Text Classification with R

Wouter van Atteveldt

June 1, 2016

Machine Learning or automatic text classification is a set of techniques to train a statistical model on a set of annotated (coded) training texts, that can then be used to predict the category or class of new texts.

R has a number of different packages for various machine learning algorithm such as maximum entropy modeling, neural networks, and support vector machines. `RTextTools` provides an easy way to access a large number of these algorithms.

In principle, like 'classical' statistical models, machine learning uses a number of (independent) variables called *features* to predict a target (dependent) category or class. In text mining, the independent variables are generally the term frequencies, and as such the input for the machine learning is the document-term matrix.

RTextTools can be installed directly from CRAN:

```
install.packages("RTextTools")
```

Obtaining data

For this example, we will use Amazon reviews from http://jmcauley.ucsd.edu/data/amazon/ and classify whether they are positive or negative. See the hand-out 'Getting Sentiment Resources' hand-out for how to download and prepare these yourlelf, or you can download the directly from http://rawgit.com/vanatteveldt/learningr/master/data/reviews.rds.

```
reviews = readRDS("data/reviews.rds")
```

Creating the Document Term Matrix

So, the first step is to create a document-term matrix. To make it run faster for testing, we take a limited data set here. Since reviews are mostly positive (taking positive to be 4 or 5 stars), we sample 500 positive and 500 negative reviews to use:

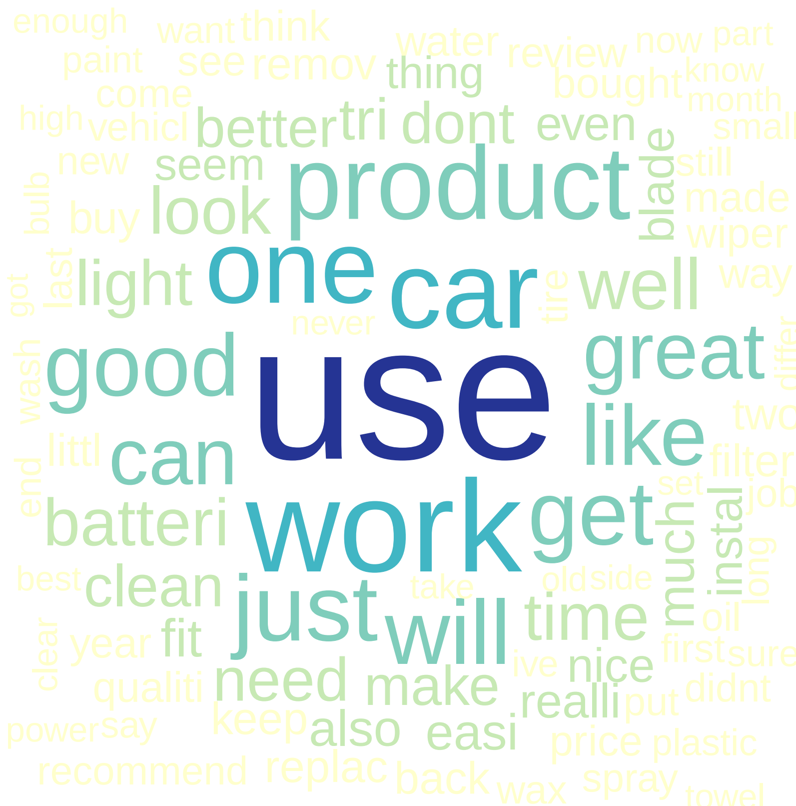
```
reviews$id = 1:nrow(reviews)
reviews$positive = as.numeric(reviews$overall >= 4)
pos = sample(reviews$id[reviews$positive == 1], 500)
neg = sample(reviews$id[reviews$positive == 0], 500)
reviews = reviews[reviews$id %in% c(pos, neg), ]
```

Now, we can create a dtm:

```
library(RTextTools)
dtm = create_matrix(reviews[c("summary", "reviewText")], language="english", stemWords=T)
```

Of course, now that we have a DTM we can plot a word cloud to get some feeling of the most frequent words:

```
library(corpustools)
dtm.wordcloud(dtm)
```



(we could also e.g. compare the words in positive and negative reviews, or run a topic model on only the positive or negative terms; see the handouts `comparing.pdf` and `lda.pdf`, respectively)

Preparing the training and testing data

The next step is to create the RTextTools *container*. This contains both the dt matrix and the manually coded classes, and you specify which parts to use for training and which for testing.

