Classification: Is the song Spanish, Korean, or Arabic?

Julia Stelman

30/9/2019

Introduction

In this study, I look at a number of audio features of 516 songs and see if I can try to predict the language of the song's lyrics with statistical modeling. I randomly picked three different languages to look at, each from a different language family. I use a multinomial regression approach to formulate my model. The audio features eventually used are

- Danceability (number between 0 and 1)
- Energy (number between 0 and 1)
- Loudness (number between -25 and 0)
- Speechiness (number between 0 and 1)
- Tempo (number between 50 and 250)

The languages I will be looking at today are

- Spanish
- Korean
- Arabic

Thank you to Spotify, Musixmatch, Everynoise.com, and lang-detect for the help I got from your libraries, websites, and APIs in the data collection and cleaning phase of this project, which I did in Python.

Also thank you to the R libraries quanteda, tidyverse, nnet, DescTools, caret, leaps, car, and knitr.

Data

A preview of the data

	danceability	energy	loudness	speechiness	acousticness	liveness	tempo	lang
129	0.647	0.489	-11.058	0.0330	0.730	0.131	94.967	ko
256	0.547	0.797	-5.378	0.0472	0.047	0.193	139.884	ar
387	0.732	0.510	-6.192	0.2600	0.111	0.102	87.837	es

Methods

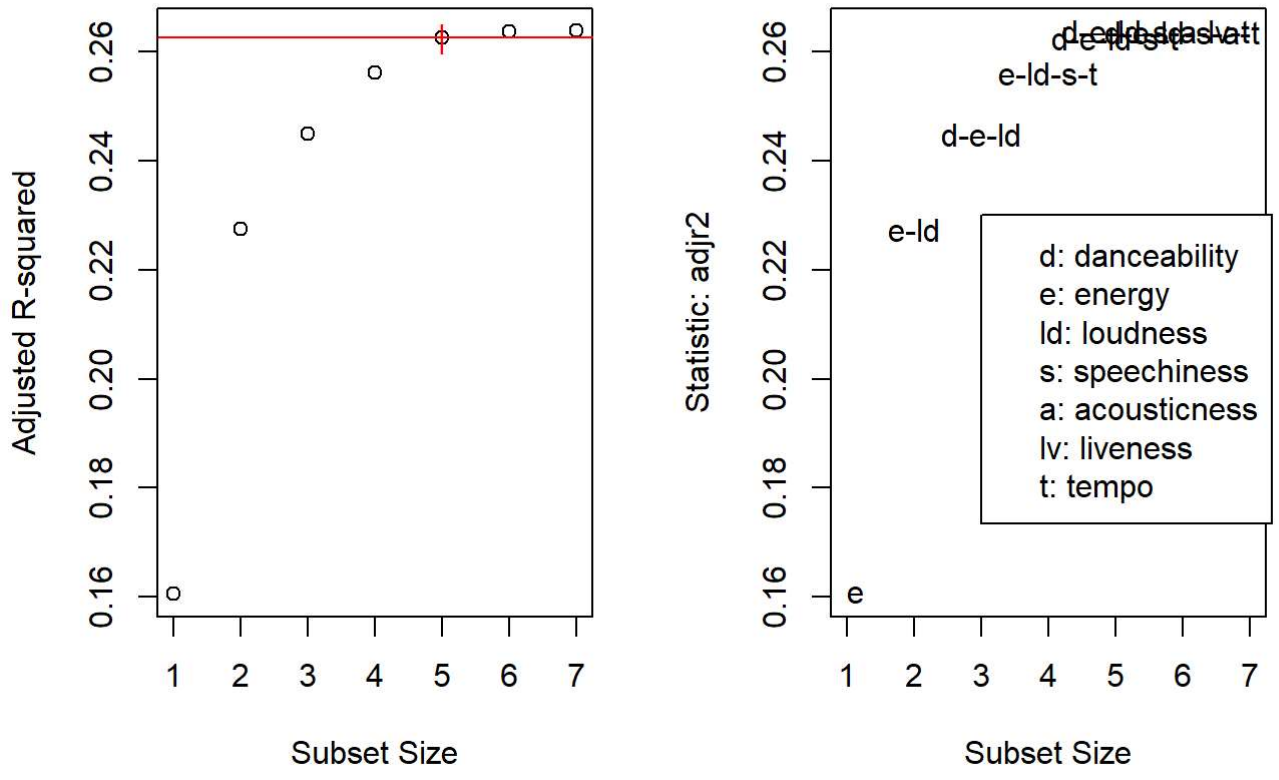
I used R to fit a multinomial regression model to the data.

