Topic Modeling: Genre by Lyrics

Our project is an analysis of the relationship between song lyrics and song genres and seeking substantial clues in lyrics for determining the genre of the songs. Methods used include topic modeling, term frequency, LDA, GLM, stemming, and more. We explored models on this relationship, word correlations and associations, and potential hidden semantic patterns. By design, the project uses unsupervised methods. The first model is a 5 topic classification model using LDA. The second is a 2-topic classification model also using LDA.

1/8/18

Josue Garcia

Harvey Lao

Contents

[Abstract 2](#_Toc503232692)

[Introduction 2](#_Toc503232693)

[Figure 1: Raw Data 2](#_Toc503232694)

[Cleaning our Data 2](#_Toc503232695)

[Figure 2: Mutated Wordcount Column 3](#_Toc503232696)

[Figure 3: Analyzing Lyrics Distribution 3](#_Toc503232697)

[Figure 5: Top Words of Final Subset 4](#_Toc503232698)

[Figure 5: Removing Non-English Text 4](#_Toc503232699)

[Figure 6: I Wanna Be With You by Dj Khaled 5](#_Toc503232700)

[Corpus, Document Term Matrix and LDA 5](#_Toc503232701)

[Figure 7: Top Frequencies 6](#_Toc503232702)

[Figure 8: Bottom Frequencies 6](#_Toc503232703)

[Figure 9: Top Words 6](#_Toc503232704)

[Figure 10: Bottom Words 6](#_Toc503232705)

[Latent Dirichlet Allocation 7](#_Toc503232706)

[Figure 11: Normal Mixed Model Example 7](#_Toc503232707)

[5-Topic LDA Model 7](#_Toc503232708)

[Figure 12: Beta 8](#_Toc503232709)

[Figure 13: Gamma 8](#_Toc503232710)

[Figure 15: Top Words by Topic (Genre) 9](#_Toc503232711)

[2-Topic LDA Model 9](#_Toc503232712)

[Figure 16: 2-Topic LDA Results 10](#_Toc503232713)

[2-Topic GLM Model 10](#_Toc503232714)

[Figure 17: GLM Deviance from Residuals: Top (left) Bottom(right) 10](file:///C:\Users\Harvey\OneDrive\Documents\131%20Project%20Report%201.3.docx#_Toc503232715)

[Word Association 11](#_Toc503232716)

[Figure 18: Word Associations 11](#_Toc503232717)

[Results 11](#_Toc503232718)

[References 12](#_Toc503232719)

[Appendix 13](#_Toc503232720)

# Abstract

Our project is an analysis of the relationship between song lyrics and song genres. We explored models on this relationship, word correlations and associations, and potential hidden semantic patterns. By design, the project uses unsupervised methods. The first model is a 5 topic classification model using LDA. The second is a binomial model that can differentiate between two genres based (Hip-Hop and Metal). Lastly we fit a generalized fitted model based off the song’s artist, year, and word count. The 5-topic model was unable to correctly distinguish between Pop, Rock, and Metal. Consequently, this model yielded poor results with a ~48.5% accuracy rate. The second model yielded more promising results, at 89.35% accuracy rate, suggesting that Hip-Hop was relatively easy to differentiate from Metal. The GLM model was unsuccessful, due to the large variety of parameters and their respective ranges.

# Introduction

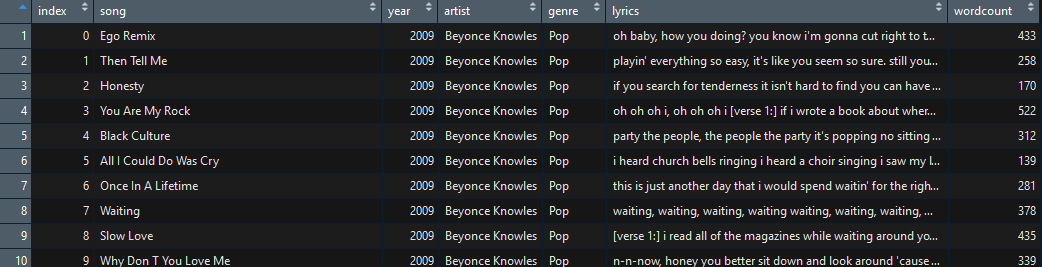
Acquiring the data from Kaggle, we can take a look at the raw data in excel:



## Figure 1: Raw Data

We can see 6 variables with over 362,000 observations. For the first part of our project, we are only interested in the lyrics. The universal terms for the objects used in topic modeling are called topics, documents, and terms. For this project, genre means topics, songs means documents, and lyrics/words mean terms.

# Cleaning our Data

First, we subset from the original data to remove genres in which we were uninterested. Removing the genres with the least words (Jazz, Folk, Indie, R&B, Electronic, Other, and Not Available), we were left with Rock, Pop, Metal, Hip Hop, and Country. We then mutated a new column into the dataset labeled ‘wordcount’, which counted the number of words in each song. Below is our dataset after we cleaned the song names, artist, genre, and added the column for wordcount.

## Figure 2: Mutated Wordcount Column

We then observed the distribution of songs based on the size of lyrics.

