoDB op te slaan. Ik zorg ervoor dat de gehele collectie eerst geleegd wordt voordat de nieuwe data toegevoegd wordt.

```
## Save to Mongo
con=mongo(collection="reviews",db="movies")
con$drop()
con$insert(data)
```

Figuur 2 - Opslaan van data in MongoDB

Op dit moment bevat de totale dataset 27.000 records. Het importeerproces is binnen een paar minuten afgehandeld.

Corpus

Nadat ik alle data had verzameld en in MongoDB had opgeslagen, bedacht ik me hoe ik de data voor de classifier moest voorbereiden. Om deze reden heb ik de data als eerst in een corpus opgeslagen en vervolgens opgeschoond.

```
## Clean the corpus
clean_corpus <- function(corpus) {
  corpus %>%
    tm_map(content_transformer(tolower)) %>%
    tm_map(removePunctuation) %>%
    tm_map(removeNumbers) %>%
    tm_map(removeWords, stopwords(kind="en"))
}
```

Figuur 3 - Opschonen van het corpus

Ik heb ervoor gekozen om het corpus door een dplyr functie te halen, zodat het makkelijk op te schrijven en te lezen is. Alle tekst wordt omgezet naar kleine letters, de interpunctie wordt verwijderd net als cijfers, en alle Engelse stopwoorden worden eruit gefilterd. Op die manier blijft er een schone dataset over waarmee een model getraind kan worden.

Classifier

Nadat alle data gereed was, moest ik kiezen voor een geschikte classifier. Ik twijfelde tussen Naive Bayes en Random Forest. Het voordeel van het Random Forest is dat het model beter nauwkeurig te krijgen is. Dat terwijl Naive Bayes al redelijk nauwkeurig van zichzelf is, maar moeilijk nauwkeuriger te maken is. Omdat het voor mij al redelijk een uitdaging was om dit model te schrijven, heb ik gekozen voor Naive Bayes. Ik wilde kijken hoe nauwkeurig de uitkomst was, en indien nodig later alsof overgaan op Random Forest.

Om de Naive Bayes classifier in de praktijk op te zetten ben ik wel wat langer bezig geweest. Ik kon naar mijn mening niet voldoende informatie halen uit Datacamp en moest daardoor verder op het internet doorzoeken. Ik wam vervolgens verschillende implementaties van Naive Bayes in R tegen. Uiteindelijk heb ik gekozen voor een methode die gebruik maakt van de findFreqTerms functie uit de Text Mining Package (https://www.rdocumentation.org/packages/tm/versions/0.7-3). Over het tweaken van deze functie, zie de kop Testen van de classifier.

```
## Create the DocumentTermMatrices using the frequent_terms dictionary
dtm_train_nb <- DocumentTermMatrix(corpus_clean_train, control=list(dictionary = frequent_terms))
dtm_test_nb <- DocumentTermMatrix(corpus_clean_test, control=list(dictionary = frequent_terms))

## Check for word occurence instead of frequency. This improves the accuracy of the model
train_nb <- apply(dtm_train_nb, 2, convert_count)
test_nb <- apply(dtm_test_nb, 2, convert_count)

## Train the naiveBayes model & predict using the test set
classifier <- naiveBayes(train_nb, df_train$sentiment)
prediction <- predict(classifier, newdata=test_nb)
```