Here is the model I ended up using

```
## # weights:  21 (12 variable)
## initial  value 425.162956
## iter  10 value 343.533374
## iter  20 value 342.512782
## final  value 342.512737
## converged
```

```
## Call:
## multinom(formula = lang ~ danceability + energy + loudness +
##     speechiness + tempo, data = train_ska_df)
##
## Coefficients:
##    (Intercept) danceability    energy    loudness speechiness       tempo
## es   -2.494691     2.559326   0.51073 -0.01022033   -6.880827 0.007887857
## ko    9.116123    -2.426092 -10.35331  0.39921908   -4.592110 0.012803649
##
## Residual Deviance: 685.0255
## AIC: 709.0255
```

I deducted that model from the original 7-variable model (which also included *acousticness* and *liveness*) using several handy R packages (which I have listed at the bottom of the intro) to find the set of predictor variables that would minimize the Adjusted R-squared.



The plot on the left, uses an Adjusted R-squared maximizer to decide when to stop adding variables.

We see here that five variables, *danceability, energy, loudness, speechiness,* and *tempo* should be kept, and the other two *acousticness* and *liveness* should be dropped.

I have to say, I am very happy with the coefficient of determination I got. The adjusted R-squared of the model is .026, which, with respect to the nature of this data, not too shabby.

Results

Confusion Matrix

	ar	es	ko
ar	15	10	1
es	22	28	3
ko	6	5	39

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   6.356589e-01   4.534884e-01   5.463743e-01   7.185628e-01   3.333333e-01
## AccuracyPValue  McnemarPValue
##   2.234871e-12   3.556654e-02
```

The confusion matrix above has the true labels as columns and predictions as rows. The model has a 63.6% accuracy rate, the p-value for the one-tailed test of whether or not the model predicts the a song's lyric language better than no information is very low (0.00000022), and the Mcnemar's Test p-value is 0.035. That's two statistically significant p-values!

Each person can judge for themself whether or not 63% is a good enough accuracy rate for what we want to predict, but it's significantly better than random guessing. It's clear that Spanish vs Arabic were confused more often than either one vs Korean. There is higher a tendency toward misclassifying arabic songs as Spanish.

Discussion

This excercise was a fun opportunity to try out new techniques I'm using in my classes. It was a small sample size, and that may or may not have worked to my advantage. Though all in all, I was pleasantly surprised by how nicely the results came out. (The adjusted coefficient of determination was 0.26!) I chose this trio more or less at random. (I did consider sample pool sizes and linguistic root similarity.)

Obviously, there are many factors I didn't have the capacity to account for. A big one is genre. As of now, I have not found a way to get reliably- and uniformly- sourced genre information on all the songs in my data. It's something I hope to keep looking into, though it is one of many symptoms of the data-collection limitations I faced. I lost a lot of data due to non-response bias by the lyric searcher.

I plan to explore other combinations of languages and language families in the near future, as I am curious to know if that this set of languages was just a lucky trifecta, or if I get similar results for other groups. I am a statistician, so my guess is that this one lies somewhere in the middle.

As always, I hope to keep exploring this awesome dataset as time allows. As a full-time masters student, I'm not sure what that means for my availability, but stick with me. My next idea is to look at classifying languages into language families based on audio features.

Appendix

```
# Prepare to do a classication training and then testing.