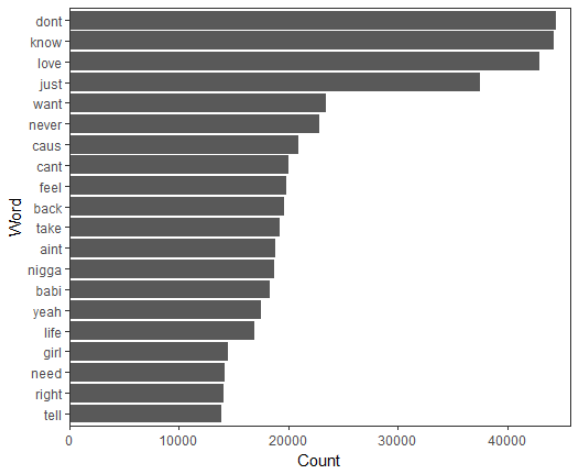
|  |  |
| --- | --- |
| # of words in lyrics | # of songs |
| Less than 200 | 112,265 |
| Less than 150 | 70,181 |
| Less than 100 | 28,986 |
| Less than 50 | 8,667 |

## Figure 3: Analyzing Lyrics Distribution

Analyzing the word count, we found that removing all songs with fewer than 100 words would not affect our dataset significantly, while simplifying our training model. Removing these 29,057 songs – roughly 13% of our data *after* we removed the other genres – we still had nearly 200,000 observations to work with.

We found that there was still too much data for our computers to efficiently run, so we sampled from our large dataset and obtained 6000 observations from each of the 5 genres, a 30000 point dataset. Let us call this new dataset our final subset.

Figure 4: Example of Final Subset



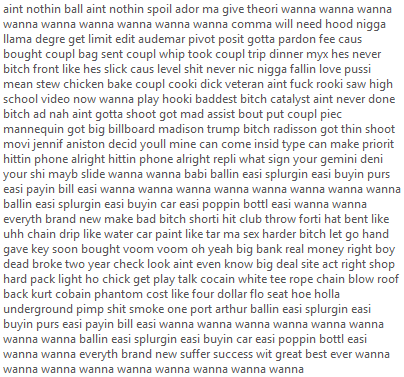
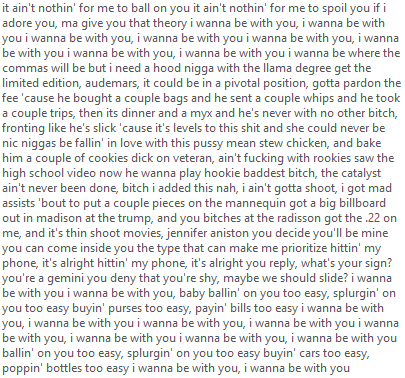
## Figure 5: Top Words of Final Subset

In order to run meaningful analysis on our final subset, we needed to clean the lyrics. We removed numbers, stopwords, punctuation, the words “intro/chorus/bridge/hook,” and many other non-contextual words. We found ways to be highly selective of what we were removing, such as words between brackets and parentheses. We had extreme difficulty removing non-English symbols because some “English” symbols were converted into incompatible symbols hidden in our lyrics (such as í).



Figure 5: Removing Non-English Text

Subsequently, we stemmed the words so that words suggesting the same context were deleted. If two different words had the same root, they were likely to be a different tense of each other. Stemming the words simplifies our model by reducing variance and standardizing how words are treated.

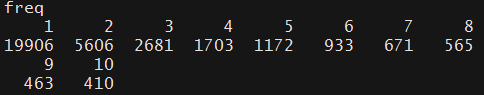


## Figure 6: I Wanna Be With You by Dj Khaled

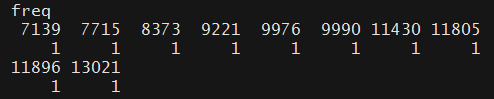
On the left side, we have half of the lyrics of this song containing punctuation and complete words. On the right, we have all the lyrics, stemmed and cleaned and ready for analysis. This step saves the model and environment much time and confusion when performing functions on the dataset. After cleaning the lyrics, we are finally ready to create and test training and test sets. We took the final subset with clean lyrics; we converted the Corpus into a data frame and mutated the clean lyrics back into our Final Subset. After this process, we were ready to create training and test sets.

# Corpus, Document Term Matrix and LDA

The training and test sets were then put into Corpus objects. We held the randomness constant using set.seed(1) in order for our results to be reproducible. We made a corpus, a collection of documents, of only the lyrics from the data frame. From here we utilized the tm package to clean and stem the lyrics. We put the cleaned lyrics back into the data frame and finished up the cleaning by removing non ASCII terms that slipped through the corpus. From here we converted the column with the cleaned lyrics into a Document Term Matrix, a matrix with each document as a row and each word as a column. From the DTM, we were able to show the most frequent words, as well as the least frequent words, and how often they were each observed.

