Neuralnet Sentiment

vak Data Mining

jaar 2016

Author: Stefan Schenk
500600679

Hogeschool van Amsterdam
29-12-16

Data Mining - BD2
Saskia Robben

Index

1. Introduction

The subject Data Mining requires each student to write a classifier in order to compare the classifications of two movie datasets to answer a research question. The first dataset is supplied by the HvA, the second must be found and cleaned.

The research question that will be answered in this report is:

> *How much influence does the content of a training-dataset have on the accuracy of a neural network?*

2. Data discovery

Overview

I have used the supplied dataset of IMDB and the dataset of Polarity in combination with a homemade dataset (Homebrew).

	IMDB	**Polarity**	**Homebrew**
Description	IMDb (imdb.com) is the world's most popular and authoritative source for movies, TV and celebrity content.	A thousand positive and thousand negative movie reviews, published for sentiment experiments.	A homemade dataset with made up reviews, very hard to classify.
Structure	Id, (sentiment), review	Id, sentiment, review	Id, sentiment, review
Format	tsv -> csv	txt -> csv	csv
Source	https://www.kaggle.com/c/word2vec-nlp-tutorial/data	http://www.cs.cornell.edu/People/pabo/movie-review-data	Homemade
Amount	Test = 25.000 Unlabeled = 50.000 Labeled = 25.000	Labeled = 2000	Labeled = 10

Clarification of choice

I've tried downloading and cleaning different datasets. These sets had no sentiment linked to the reviews by default. Therefore it would be very hard to determine the precision of a classifier, or all reviews had to be labeled manually. That is why I've chosen to clean and use the Polarity dataset, to be able to point out the differences in a confusion matrix easily. That said, it's great to have a separation between negative and positive reviews to get an impression of the quality of the classifier.

3. Data preparation

IMDB data preparation

The IMDB dataset consists of tab separated files. I have used the read.table method to read the data before writing it to comma separated files using the write.csv method. I constructed a method to demonstrate the cleaning of the IMDB dataset:

```
##################
#   TSV to CSV   #  Used for IMDB dataset
##################

convertTsvToCsv <- function(inputFile, outputFile) {
  if(is.null(inputFile) || is.null(outputFile))
    stop("Give input and output path of file")
  data <- read.table(inputFile, sep="\t", quote="", header=T)
  write.csv(data, outputFile)
}
```

The first review text looked like this:

```
> document <- data[1,]$review
> document
[1] With all this stuff going down at the moment with MJ i've started listening to his music, watching the odd doc
umentary here and there, watched The Wiz and watched Moonwalker again. Maybe i just want to get a certain insight
into this guy who i thought was really cool in the eighties just to maybe make up my mind whether he is guilty or
innocent. Moonwalker is part biography, part feature film which i remember going to see at the cinema when it was
originally released. Some of it has subtle messages about MJ's feeling towards the press and also the obvious mess
age of drugs are bad m'kay.<br /><br />Visually impressive but of course this is all about Michael Jackson so unle
ss you remotely like MJ in anyway then you are going to hate this and find it boring. Some may call MJ an egotist
for consenting to the making of this movie BUT MJ and most of his fans would say that he made it for the fans whic
h if true is really nice of him.<br /><br />The actual feature film bit when it finally starts is only on for 20 m
inutes or so excluding the Smooth Criminal sequence and Joe Pesci is convincing as a psychopathic all powerful dru
g lord. Why he wants MJ dead so bad is beyond me. Because MJ overheard his plans? Nah, Joe Pesci's character rante
d that he wanted people to know it is he who is supplying drugs etc so i dunno, maybe he just hates MJ's music.<br
 /><br />Lots of cool things in this like MJ turning into a car and a robot and the whole Speed Demon sequence. Al
so, the director must have had the patience of a saint when it came to filming the kiddy Bad sequence as usually d
irectors hate working with one kid let alone a whole bunch of them performing a complex dance scene.<br /><br />Bo
ttom line, this movie is for people who like MJ on one level or another (which i think is most people). If not, th
en stay away. It does try and give off a wholesome message and ironically MJ's bestest buddy in this movie is a gi
rl! Michael Jackson is truly one of the most talented people ever to grace this planet but is he guilty? Well, wit
h all the attention i've gave this subject....hmmm well i don't know because people can be different behind closed
 doors, i know this for a fact. He is either an extremely nice but stupid guy or one of the most sickest liars. I
hope he is not the latter.
8685 Levels: ''The 40 Year Old Virgin'''made me laugh a lot. I don't care if it is considered to be a very sexual
comedy, I just enjoyed many of the jokes and scenes present in this movie. Steve Carell is perfect as the virgin n
erd Andy Stitzer and I think the scene where Andy has his chest hair removed by wax one of the coolest, specially
because it is real. Many of the actors and actresses present in this movie are well known or already famous,by the
 way.<br /><br />Andy Stitzer has a peaceful life. He is a little bit strange and collects lots of toys, but seems
 harmless. One day, while playing poker with his friends of his work, they discover that Andy is in fact...virgin!
 And he is already 40 years old! After this surprising revelation, all his friends are trying to make Andy sleep w
ith a woman...the problem is the confusions in which Andy gets in,specially now that he is really starting to like
 Trish, a woman he met when she was buying a DVD player in the store he works at. ...
>
```