Testen van de classifier

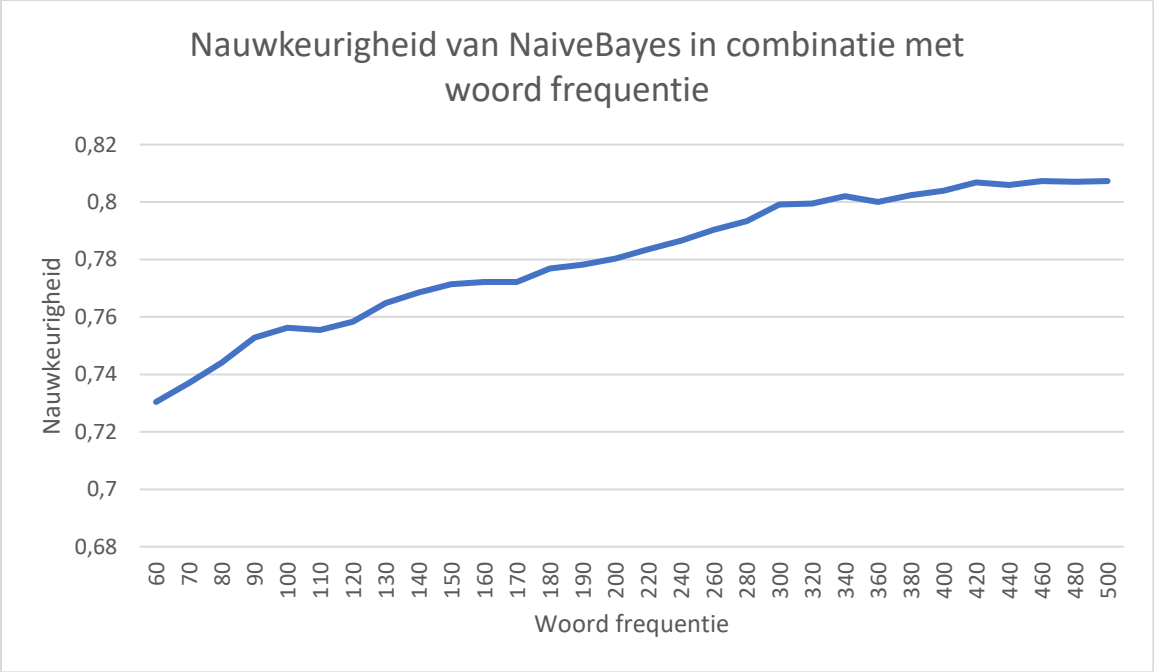
Om de classifier te trainen heb ik 75% van de dataset gebruikt, en 25% om het model te testen. Het is ook mogelijk om voor een 70/30 of 80/20 verhouding te kiezen, maar voor de nauwkeurigheid maakt het amper uit heb ik gemerkt.

Tussentijdse resultaat

In eerste instantie begon ik met een “term frequency” van 60. Dat wil dus zeggen, elk woord wat meer dan 60 keer voorkomt wordt opgeslagen en gebruikt om het model te trainen. Ik merkte dat het trainen van het model al enorm lang duurde, omdat er een enorme set is aan veelvoorkomende woorden. De nauwkeurigheid van het model kwam hierdoor ook op een schamele 0.73.

Nauwkeurigheid verbeteren

Om de nauwkeurigheid te verbeteren ben ik aan de slag gegaan met de findFreqTerms functie. Ik heb vanaf de waarde 60 proeven gedaan door er steeds 20 aan toe te voegen. Daardoor kreeg ik een stijgende lijn te zien in mijn nauwkeurigheid, en werd de tijd die de uitvoering innam steeds kleiner.



Uiteindelijk ben ik blijven hangen op de woord frequentie van 420, omdat ik zag dat het daarna weinig verschil bracht. Om de Naive Bayes classifier aan te maken ben je met 20.250 records 3 seconden bezig. Om vervolgens de voorspelling te maken op basis van de test-set van 6750 records ben je een kleine 2 minuten onderweg.

```
> system.time( classifier <- naiveBayes(trainNB, df_train$sentiment))
   user  system elapsed
   2.43    0.46    3.48
> system.time( prediction <- predict(classifier, newdata=testNB))
   user  system elapsed
 115.92    0.04  116.61
```

Uiteindelijke resultaat

De confusion matrix bij een woord frequentie van 420 ziet er als volgt uit:

```
          Reference
Prediction    0    1
         0 2640  570
         1  659 2881

               Accuracy : 0.8179
                 95% CI : (0.8085, 0.8271)
    No Information Rate : 0.5113
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.6355
 Mcnemar's Test P-Value : 0.01207

            Sensitivity : 0.8002
            Specificity : 0.8348
         Pos Pred Value : 0.8224
         Neg Pred Value : 0.8138
             Prevalence : 0.4887
         Detection Rate : 0.3911
   Detection Prevalence : 0.4756
      Balanced Accuracy : 0.8175

       'Positive' Class : 0
```

Er komt op dit moment een accuraatheid van 0.8179 uit, wat voor Naive Bayes erg netjes is.

Ik heb vervolgens nog een andere set aan 10 andere reviews toegevoegd. Het zijn 5 positieve reviews en 5 negatieve reviews. Tot mijn verbazing kwam hier een perfecte voorspelling uit:

```
Prediction
0 1
5 5
```

```
## My own reviews
my_reviews <- c("As you expected, Deadpool 2 is all that we expected and
                "Your going to be hard pressed to find a movie more o
                "Summer movies often hype themselves as spectacular e
                "Hello, For one thing I have to say that I have waite
                "I have never seen such an amazing film since I saw T
                "I didn't think an all women movie could be any worse
                "Good heavens... I didn't make it far into this embar
                "The filmmakers had absolutely no idea what they want
                "Yes, my wife and I laughed once in the first 25 minu
                "This was the worst movie I have ever seen. And I'm n
my_prediction <- predict_by_classifier(my_reviews, classifier, fivefreq)
table("Prediction"=my_prediction)
```