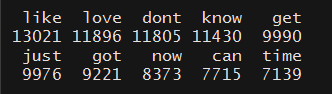


## Figure 7: Top Frequencies

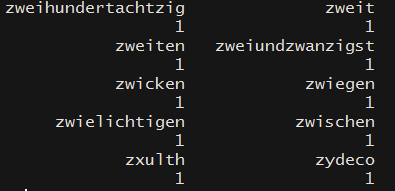


## Figure 8: Bottom Frequencies

We can see that there were 19,906 words that appeared only once, 5606 that appeared twice, and so on.



## Figure 9: Top Words

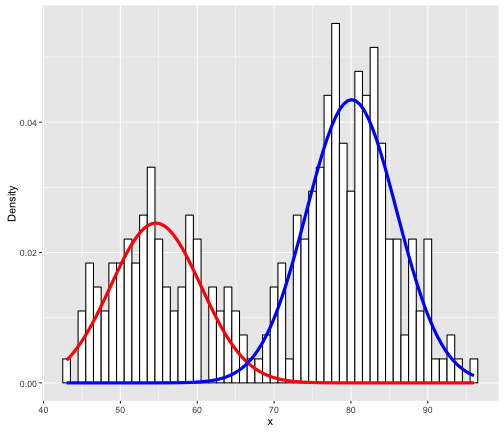


## Figure 10: Bottom Words

We can also take a peek at which words are most and least frequent. We see that the top words in the frequency table correspond to the top words in the word table. We can use this DTM object to create a classification model on our training set.

# Latent Dirichlet Allocation

Running Latent Dirichlet Allocation on the document term matrix, we created an unsupervised topic model with 5 topics. The idea behind a document term matrix is that each document is a mixture of words, where each word ‘belongs’ to a topic. Thus, a document is a mixture of topics. Latent Dirichlet Allocation utilizes a mixture model made up of multiple normal curves; here is an example:

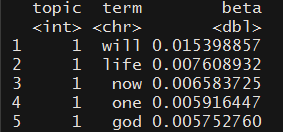


## Figure 11: Normal Mixed Model Example

The transformation that we chose to perform on our model was TF-IDF (term frequency-inverse document frequency). TF-IDF is a special case of Term Frequency. Term Frequency - the untransformed version of TF-IDF - says that if a song had many lyrics that belonged in a particular genre, we should assign this song to that particular genre. This assumes that all words are weighted evenly. However, it makes sense to say that some words are more a more decisive factor than others. To compensate for this, TF-IDF gives each term in lyrics a weight “proportionate to the number of times it appears in the document, but is offset by the frequency of the term in all songs” (Wikipedia).

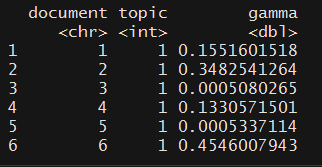
# 5-Topic LDA Model

The LDA model that we created assigned each term a beta value. Beta represents the probability that a word belongs to a genre.



## Figure 12: Beta

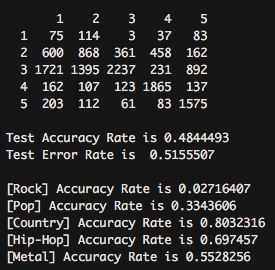
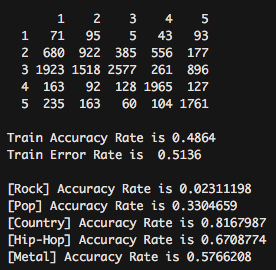
We can interpret this example displayed from our document term matrix. The word ‘will’ has a ~0.01539 beta value for topic 1. In other words, the word will appears in 1.5% of documents of topic 1. In addition to Beta, we can also explore Gamma.

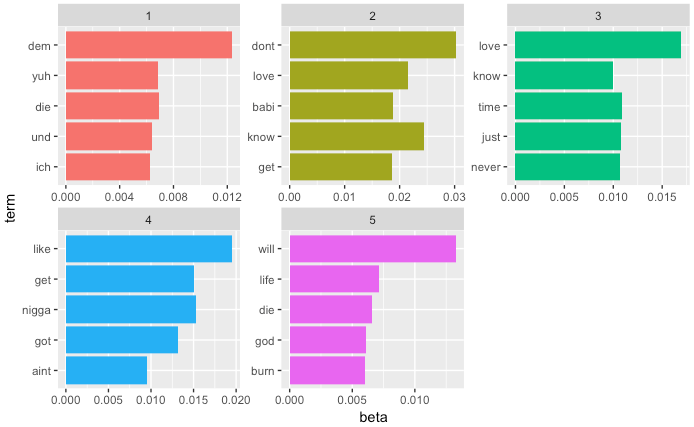


## Figure 13: Gamma

Here, we can see that in the first row, document 1 has a gamma value of ~0.15516. This means that 15.516% of document 1 belongs to topic 1.

Running the model and comparing our unsupervised topic model to the actual genres from our raw data, this is what we see:



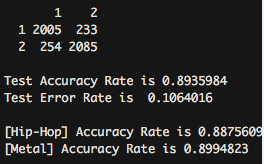
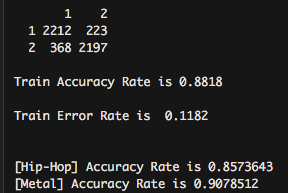


## Figure 15: Top Words by Topic (Genre)

We can see that the low accuracy of this model suggests that there are no significant differences between the vocabulary profiles of 3 of our genres. However, we cannot be sure which genres are being labelled as which number topic because this is an unsupervised model.