Figuur 1 - Impression cleaned review text

Polarity data preparation

The Polarity dataset consists of two folders in which a thousand reviews reside. I've converted these reviews to the same format as the IMDB reviews, merged them together and saved them as a csv file using the following method:

```
##################
#  Files to CSV  #  Used for Polarity dataset
##################

convertFilesToCsv <- function(folder, outputFile, sentiment) {
  if(is.null(folder) || is.null(outputFile) || is.null(sentiment))
    stop("Give input and output path of file")
  files <- list.files(folder)
  ans <- readline(paste("Read", length(files), "files? Y/N "))
  if (ans != "Y")
    return()
  matrix  <- matrix(nrow = length(files), ncol = 4)
  folder <- if(substr(folder, nchar(folder), nchar(folder)) == "/") folder else paste0(folder, "/")
  i <- 1
  for(f in files) {
    id <- f
    matrix[i,1] <- i
    matrix[i,2] <- id
    matrix[i,3] <- sentiment
    matrix[i,4] <- readChar(paste0(folder, f), file.info(paste0(folder, f))$size)
    i <- i+1
  }
  data <- as.data.frame(matrix)
  colnames(data) <- c("X", "id", "sentiment", "review")
  write.csv(data, outputFile)
}
```

Homebrew data preparation

The self written homebrew dataset was immediately created in the right format. Therefore it didn't require any preparation.

Tokenization

To make sure that the classifier was as generic as possible, I made sure that any further cleaning was consistently done right before classification. I've used a cleaning method that tokenizes all words in a document equally, so that different datasets, cleaned or not cleaned, can be used for the classifier:

```r
# Tokenize a document to produce a list of words
tokenize <- function(document) {
  # Lowercase all words for convenience
  doc <- tolower(document)
  # Remove all #hashtags and @mentions
  doc <- gsub("(?:#|@)[a-zA-Z0-9_]+ ?", "", doc)
  # Remove words with more than 3 numbers in them
  doc <- gsub("[a-zA-Z]*([0-9]{3,})[a-zA-Z0-9]* ?", "", doc)
  # Remove all punctuation
  doc <- gsub("[[:punct:]]", " ", doc)
  # Remove all newline and standalone characters
  doc <- gsub("*\\b[[:alpha:]]{1,2}\\b*", "", doc)
  # Remove all newline characters
  doc <- gsub("[\r\n]", "", doc)
  # Regex pattern for removing stop words
  stop_pattern <- paste0("\\b(", paste0(stopwords("en"), collapse="|"), ")\\b")
  doc <- gsub(stop_pattern, "", doc)
  # Replace whitespace longer than 1 space with a single space
  doc <- gsub(" {2,}", " ", doc)
  # Split on spaces and return list of character vectors
  doc_words <- strsplit(doc, " ")
  return(doc_words)
}
```