## take an equal random sample from all three languages of interest
eS_ids <- subset(x=df,subset = lang == 'es')$sid
## n = 172 because there are exactly 174 songs in Arabic, the language with the fewest in the th
e trio (Spanish, Korean, Arabic)
eS_ids <- sample(eS_ids,172)
Ar_ids <- subset(x=df,subset = lang == 'ar')$sid
Ar_ids <- sample(Ar_ids,172)
Ko_ids <- subset(x=df,subset = lang == 'ko')$sid
Ko_ids <- sample(Ko_ids,172)

## take the subset of data with only those 172*3 songs and call it a new name
ska_df <- subset(x = df,select = names(df),subset = sid %in% c(eS_ids,Ar_ids,Ko_ids))

## take 1/4 of the song ids sampled from each language and put them aside for test data
test_ska_ids <- c(
  as.character(ska_df[ska_df$sid %in% sample(eS_ids,length(eS_ids) %/% 4),'sid']),
  as.character(ska_df[ska_df$sid %in% sample(Ar_ids,length(Ar_ids) %/% 4),'sid']),
  as.character(ska_df[ska_df$sid %in% sample(Ko_ids,length(Ko_ids) %/% 4),'sid']))

## the rest will be the training data
```

```
## use the test ids to subset a test data frame.
test_ska_df <- subset(x = ska_df, select = names(ska_df), subset = sid %in% test_ska_ids)
## use the remaining ids to subset a training dataframe
train_ska_df <- subset(x = ska_df, select = names(ska_df), subset = !(sid %in% test_ska_ids))

## get rid of the spotify id column, it's not needed anymore
test_ska_df <- select(test_ska_df, - sid)
train_ska_df <- select(train_ska_df, - sid)

## factorize the character columns that are left (there should only be one, language)
test_ska_df <- test_ska_df %>%  mutate_if(is.character, as.factor)
train_ska_df <- train_ska_df %>%  mutate_if(is.character, as.factor)
```

```
train_ska_df[c(129,256,387),]
```

```
#train the model
#### let's see which ones we really need

## first try all 7
mr_train_fit <- multinom(lang ~ danceability + energy + loudness +
                     speechiness + acousticness + liveness + tempo,
                     data = train_ska_df)
summary(mr_train_fit)
```

```
#### let's see which ones we really need
AFmatTr <- as.matrix(train_ska_df[,-8])
langVecTr <- train_ska_df[,8]
MSO <- regsubsets(AFmatTr,langVecTr)  #model selection object
regsub <- summary(MSO)
par(mfrow=c(1,2))
plot(1:7,regsub$adjr2,xlab="Subset Size",ylab="Adjusted R-squared")
abline(h = regsub$adjr2[5],col='red')
points(x = 5,y = regsub$adjr2[5],col='red', pch = '|')
subsets(MSO,statistic=c("adjr2"),legend = c(3,0.23))
```

```
##### Remove the two coefficients that we just cut using the Adjusted R-squared maximizer
###### Refit the model.

mr_train_fitr <- multinom(lang ~ danceability + energy + loudness + speechiness + tempo,
                      data = train_ska_df)

summary(mr_train_fitr)
```

```
##Now let's see how well it does on our test data

# Next, we create a data based on our fit.
prob_disc <- cbind(test_ska_df, predict(mr_train_fitr, newdata = test_ska_df,
                                    type = "probs", se = TRUE))

labeler <- function(id){
  if (id == 1) 'ar'
  else if (id == 2) 'es'
  else if (id == 3) 'ko'
}

predictions <- c(sapply(max.col(prob_disc[,9:11]),labeler))
truth <- sapply(as.character(prob_disc$lang),paste)

confusion <- c(truth,predictions)
confusion <- as.data.frame(confusion)
confusion <- confusion  %>%  mutate_if(is.character, as.factor)
confusion1 <- cbind(confusion[1:(length(confusion)%/%2)],
                    confusion[((length(confusion)%/%2)+1):length(confusion)])
names(confusion1) <- c("truth","predictions")
rm(confusion)


cm <- confusionMatrix(confusion1$predictions[1:129],
                 confusion1$truth[130:258])
knitr::kable(cm$table,caption = "Confusion Matrix")
cm$overall
```