To make sure that we get a random sample of documents for training and testing, we sample 80% of the set for training and the remainder for testing. (Note that it is important to sort the indices as otherwise GLMNET will fail)

```
n = nrow(dtm)
train = sort(sample(1:n, n*.8))
test = sort(setdiff(1:n, train))
```

Now, we are ready to create the container:

```
c = create_container(dtm, reviews$positive, trainSize=train, testSize=test, virgin=F)
```

Using this container, we can train different models:

```
SVM <- train_model(c,"SVM")
MAXENT <- train_model(c,"MAXENT")
GLMNET <- train_model(c,"GLMNET")
```

Testing model performance

Using the same container, we can classify the 'test' dataset

```
SVM_CLASSIFY <- classify_model(c, SVM)
MAXENT_CLASSIFY <- classify_model(c, MAXENT)
GLMNET_CLASSIFY <- classify_model(c, GLMNET)
```

Let's have a look at what these classifications yield:

```
head(SVM_CLASSIFY)
```

SVM_LABEL	SVM_PROB
0	0.6274446
1	0.5109576
1	0.6390636
1	0.7141816
1	0.7563346
1	0.7438113

For each document in the test set, the predicted label and probability are given. We can compare these predictions to the correct classes manually:

```
t = table(SVM_CLASSIFY$SVM_LABEL, as.character(reviews$positive[test]))
t
```

```
##
##      0  1
##   0 69 31
##   1 27 73
```

(Note that the as.character cast is necessary to align the rows and columns) And compute the accuracy:

```
sum(diag(t)) / sum(t)
```

```
## [1] 0.71
```

Analytics

To make it easier to compute the relevant metrics, RTextTools has a built-in analytics function:

```
analytics <- create_analytics(c, cbind(SVM_CLASSIFY, GLMNET_CLASSIFY, MAXENT_CLASSIFY))
names(attributes(analytics))
```

```
## [1] "label_summary"     "document_summary"  "algorithm_summary"
## [4] "ensemble_summary"  "class"
```

The `algorithm_summary` gives the performance of the various algorithms, with precision, recall, and f-score given per algorithm:

```
head(analytics@algorithm_summary)
```

	SVM_PRECISION	SVM_RECALL	SVM_FSCORE	GLMNET_PRECISION	GLMNET_RECALL	GLMNE
0	0.69	0.72	0.70	0.69	0.80	
1	0.73	0.70	0.71	0.78	0.66	

The `label_summary` gives the performance per label (class):

```
head(analytics@label_summary)
```

	NUM_MANUALLY_CODED	NUM_CONSENSUS_CODED	NUM_PROBABILITY_CODED	PCT_CONSEN
0	96	107	106	
1	104	93	94	

Finally, the `ensemble_summary` gives an indication of how performance changes based on the amount of classifiers that agree on the classification:

```
head(analytics@ensemble_summary)
```

	n-ENSEMBLE COVERAGE	n-ENSEMBLE RECALL
n >= 1	1.00	0.72
n >= 2	1.00	0.72
n >= 3	0.71	0.77

The last attribute, `document_summary`, contains the classifications of the various algorithms per document, and also lists how many agree and whether the consensus and the highest probability classifier where correct:

```
head(analytics@document_summary)
```

SVM_LABEL	SVM_PROB	GLMNET_LABEL	GLMNET_PROB	MAXENTROPY_LABEL	MAXENTROPY_
0	0.6274446	0	0.7147742	1	0.
1	0.5109576	1	0.5134027	1	0.
1	0.6390636	0	0.6639251	1	0.
1	0.7141816	1	0.9048356	1	1.
1	0.7563346	1	0.9905123	1	1.
1	0.7438113	1	0.9405767	1	1.

Classifying new material

New material (called 'virgin data' in RTextTools) can be coded by placing the old and new material in a single container. Let's assume that we don't know the sentiment of 20% of our material:

```
reviews$positive[1:200] = NA
```

We now set all documents with a sentiment score as training material, and specify `virgin=T` to indicate that we don't have coded classes on the test material:

```
coded = which(!is.na(reviews$positive))
c = create_container(dtm, reviews$positive, trainSize=coded, virgin=T)
```

We can now build and test the model as before:

```
SVM <- train_model(c,"SVM")
MAXENT <- train_model(c,"MAXENT")
GLMNET <- train_model(c,"GLMNET")
SVM_CLASSIFY <- classify_model(c, SVM)
MAXENT_CLASSIFY <- classify_model(c, MAXENT)
GLMNET_CLASSIFY <- classify_model(c, GLMNET)
analytics <- create_analytics(c, cbind(SVM_CLASSIFY, GLMNET_CLASSIFY, MAXENT_CLASSIFY))
names(attributes(analytics))
```

```
## [1] "label_summary"    "document_summary" "class"
```

As you can see, the analytics now only has the `label_summary` and `document_summary`:

```
analytics@label_summary
```

```
##   NUM_CONSENSUS_CODED NUM_PROBABILITY_CODED
## 0                 421                   418
## 1                 379                   382
```

```
head(analytics@document_summary)
```

SVM_LABEL	SVM_PROB	GLMNET_LABEL	GLMNET_PROB	MAXENTROPY_LABEL	MAXENTROPY_
0	0.7845596	0	0.7190914	0	0.
1	0.7788332	1	0.9172107	1	0.
1	0.7657043	1	0.8136316	1	0.
0	0.8481629	0	0.8979255	0	1.
1	0.6769943	1	0.7211425	1	0.
1	0.6108151	1	0.7173253	1	0.

The label summary now only contains an overview of how many where coded using consensus and probability. The document_summary lists the output of all algorithms, and the consensus and probability code.