In the tokenizer function, all strange characters, punctuation, whitespace, standalone characters and stop words are filtered out so that the useful words remain:

```
> tokenize(document)
[[1]]
  [1] ""            "stuff"       "going"       "moment"      "started"     "listening"   "music"
  [8] "watching"    "odd"         "documentary" "watched"     "wiz"         "watched"     "moonwalker"
 [15] "maybe"       "just"        "want"        "get"         "certain"     "insight"     "guy"
 [22] "thought"     "really"      "cool"        "eighties"    "just"        "maybe"       "make"
 [29] "mind"        "whether"     "guilty"      "innocent"    "moonwalker"  "part"        "biography"
 [36] "part"        "feature"     "film"        "remember"    "going"       "see"         "cinema"
 [43] "originally"  "released"    "subtle"      "messages"    "feeling"     "towards"     "press"
 [50] "also"        "obvious"     "message"     "drugs"       "bad"         "kay"         "visually"
 [57] "impressive"  "course"      "michael"     "jackson"     "unless"      "remotely"    "like"
 [64] "anyway"      "going"       "hate"        "find"        "boring"      "may"         "call"
 [71] "egotist"     "consenting"  "making"      "movie"       "fans"        "say"         "made"
 [78] "fans"        "true"        "really"      "nice"        "actual"      "feature"     "film"
 [85] "bit"         "finally"     "starts"      "20"          "minutes"     "excluding"   "smooth"
 [92] "criminal"    "sequence"    "joe"         "pesci"       "convincing"  "psychopathic" "powerful"
 [99] "drug"        "lord"        "wants"       "dead"        "bad"         "beyond"      "overheard"
[106] "plans"       "nah"         "joe"         "pesci"       "character"   "ranted"      "wanted"
[113] "people"      "know"        "supplying"   "drugs"       "etc"         "dunno"       "maybe"
[120] "just"        "hates"       "music"       "lots"        "cool"        "things"      "like"
[127] "turning"     "car"         "robot"       "whole"       "speed"       "demon"       "sequence"
[134] "also"        "director"    "must"        "patience"    "saint"       "came"        "filming"
[141] "kiddy"       "bad"         "sequence"    "usually"     "directors"   "hate"        "working"
[148] "one"         "kid"         "let"         "alone"       "whole"       "bunch"       "performing"
[155] "complex"     "dance"       "scene"       "bottom"      "line"        "movie"       "people"
[162] "like"        "one"         "level"       "another"     "think"       "people"      "stay"
[169] "away"        "try"         "give"        "wholesome"   "message"     "ironically"  "bestest"
[176] "buddy"       "movie"       "girl"        "michael"     "jackson"     "truly"       "one"
[183] "talented"    "people"      "ever"        "grace"       "planet"      "guilty"      "well"
[190] "attention"   "gave"        "subject"     "hmmm"        "well"        "don"         "know"
[197] "people"      "can"         "different"   "behind"      "closed"      "doors"       "know"
[204] "fact"        "either"      "extremely"   "nice"        "stupid"      "guy"         "one"
[211] "sickest"     "liars"       "hope"        "latter"
```

Figuur 2 - Tokens from the review text

Word List data preparation

Every document is filtered on positive and negative words before the classification of a document, this is called word normalization. I have used a negative and positive words list for this comparison (Liu, n.d.). These two text files were merged and converted to a csv file. Because this list contained duplicate words, it caused an error during the creation of term frequency matrices. That's why I've chosen to remove the words that were both positive and negative. These words included a total of three: “envious”, “enviously”, “enviousness”.

```
word_list <- read.csv(paste0(getwd(), "/Data/pos-neg-words/word_list.csv"))
word_list <- as.data.frame(word_list[,1], stringsAsFactors=FALSE)
word_list <- as.data.frame(unique(word_list[,1]), stringAsFactors=FALSE)
write.csv(word_list, "word_list.csv")
```

4. Model planning / model building

Development

To be able to classify data in a row, I have tried to follow a tutorial that classifies data with different types of models (“RPubs - Sentiment Analysis in R,” n.d.). The code in this tutorial was old, which caused many errors and resulted in me rewriting almost all methods used in the tutorial. An additional downside is the inability to use the different classification methods after some code changes. I was only able to use the neural network from the nnet package. (The tutorial also included: naive bayes, randomforest, logistic regression and supportive vector machines).