# 2-Topic LDA Model

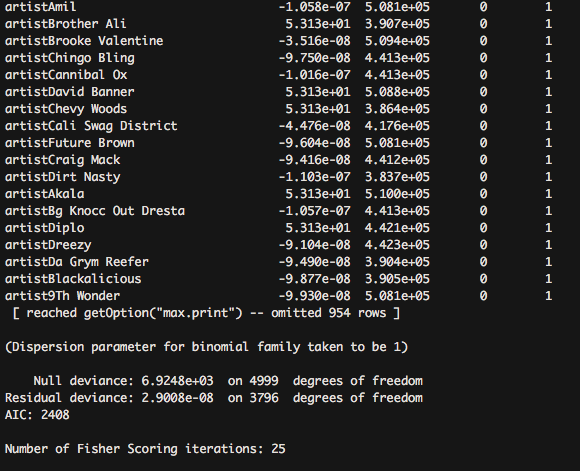
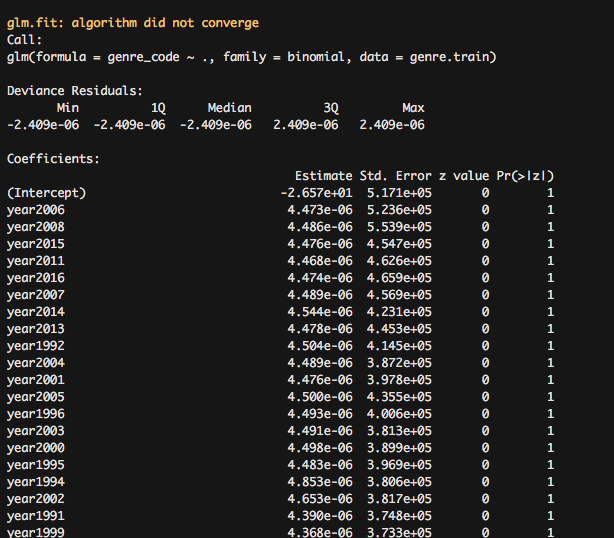
For our 2-Topic LDA model, we created a subset with 5000 observations from Hip-Hop and Metal, respectively. We followed the same procedure for our 5-topic model, where we created a corpus, a DTM, and ran LDA on these matrices. Following the LDA, we obtained the guessed topics and mutated them back into the 2-Topic subset. Again, we compared the assigned genres to the true genres and through a confusion matrix; we found our model's accuracy and error rates.



## Figure 16: 2-Topic LDA Results

# 2-Topic GLM Model

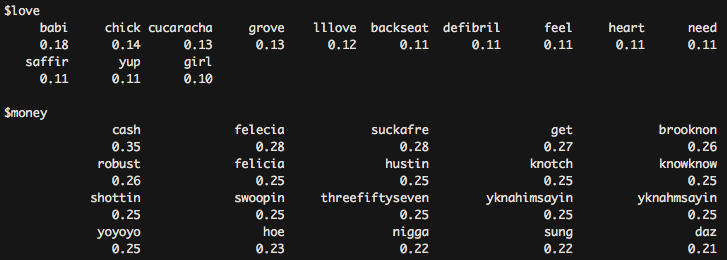
In order to prepare the data for a generalized linear model, we had to transform the variables artist and year into factors. The variable ‘genre\_code’ was created and mutated into the final subset, where Hip-Hop observations took the value of 1 and Metal observations took the value of 0. We subset from our final subset and selected only year, artist, and wordcount as regressors for genre\_code. From here we split our data into a test and training set and fit a GLM model.



## Figure 17: GLM Deviance from Residuals: Top (left) Bottom(right)

# Word Association

The following is a fun demonstration of findAssocs() function from the {tm} package. Here we have the association between ‘love’ and ‘money’. It suggests no significant correlation.



## Figure 18: Word Associations

# Results

For our 5-Topic LDA model, we were not able to successfully predict on the test set. The accuracy of the model was quite poor, and the model was unable to distinguish Rock, Country, and Pop. We are also aware that the test-accuracy is not true. Since the model was unsupervised, there is not actually any way for us to determine what genre topic 1 actually is, although figure 15 gave us pretty good ideas. We see that in the first two rows, there are significantly fewer songs being assigned to topic 1 and 2. A large number of songs are assigned to genre number 3, meaning that 3 of our genres were too similar for any statistical TF-IDF values to manifest. However, Hip-Hop and Metal made notable impact; perhaps our dataset still contained too much noise. With Country at 81%, accuracy, we see a perplexing distribution of accuracy across the genres.

After running LDA on our 2-Topic model, we found the training accuracy rate to be 88.18% and the test accuracy rate to be 89.35%. Our model was far more successful with only 2 topics to distinguish apart from.