After creating the tokenized document, it is compared to the word list with positive and negative words. The amount of occurrences of words being present, in both the document and the word list, are being stored in a so called corpus (Collection of documents).

```
corpus_freq <- function(tokens, corpus_size=NULL, word_list = NULL){
  # Concatenate all tokenized words into a single character list
  all_words <- do.call(c, tokens)

  #If corpus size is not blank, and word list is, create a word frequency frame
  #take the top occuring words up to the length of corpus_size
  #and reorder alphabetically

  #This gives us an data frame of the most frequent words in our corpus, ordered alphabetically
  #sized by the corpus_size parameter
  if(is.null(word_list) & !is.null(corpus_size)){
    corpusfreq <- data.frame(table(all_words))
    names(corpusfreq) <- c("Word", "Freq")
    corpusfreq$Word <- as.character(corpusfreq$Word)
    corpusfreq$Freq <- as.numeric(corpusfreq$Freq)
    corpusfreq <- corpusfreq[order(-corpusfreq$Freq), ]
    corpusfreq <- corpusfreq[1:corpus_size, ]
    corpusfreq <- corpusfreq[order(corpusfreq$Word), ]
  }
  #Else it is assumed a pre-compiled word list has been passed into the function
  corpusfreq <- data.frame(word_list)
  names(corpusfreq) <- c("Word")

  # N docs is where we will store the document frequency (I.E how many documents a word appears in)
  # We'll need this to calculate TF-IDF
  corpusfreq$n_docs <- 0

  # For every vector of words in our tokenized list, count how many times each word in our corpus occurs
  for(token_list in tokens){
    t <- data.frame(table(token_list))
    names(t) <- c("Word", "n_docs")
```

```
    t$n_docs <- 1
    t_freq <- merge(x=corpusfreq, y=t, by="Word", all.x=TRUE)
    t_freq$n_docs.y[is.na(t_freq$n_docs.y)] <- 0
    corpusfreq$n_docs <- corpusfreq$n_docs + t_freq$n_docs.y
  }
  return(corpusfreq)
}
```

The corpus is used to determine the Term Frequency - Inverse Document Frequency (tf-ifd) for each token in the document (“Tf-idf :: A Single-Page Tutorial - Information Retrieval and Text Mining,” n.d.).

```
tfidf <- function(document, corpus){

  doc_f <- data.frame(unlist(table(document)))
  names(doc_f) <- c("Word", "Freq")

  #Get a data frame of the words in the corpus found in the current document
  in_doc <- intersect(doc_f$Word, corpus$Word)
  not_in_doc <- data.frame(Word=corpus[!corpus$Word %in% doc_f$Word, ]$Word)
  doc_f <- doc_f[doc_f$Word %in% in_doc, ]

  #Get a data frame of the words in the corpus not found in the current document
  #Set their frequency to 0
  # not_in_doc <- data.frame(Word=setdiff(corpus$Word, document))
  not_in_doc$Freq <- 0

  #Bind our two data frames, we now have frequencies for the words that are in our corpus, and 0s everywhere else
  tf <- rbind.data.frame(doc_f, not_in_doc)
  tf$Word <- as.character(tf$Word)
  tf$Freq <- as.numeric(tf$Freq)

  #Order alphabetically again so it remains compatible with our corpus data frame
  tf <- tf[order(tf$Word), ]

  #Calculate the tfidf
  #log1p is the same as log(1+___)
  log_freq <- log1p(tf$Freq)
  log_doc_freq <- log1p(nrow(corpus)/corpus$n_docs)
  log_doc_freq[which(!is.finite(log_doc_freq))] <- 0
  if(length(log_freq) != length(log_doc_freq)) browser()
  tf$tfidf <- log_freq * log_doc_freq

  #Divide by zero errors get NA values, but should be 0s
  tf$tfidf[is.na(tf$tfidf)] <- 0
  return(tf)
}
```

The td-idf results were saved, for each word per document, in a feature vector. Whereupon the feature vector was added to the feature matrix with dimensions of *length(documents) x nrow(corpus)*.

During the creation of the vectors, an error may occur. Letting the user know that the limit of physical ram was reached. To solve this problem, one could increase the amount of memory that is available to R processes (“Increasing (or decreasing) the memory available to R processes,” n.d.).

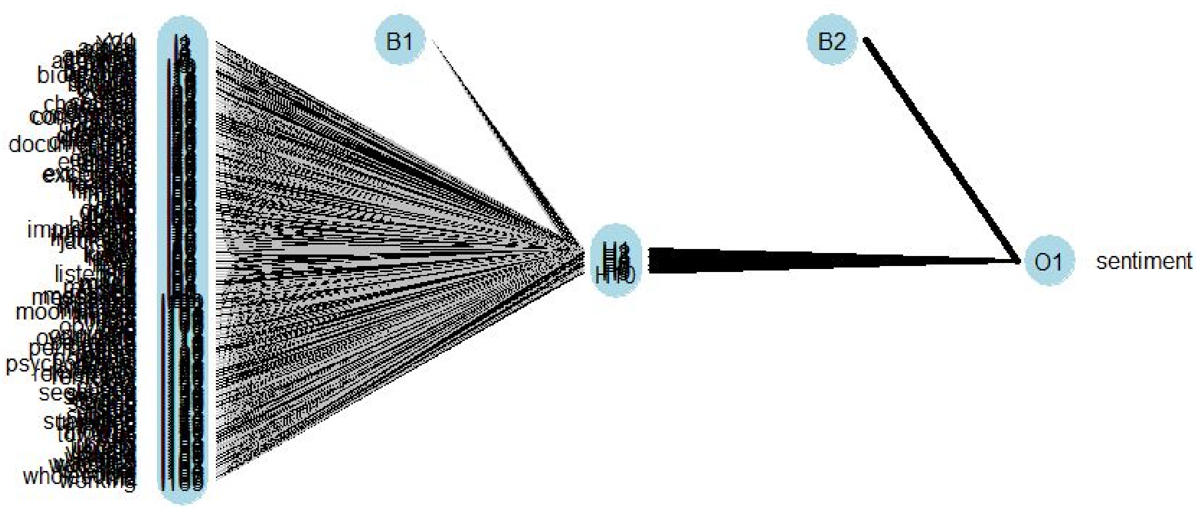
While training the model, the sentiment is added to the feature matrix so that the model can learn which word combinations contribute to which sentiment.

The neural network is built using the nnet method from the nnet package:

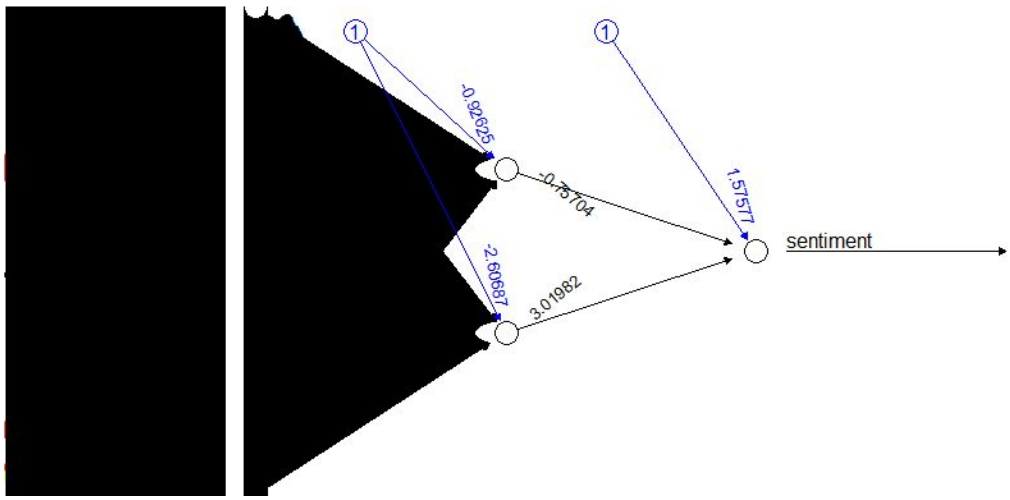
```
> m_nnet <- nnet(form, data=train, size=2,  MaxNWts=100000, maxit = 20)
# weights:  13575
initial  value 111.348331
iter  10 value 76.220294
iter  20 value 39.805097
final  value 39.805097
stopped after 20 iterations
> 
```

Figuur 3 - Print of the neural network creation

If the nnet package was used for creating a neural network, it can be plotted with the default plot method, or with the improved updated neural network plot method. The plots show the rather large amount of weights in the input layer.