Our 2-Topic GLM model was a poor fit for our model. The p-values for every regressor was significant at a level of 1, the data must have been too complicated or estimating itself in the model. Large AIC indicated a poor fit model, large penalties on the model. The null deviance is large, at 6925, while the residual deviance is dropped to 2.9 x10(^-8) when including all predictors.

Given we had more time, we would continue to explore many other areas, especially areas we could not complete. These include wordclouds, term document matrix visuals, PCA, cross-validation, ROC curves, k-means cluster, and more association analysis. An interesting idea would be to compare TF with TF-IDF, but no significant improvement is expected. Had we known our dataset contained a plethora of non-English songs, we would have attempted to analyze only English songs. We also realized there were no appropriate tools for supervised learning topic modeling. Although we had the true genre raw data, we were unable to use it for any machine learning. The true genres only served to compare with our model predictions. If we had the resources to work with bigger data, we would have ran more data.

# References

* **Special thanks to Professor Franks for his advising this project!**
* Kaggle, <https://www.kaggle.com/gyani95/380000-lyrics-from-metrolyrics/data>
* Julia Silge and David Robinson, 2017. “Text Mining with R”, https://www.tidytextmining.com/
* Philip Murphy, 2017. “Basic Text Mining in R” <https://rstudio-pubs-static.s3.amazonaws.com/265713_cbef910aee7642dc8b62996e38d2825d.html>
* [www.jstor.org/stable/2283270?seq=1#page\_scan\_tab\_contents](http://www.jstor.org/stable/2283270?seq=1#page_scan_tab_contents)
* <https://cran.r-project.org/web/packages/topicmodels/vignettes/topicmodels.pdf>
* <https://cran.r-project.org/web/packages/tm/vignettes/tm.pdf>
* <https://cran.r-project.org/web/packages/tidytext/vignettes/tidytext.html>
* <https://cran.r-project.org/web/packages/dplyr/dplyr.pdf>
* https://cran.r-project.org/web/packages/MASS/MASS.pdf
* <https://cran.r-project.org/web/packages/tm/vignettes/tm.pdf>
* <https://campus.datacamp.com/courses/intro-to-text-mining-bag-of-words/battle-of-the-tech-giants-for-talent?ex=11>
* [https://cran.r-project.org/web/packages/LSAfun/LSAfun.pdf](https://www.google.com/url?q=https%3A%2F%2Fcran.r-project.org%2Fweb%2Fpackages%2FLSAfun%2FLSAfun.pdf&sa=D&sntz=1&usg=AFQjCNFST6AziG2gF-GMNiF0lzcVUQTEvw)
* <https://stackoverflow.com/questions/910793/detect-encoding-and-make-everything-utf-8#answer-3479832>
* <https://stackoverflow.com/questions/33193152/unable-to-convert-a-corpus-to-data-frame-in-r>
* <https://stackoverflow.com/questions/37328244/how-to-remove-crazy-characters-like-002%C3%BF%C3%BE%C3%83%C3%83%C3%85-%C3%A2%E2%82%AC%C3%83%C2%A8%C3%83%C2%A5%C3%A2%E2%82%AC-from-text-in-r>
* <http://www.textasdata.com/2015/02/encoding-headaches-emoticons-and-rs-handling-of-utf-816/>
* <https://rstudio-pubs-static.s3.amazonaws.com/265713_cbef910aee7642dc8b62996e38d2825d.html>
* https://en.wikipedia.org/wiki/Tf%E2%80%93idf

# Appendix

---

title: "Lyrics"

author: "Josue Garcia and Harvey Lao"

date: "12/1/2017"

output: pdf\_document

---

```{r, warning = FALSE}

library(ngram)

library(readr)

library(tm)

library(stringr)

library(dplyr)

library(tidyverse)

library(ROCR)

library(tree)

library(utils)

library(maptree)

library(class)

library(lattice)

library(tidytext)

library(ggplot2)

library(topicmodels)

library(SnowballC)

library(MASS)

library(RColorBrewer)

library(wordcloud)

library(biclust)

library(cluster)

library(igraph)

library(cluster)

library(fpc)

knitr::opts\_chunk$set(echo = FALSE)

```

# Import Data

```{r}

songs.og <- read.csv("lyrics.csv", stringsAsFactors = FALSE, encoding = "UTF-8")

```

# Copy Variable for Preprocessing

```{r}

songs = songs.og

```

# Subsetting Genres

```{r removing most genres chunk}

#Clean up artist, song name, and lyrics

library(dplyr)

#subset data, removing genres

songs.sub11 = filter(songs, (songs$genre!="Jazz"))

songs.sub10 = filter(songs.sub11, (songs.sub11$genre!="Not Available"))

songs.sub9 = filter(songs.sub10, (songs.sub10$genre!="Folk"))

songs.sub8 = filter(songs.sub9, (songs.sub9$genre!="Other"))

songs.sub7 = filter(songs.sub8, (songs.sub8$genre!="Indie"))

songs.sub6 = filter(songs.sub7, (songs.sub7$genre!="R&B"))

songs.sub5 = filter(songs.sub6, (songs.sub6$genre!="Electronic"))

#remove songs with no lyrics

songs.sub5 = songs.sub5[which(nchar(songs.sub5$lyrics) != 0), ]