Figuur 4 - Neuralnet plot with plot update



Figuur 5 - Neuralnet plot with default plot method

Improvements of the classifier

I’ve invested some time in learning other machine learning neural network packages like H2O and MXNetR. Deep learning can be applied using MXNetR using the GPU. Learning and applying these techniques took too much time.

Secondly I looked at unsupervised pre training partly because it was recommended by my teacher. Watching a video tutorial on unsupervised pre training taught me that it was mainly used on deep neural networks with at least three hidden layers. My neural network only has one hidden layer because the input is quite linearly separable. A feature is either negative, positive or in between. This is why my neural network can’t be called a deep neural network, and it’s why one hidden layer is sufficient for this kind of problem. This also means that A deep neural network implementation with unsupervised pre training could still provide good results, although creating one may be overkill for the amount of accuracy we could gain.

Options like backpropagation are either supported or not. The neuralnet package does support backpropagation, but had the same problems as the other methods introduced in the tutorial.

Lastly I tinkered with the arguments of the nnet method. Increasing the size and minimal weights. So far the amount of iterations had a direct impact on the accuracy of the classifier, of which the results will be presented in the next chapter.

Clarification of classifier choice

For the sentiment analysis problem, I picked the neural network classifier model because I wanted the different combinations of words to count towards the overall sentiment of a document. My neural network has one hidden layer that combines the weights of the input layer. Because not all separate words contribute independently to the sentiment of a document, a neural network is better fit for this problem than Naive Bayes for example. In which case bigrams, trigrams or quadrigrams should've been used instead. Decision trees could become way too complex, taking all the different words in the word list into account (approximately 6700 words).

The second reason I first picked the neural network is based on my interest in machine learning and artificial intelligence. I think that investigating how a neural network works contributes more to my understanding of machine learning because it's a concept I didn't fully understand. This is a good opportunity to play around with a neural network in R.

It would've been really great if I could fix all the other classification methods in the tutorial in order to compare the results. But unfortunately the input that would normally be consistent for each method, was changed too much because of the fixes I implemented. This is why I mainly focussed on getting one method to work.

5. Communicate the testing results

Using the *table()* function, true positives, true negatives, false positives and false negatives can be shown as a confusion matrix. With *table(prediction == reality)*, the amount of correct and incorrect predictions can be displayed as TRUE and FALSE values.

I've used the table method to create a function that displays the accuracy, sensitivity and fall-out of predictions in one overview:

```r
# Display results
results <- function(prediction, reality) {

  message("TRUE POSITIVES")
  t <- table(prediction, reality)
  tp <- t[2,2]
  fp <- t[2,1]
  tn <- t[1,1]
  fn <- t[1,2]
  cat(paste("TN:", tn, "FN:", fn, "\n"))
  cat(paste("FP:", fp, "TP:", tp))

  message("AMOUNT OF CORRECT PREDICTIONS")
  cat(paste("Correct:", (tp + tn), "/ Incorrect:", (fp + fn)))
  message("ACCURACY")
  cat((tp + tn) / (fp + fn + tp + tn))
  message("TRUE POSITIVE RATE (SENSITIVITY)")
  cat(tp / (tp + fn))
  message("FALSE POSITIVE RATE (FALL-OUT)")
  cat(fp / (fp + tn))
}
```

The output of twenty documents look as follows:

```
TRUE POSITIVES
TN: 11 FN: 2
FP: 0 TP: 7
AMOUNT OF CORRECT PREDICTIONS
Correct: 18 / Incorrect: 2
ACCURACY
0.9
TRUE POSITIVE RATE (SENSITIVITY)
0.7777778
FALSE POSITIVE RATE (FALL-OUT)
0
```

Figuur 8 - Output results( ) method

In a special case, the predictions of a dataset can all result in a positive (or negative) sentiment. In this case the method cannot determine what the TPR or FPR is, because this data can't be present.