```

# Removing Non-Contextual Characters

```{r removing non-contextual chunk}

songs.sub5$song <- str\_replace\_all(songs.sub5$song, "-", " ")

songs.sub5$song<-  str\_to\_title(songs.sub5$song)

songs.sub5$artist <- str\_replace\_all(songs.sub5$artist, "-", " ")

songs.sub5$artist <- str\_to\_title(songs.sub5$artist)

songs.sub5$lyrics <- str\_replace\_all(songs.sub5$lyrics, "\n", " ")

songs.sub5$lyrics <- tolower(songs.sub5$lyrics)

songs.sub5$lyrics <- str\_replace\_all(songs.sub5$lyrics, c("verse", "chorus","bridge","hook","intro", "the", "you", "and", "that", "your", "I'm", "for", "with") , " ")

```

# Word Count

```{r word count chunk}

library(ngram)

#Create word\_count vector

word\_count = c()

#Fill in the values of word\_count

for (i in 1:nrow(songs.sub5)){

 word\_count[i] = wordcount(songs.sub5$lyrics[i])

}

#Mutate word\_count into a new column

songs.sub5 = songs.sub5 %>%

 mutate(wordcount = word\_count)

cat( " Average word count in lyrics is equal to ", mean(word\_count),"\n.")

cat(" After removing all Jazz, Not Available, Folk, Other, Indie, R&B, Electronic, and songs with no lyrics, the song subset now has", nrow(songs.sub5), " observations.\n")

```

# Analyze Word Count

```{r analyzing word count chunk}

set.seed(1)

# number of songs in [0.200]

under200 = filter(songs.sub5, songs.sub5$wordcount <=200)

nrow(under200)

# number of songs in [0,150]

under150 = filter(songs.sub5, songs.sub5$wordcount<=150)

nrow(under150)

# number of songs in [0,100]

under100 = filter(songs.sub5, songs.sub5$wordcount<=100)

nrow(under100)

# number of songs in [0,50]

under50 = filter(songs.sub5, songs.sub5$wordcount<=50)

nrow(under50)

cat ("Since our average word count is", mean(songs.sub5$wordcount), "characters in lyrics, this is why we choose to remove lyrics with less than 100 characters. By removing all songs with lyrics under 100 characters, we remove only", nrow(under100)/nrow(songs.sub5)\*100, "% of the total data. \n")

#remove songs with lyrics under 150

songs.sub5 <- filter(songs.sub5, songs.sub5$wordcount>=100)

cat ("Number of songs with lyrics under 200:", nrow(under200), "\n")

cat ("Number of songs with lyrics under 150:", nrow(under150), "\n")

cat ("Number of songs with lyrics under 100:", nrow(under100), "\n")

cat ("Number of songs with lyrics under 50:", nrow(under50), "\n")

```

```{r table genre chunk}

#Table of genres after removing:

     #genres

     #songs with wordcount < 150

       #including songs with wordcount=0

table(songs.sub5$genre)

```

Let's sample 6000 observations from each genre as a training set.

```{r genre subsetting chunk}

#sample 2000 obs from each

set.seed(1)

#country = subset(songs.sub5, songs.sub5$genre == "Country")

country.sub = data.frame(sample\_n(subset(songs.sub5, songs.sub5$genre == "Country"), size = 6000))

hiphop.sub = data.frame(sample\_n(subset(songs.sub5, songs.sub5$genre == "Hip-Hop"), size = 6000))

metal.sub = data.frame(sample\_n(subset(songs.sub5, songs.sub5$genre == "Metal"), size = 6000))

pop.sub = data.frame(sample\_n(subset(songs.sub5, songs.sub5$genre == "Pop"), size = 6000))

rock.sub = data.frame(sample\_n(subset(songs.sub5, songs.sub5$genre == "Rock"), size = 6000))

final.subset = bind\_rows(country.sub, hiphop.sub, metal.sub, pop.sub, rock.sub)

```

```{r intensive cleaning chunk}

library(tm)

library(tidyr)

#Intensive cleaning lyrics (Punctuation, Stem, StopWords)

lyrics <- VCorpus(VectorSource(final.subset$lyrics))

lyrics <- tm\_map(lyrics, removePunctuation)

lyrics <- tm\_map(lyrics, removeNumbers)

lyrics <- tm\_map(lyrics, tolower)

lyrics <- tm\_map(lyrics, PlainTextDocument)

lyrics <- tm\_map(lyrics, removeWords, stopwords('english'))

lyrics <- tm\_map(lyrics, PlainTextDocument)

lyrics <- tm\_map(lyrics, stemDocument)

lyrics <- tm\_map(lyrics, stripWhitespace)

lyrics <- tm\_map(lyrics, PlainTextDocument)

#lyrics <- tm\_map(lyrics, removeWords, c("verse", "chorus","bridge","hook","intro", "the", "you", "and", "that", "your", "I'm", "for", "with"))

#creating lyrics VCorpus to dataframe

lyrics.dataframe<-data.frame(text=unlist(sapply(lyrics, `[`, "content")), stringsAsFactors=F)

final.subset = final.subset %>%

mutate(lyrics\_cleaned = lyrics.dataframe$text)

```

```{r removing trash lyrics chunk, results=FALSE}

#removing non-english words pt.19

#finally.subset is a copy of final.subset

finally.subset = final.subset

trash.lyrics <- tools::showNonASCII(finally.subset$lyrics\_cleaned)

bad <- which(finally.subset$lyrics\_cleaned %in% trash.lyrics)

finally.subset <- finally.subset[-bad,]

```

```{r removing sparcity before train/test sets}

#start as data frame

corpFinally <- VCorpus(VectorSource(finally.subset$lyrics\_cleaned))

dtmFinally = DocumentTermMatrix(corpFinally, list(globaln = c(2, Inf), weightTfIdf = TRUE))

#dtmsFinally = removeSparseTerms(dtmFinally, .97)

#rowTotals = apply(dtmsFinally, 1, sum) #Find the sum of words in each Document

#dtmsFinally = dtmsFinally[rowTotals> 0, ] #remove all docs without words

```

#Subsetting Training and Data Sets

```{r indices chunk}

set.seed(1)

test.indices = sample(1:nrow(finally.subset), 15000)

songs.train = finally.subset[test.indices,]

songs.test =finally.subset[-test.indices,]

print(dim(songs.train))

print(dim(songs.test))

```

On with the modelling:

```{r test/train objects chunk}

library(tm)

#Corpus objects

corpTrain <- VCorpus(VectorSource(songs.train$lyrics\_cleaned))

corpTest <- VCorpus(VectorSource(songs.test$lyrics\_cleaned))

#DTM

dtmTrain = DocumentTermMatrix(corpTrain, list(globaln = c(2, Inf), weightTfIdf = TRUE))

dtmTest = DocumentTermMatrix(corpTest, list(globaln = c(2, Inf), weightTfIdf = TRUE))

```

#Exploring our entire Data

```{r before removing sparsity}

freq <- colSums(as.matrix(dtmFinally))

length(freq) #this should display the number of terms in our whole training set

head(table(freq), 10) #displays # of words of bottom freq

tail(table(freq), 10) #displays # of words of top freq

#Let's see which terms are most/least frequent

freq.sort <- sort(colSums(as.matrix(dtmTrain)), decreasing=TRUE)

head(freq.sort, 10)

tail(freq.sort, 10)

```

#Removing sparse terms and re-observe term frequencies

```{r}

#Removing Sparse Terms

# dtmsTrain = removeSparseTerms(dtmTrain, .97)

# dtmsTest = removeSparseTerms(dtmTest, .97)

# tdmsTrain = removeSparseTerms(tdmTrain, .97)

# tdmsTest = removeSparseTerms(tdmTest, .97)

# freq.sparse <- sort(colSums(as.matrix(dtmsTrain)), decreasing=TRUE)

# head(freq.sparse, 10) #display most frequent terms

# tail(freq.sparse, 10) #display least frequent terms

```

<!-- Cluster Dendogram -->

<!-- ```{r} -->

<!-- # d <- dist(t(dtmTrain), method="euclidian") -->

<!-- # fit <- hclust(d=d, method="complete")   # for a different look try substituting: method="ward.D" -->

<!-- # fit -->

<!-- # plot.new() -->

<!-- # plot(fit, hang=-1) -->

<!-- # groups <- cutree(fit, k=5)   # "k=" defines the number of clusters you are using -->

<!-- # rect.hclust(fit, k=5, border="red") # draw dendogram with red borders around the 5 clusters -->

<!-- ``` -->

K-Means Clustering

```{r}

# library(fpc)

# d = dist(t(dtmTrain), method = "euclidean")

# kfit = kmeans(d, 5)

# clusplot(as.matrix(d), kfit$cluster, color=T, shade=T, labels = 1, lines=0)

```

LDA (Latent Dirichlet Allocation)

```{r}

set.seed(2)

library(topicmodels)

songs.lda = LDA(dtmTrain, 5)

post.lda = posterior(songs.lda, dtmTest)

#post.lda

songs.topics <- tidytext::tidy(songs.lda, matrix = "beta")

#this shows the probability 'beta' a word is in a topic

terms <- as.data.frame(t(posterior(songs.lda)$terms)) #topics 1-5 are our genres

head(terms, 10) #shows probability "beta" of word belonging to topic