Dataset	IMDB Classifier (25.000 records, 20 it)
Polarity - Negative Records: 2000 Labeled: Yes Omdat alle reviews negatief zijn, mist in de tabel de informatie om de TPR te bepalen omdat TP + FN nul is, en er niet door nul gedeeld kan worden.	Met IMDB classifier: **77%** TRUE POSITIVES TN: 771 FN: 0 FP: 229 TP: 0 AMOUNT OF CORRECT PREDICTIONS Correct: 771 / Incorrect: 229 ACCURACY 0.771 TRUE POSITIVE RATE (SENSITIVITY) NaN FALSE POSITIVE RATE (FALL-OUT) 0.229
Polarity - Positive Records: 1000 Labeled: Yes Omdat alle reviews positief zijn, mist in de tabel de informatie om de FPR te bepalen omdat FP + TN nul is, en er niet door nul gedeeld kan worden.	Met IMDB classifier: **78%** TRUE POSITIVES TN: 0 FN: 217 FP: 0 TP: 783 AMOUNT OF CORRECT PREDICTIONS Correct: 783 / Incorrect: 217 ACCURACY 0.783 TRUE POSITIVE RATE (SENSITIVITY) 0.783 FALSE POSITIVE RATE (FALL-OUT) NaN

A small vs big training dataset

Training a neural network with 2000 reviews and 20 iterations, the neural network has an accuracy of **82%**. This is higher than expected, for such a small training set.

```
TRUE POSITIVES
TN: 10394 FN: 2413
FP: 2106 TP: 10087
AMOUNT OF CORRECT PREDICTIONS
Correct: 20481 / Incorrect: 4519
ACCURACY
0.81924
TRUE POSITIVE RATE (SENSITIVITY)
0.80696
FALSE POSITIVE RATE (FALL-OUT)
0.16848
```

Figuur 6 - Accuracy using neural network trained with a small dataset.

The small training dataset has an accuracy of: 20.481 / 25.000 = 0.819 which is 82%. Training a neural network with 25.000 reviews and 20 iterations, the accuracy is **90%**. Which is even better.

```
TRUE POSITIVES
TN: 11410 FN: 1318
FP: 1090 TP: 11182
AMOUNT OF CORRECT PREDICTIONS
Correct: 22592 / Incorrect: 2408
ACCURACY
0.90368
TRUE POSITIVE RATE (SENSITIVITY)
0.89456
FALSE POSITIVE RATE (FALL-OUT)
0.0872
```

Figuur 7 - Accuracy using neural network trained with a big dataset.

Different content in the training dataset

To determine the difference of accuracy between two neural networks with different content in the training datasets, I've used two training datasets which both contained 2000 records. A labeled IMDB dataset with 1000 positive and 1000 negative records. And a labeled Polarity dataset with 1000 positive and 1000 negative records. The code block below was used to make sure that the IMDB training dataset had an exact equal amount of positive and negative reviews.

```
> negative <- imdb[imdb$sentiment == 0,]
> nrow(negative)
[1] 3923
> positive <- imdb[imdb$sentiment == 1,]
> nrow(positive)
[1] 4077
> positive <- head(positive, 1000)
> negative <- head(negative, 1000)
> nrow(positive)
[1] 1000
> imdb <- rbind(positive, negative)
> nrow(imdb)
[1] 2000
```

I then classified the entire IMDB, Polarity and Homebrew datasets using both classifiers. I calculated the accuracy in order to compare the results between the classifiers. Because only 2000 records were used during the training, the accuracy won't be as high, which is not the aim of this experiment.

The results of the experiment are shown on the next page.