```

mutating and interpreting results

```{r}

train.lda = songs.lda

train.topics = topics(train.lda)

songs.train = songs.train %>%

mutate(assigned\_genre = train.topics)

songs.train = songs.train %>%

 mutate(true\_genre = ifelse(genre=="Hip-Hop", 1, ifelse(genre=="Pop", 2, ifelse(genre=="Country", 3, ifelse(genre=="Rock", 4, 5)))))

error.train = table(songs.train$assigned\_genre, songs.train$true\_genre)

error.train #True genre is the column. Assigned genre is the row.

cat("\nTrain Accuracy Rate is", sum(diag(error.train))/sum(error.train), "\n") #train accuracy rate

cat("\nTrain Error Rate is ", 1-sum(diag(error.train))/sum(error.train), "\n \n") #train error rate

cat("\n[Hip-Hop] Accuracy Rate is", error.train[1,1]/sum(error.train[,1]))

cat("\n[Pop] Accuracy Rate is", error.train[2,2]/sum(error.train[,2]))

cat("\n[Country] Accuracy Rate is", error.train[3,3]/sum(error.train[,3]))

cat("\n[Rock] Accuracy Rate is", error.train[4,4]/sum(error.train[,4]))

cat("\n[Metal] Accuracy Rate is", error.train[5,5]/sum(error.train[,5]))

```

Run Predictions

```{r}

#Predict on test set

test.topics = posterior(songs.lda, dtmTest)

test.topics = apply(test.topics$topics, 1, which.max)

songs.test = songs.test %>%

 mutate(assigned\_genre = test.topics)

songs.test = songs.test %>%

 mutate(true\_genre = ifelse(genre=="Hip-Hop", 1, ifelse(genre=="Pop", 2, ifelse(genre=="Country", 3, ifelse(genre=="Rock", 4, 5)))))

error.test = table(songs.test$assigned\_genre, songs.test$true\_genre)

error.test #True genre is the column. Assigned genre is the row.

cat("\nTest Accuracy Rate is", sum(diag(error.test))/sum(error.test), "\n") #test accuracy rate

cat("Test Error Rate is ", 1-sum(diag(error.test))/sum(error.test), "\n") #test error rate

cat("\n[Hip-Hop] Accuracy Rate is", error.test[1,1]/sum(error.test[,1]))

cat("\n[Pop] Accuracy Rate is", error.test[2,2]/sum(error.test[,2]))

cat("\n[Country] Accuracy Rate is", error.test[3,3]/sum(error.test[,3]))

cat("\n[Rock] Accuracy Rate is", error.test[4,4]/sum(error.test[,4]))

cat("\n[Metal] Accuracy Rate is", error.test[5,5]/sum(error.test[,5]))

```

Here we have topics 1 and 5 (Hip-Hop and Metal) as the most assigned topics, thus they are very similar in vocabulary. We will now subset the data, only keeping

these 2 genres, and attempting to identify them from each other based on the lyrics.

```{r}

#sample 5000 obs from each

set.seed(1)

hiphop.sub2 = data.frame(sample\_n(subset(songs.sub5, songs.sub5$genre == "Hip-Hop"), size = 5000))

metal.sub2 = data.frame(sample\_n(subset(songs.sub5, songs.sub5$genre == "Metal"), size = 5000))

final.subset2 = bind\_rows(hiphop.sub2, metal.sub2)

```

```{r intensive cleaning chunk}

library(tm)

library(tidyr)

#Intensive cleaning lyrics (Punctuation, Stem, StopWords)

lyrics2 <- VCorpus(VectorSource(final.subset2$lyrics))

lyrics2 <- tm\_map(lyrics2, removePunctuation)

lyrics2 <- tm\_map(lyrics2, removeNumbers)

lyrics2 <- tm\_map(lyrics2, tolower)

lyrics2 <- tm\_map(lyrics2, PlainTextDocument)

lyrics2 <- tm\_map(lyrics2, removeWords, stopwords('english'))

lyrics2 <- tm\_map(lyrics2, PlainTextDocument)

lyrics2 <- tm\_map(lyrics2, stemDocument)

lyrics2 <- tm\_map(lyrics2, stripWhitespace)

lyrics2 <- tm\_map(lyrics2, PlainTextDocument)

#creating lyrics VCorpus to dataframe

lyrics2.dataframe<-data.frame(text=unlist(sapply(lyrics2, `[`, "content")), stringsAsFactors=F)

final.subset2 = final.subset2 %>%

mutate(lyrics2\_cleaned = lyrics2.dataframe$text)

```

```{r removing trash lyrics chunk, results=FALSE}

#removing non-english words pt.19

#finally.subset is a copy of final.subset

finally.subset2 = final.subset2

trash.lyrics2 <- tools::showNonASCII(finally.subset2$lyrics2\_cleaned)

bad2 <- which(finally.subset2$lyrics2\_cleaned %in% trash.lyrics2)

finally.subset2 <- finally.subset2[-bad2,]

```

```{r removing sparcity before train/test sets}

#start as data frame

corpFinally2 <- VCorpus(VectorSource(finally.subset2$lyrics2\_cleaned))

dtmFinally2 = DocumentTermMatrix(corpFinally2, list(globaln = c(2, Inf), weightTfIdf = TRUE))

dtmsFinally2 = removeSparseTerms(dtmFinally2, .97)

rowTotals2 = apply(dtmsFinally2, 1, sum) #Find the sum of words in each Document

dtmsFinally2 = dtmsFinally2[rowTotals2> 0, ] #remove all docs without words

```

#Subsetting Training and Data Sets pt.2

```{r indices chunk}

set.seed(2)

test.indices2 = sample(1:nrow(finally.subset2), 5000)

...