Dataset	IMDB Classifier (2000 records, 20 it)	Polarity Classifier (2000 records, 20 it)
IMDB Records: 25.000 Labeled: Yes De IMDB classifier scoort het hoogst op de IMDB dataset.	Accuracy: **82%** F1-score: **82%** TRUE POSITIVES TN: 10394 FN: 2413 FP: 2106 TP: 10087 AMOUNT OF CORRECT PREDICTIONS Correct: 20481 / Incorrect: 4 ACCURACY 0.81924 TRUE POSITIVE RATE (SENSITIVI 0.80696 FALSE POSITIVE RATE (FALL-OUT 0.16848	Accuracy: **57%** F1-score: **63%** TRUE POSITIVES TN: 5196 FN: 3535 FP: 7304 TP: 8965 AMOUNT OF CORRECT PREDICTIONS Correct: 14161 / Incorrect: 10 ACCURACY 0.56644 TRUE POSITIVE RATE (SENSITIVITY 0.7172 FALSE POSITIVE RATE (FALL-OUT) 0.58432
Polarity Records: 2000 Labeled: Yes	Accuracy: **79%** F1-score: **79%** TRUE POSITIVES TN: 787 FN: 215 FP: 213 TP: 785 AMOUNT OF CORRECT PREDICTIONS Correct: 1572 / Incorrect: 42 ACCURACY 0.786 TRUE POSITIVE RATE (SENSITIVI 0.785 FALSE POSITIVE RATE (FALL-OUT 0.213	Accuracy: **87%** F1-score: **88%** TRUE POSITIVES TN: 806 FN: 62 FP: 194 TP: 938 AMOUNT OF CORRECT PREDICTIONS Correct: 1744 / Incorrect: 256 ACCURACY 0.872 TRUE POSITIVE RATE (SENSITIVITY 0.938 FALSE POSITIVE RATE (FALL-OUT) 0.194
Homebrew Records: 10 Labeled: Yes Ik bij een paar reviews zo hard mogelijk geprobeerd de classifier te misleiden door positieve woorden te gebruiken in negatieve reviews en andersom. Hierdoor zijn de false positives die zijn voorspeld logisch.	Accuracy: **50%** F1-score: **55%** TRUE POSITIVES TN: 2 FN: 2 FP: 3 TP: 3 AMOUNT OF CORRECT PREDICTIONS Correct: 5 / Incorrect: 5 ACCURACY 0.5 TRUE POSITIVE RATE (SENSITIVI 0.6 FALSE POSITIVE RATE (FALL-OUT 0.6	Accuracy: **60%** F1-score: **60%** TRUE POSITIVES TN: 3 FN: 2 FP: 2 TP: 3 AMOUNT OF CORRECT PREDICTIONS Correct: 6 / Incorrect: 4 ACCURACY 0.6 TRUE POSITIVE RATE (SENSITIVITY 0.6 FALSE POSITIVE RATE (FALL-OUT) 0.4

The accuracy is written in color above each screenshot, the F-score is written next to it. To determine the score of each classifier, I used the F1-score formula (“F1 score - Wikipedia,” n.d.). The formula is given by: F1 = 2TP / (2TP + FP + FN).

```
>>> def f1(tp, fp, fn):
...   return( 2*tp / (2*tp + fp + fn) )
```

Keep in mind that the Homebrew dataset only has 10 records, therefore it has a great impact on the overall score. Nevertheless, the IMDB trained classifier beats the Polarity trained classifier with the results of both accuracy measures.

Method	IMDB	Polarity
Accuracy	**70%**	**68%**
F1	**78%**	**70%**

Conclusion

During the creation of a neural network, the amount of iterations and the size of the training dataset has a direct influence on the accuracy of the classifier.

The research question in this report is:

> *How much influence does the content of a training-dataset have on the accuracy of a neural network?*

From the test results, it is concluded that the IMDB classifier is more accurate than the Polarity classifier based on the accuracy and F1 scores. The IMDB classifier accurately predicted 79% of the sentiment in the Polarity dataset while the Polarity classifier accurately predicted 57% of the sentiment in the IMDB dataset. Therefore it can also be concluded, knowing that both training datasets contained an equal amount of reviews of both positive and negative nature, that the content of the reviews has a direct impact of the accuracy of a classifier. So either the style or complexity in which the reviews were written contributed to the performance of the classifiers.

Bibliography

F1 score - Wikipedia. (n.d.). Retrieved January 5, 2017, from https://en.wikipedia.org/wiki/F1_score

Increasing (or decreasing) the memory available to R processes. (n.d.). Retrieved December 17, 2016, from http://stackoverflow.com/questions/1395229/increasing-or-decreasing-the-memory-available-to-r-processes

Liu, B. (n.d.). Opinion Mining, Sentiment Analysis, Opinion Extraction. Retrieved December 14, 2016, from https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon

RPubs - Sentiment Analysis in R. (n.d.). Retrieved December 10, 2016, from http://rpubs.com/lgendrot/sentiment

Tf-idf :: A Single-Page Tutorial - Information Retrieval and Text Mining. (n.d.). Retrieved December 19, 2016, from http://www.tfidf.com/