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.

Josue Garcia
Harvey Lao

1/8/18

Contents

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:

	A	B	C	D	E	F
1	index	song	year	artist	genre	lyrics
2	0	ego-remi	2009	beyonce-I	Pop	Oh baby,
3	1	then-tell-	2009	beyonce-I	Pop	playin'
4	2	honesty	2009	beyonce-I	Pop	If you
5	3	you-are-m	2009	beyonce-I	Pop	Oh oh oh
6	4	black-cult	2009	beyonce-I	Pop	Party the
7	5	all-i-could	2009	beyonce-I	Pop	I heard
8	6	once-in-a-	2009	beyonce-I	Pop	This is
9	7	waiting	2009	beyonce-I	Pop	Waiting,
10	8	slow-love	2009	beyonce-I	Pop	[Verse
11	9	why-don-	2009	beyonce-I	Pop	N-n-now,
12	10	save-the-I	2009	beyonce-I	Pop	I lay

362567	362226	you-ll-nev	2015	dee-smgn	Other	You'll
362568	362227	too-good-	2012	edens-ed	Country	You
362569	362228	skinny-dip	2012	edens-ed	Country	I warned
362570	362229	cherry-pie	2012	edens-ed	Country	To my
362571	362230	feels-so-r	2012	edens-ed	Country	It was
362572	362231	swingin-d	2012	edens-ed	Country	You've
362573	362232	who-am-i-	2012	edens-ed	Country	I gotta
362574	362233	liar	2012	edens-ed	Country	I helped
362575	362234	last-suppe	2012	edens-ed	Country	Look at
362576	362235	christ-alor	2012	edens-ed	Country	When I
362577	362236	amen	2012	edens-ed	Country	I heard

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.

	index	song	year	artist	genre	lyrics	wordcount
1	0	Ego Remix	2009	Beyonce Knowles	Pop	oh baby, how you doing? you know i'm gonna cut right to t...	433
2	1	Then Tell Me	2009	Beyonce Knowles	Pop	playin' everything so easy, it's like you seem so sure. still you...	258
3	2	Honesty	2009	Beyonce Knowles	Pop	if you search for tenderness it isn't hard to find you can have ...	170
4	3	You Are My Rock	2009	Beyonce Knowles	Pop	oh oh oh i, oh oh oh i [verse 1:] if i wrote a book about wher...	522
5	4	Black Culture	2009	Beyonce Knowles	Pop	party the people, the people the party it's popping no sitting ...	312
6	5	All I Could Do Was Cry	2009	Beyonce Knowles	Pop	i heard church bells ringing i heard a choir singing i saw my l...	139
7	6	Once In A Lifetime	2009	Beyonce Knowles	Pop	this is just another day that i would spend waitin' for the righ...	281
8	7	Waiting	2009	Beyonce Knowles	Pop	waiting, waiting, waiting, waiting waiting, waiting, waiting, ...	378
9	8	Slow Love	2009	Beyonce Knowles	Pop	[verse 1:] i read all of the magazines while waiting around yo...	435
10	9	Why Don T You Love Me	2009	Beyonce Knowles	Pop	n-n-now, honey you better sit down and look around 'cause ...	339

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.

index	song	year	artist	genre	lyrics
56	Mine	2013	Beyonce Knowles	Pop	[verse 1: beyonce] i've been watching for the signs took a tri...
57	Superpower	2013	Beyonce Knowles	Pop	[verse 1] when palm of my two hands hold each or that feels...
58	Haunted	2013	Beyonce Knowles	Pop	[intro: presenter] the winner is beyonce knowles female pop ...
59	Flawless	2013	Beyonce Knowles	Pop	[intro] your challengers are a young group from houston wel...
60	Partition	2013	Beyonce Knowles	Pop	part 1: "yoncã©" [intro] let me hear you say "hey, ms. carter!...
61	Ghost	2013	Beyonce Knowles	Pop	[intro: presenter] the winner is beyonce knowles female pop ...
62	Xo	2013	Beyonce Knowles	Pop	[verse 1] your love is bright as ever even in the shadows baby...
63	Single Ladies Put A Ring On It	2013	Beyonce Knowles	Pop	all the single ladies (all the single ladies) all the single ladies (...
64	On The Run	2013	Beyonce Knowles	Pop	[intro: beyoncã©] who wants that perfect love story any way...

Figure 4: Example of Final Subset

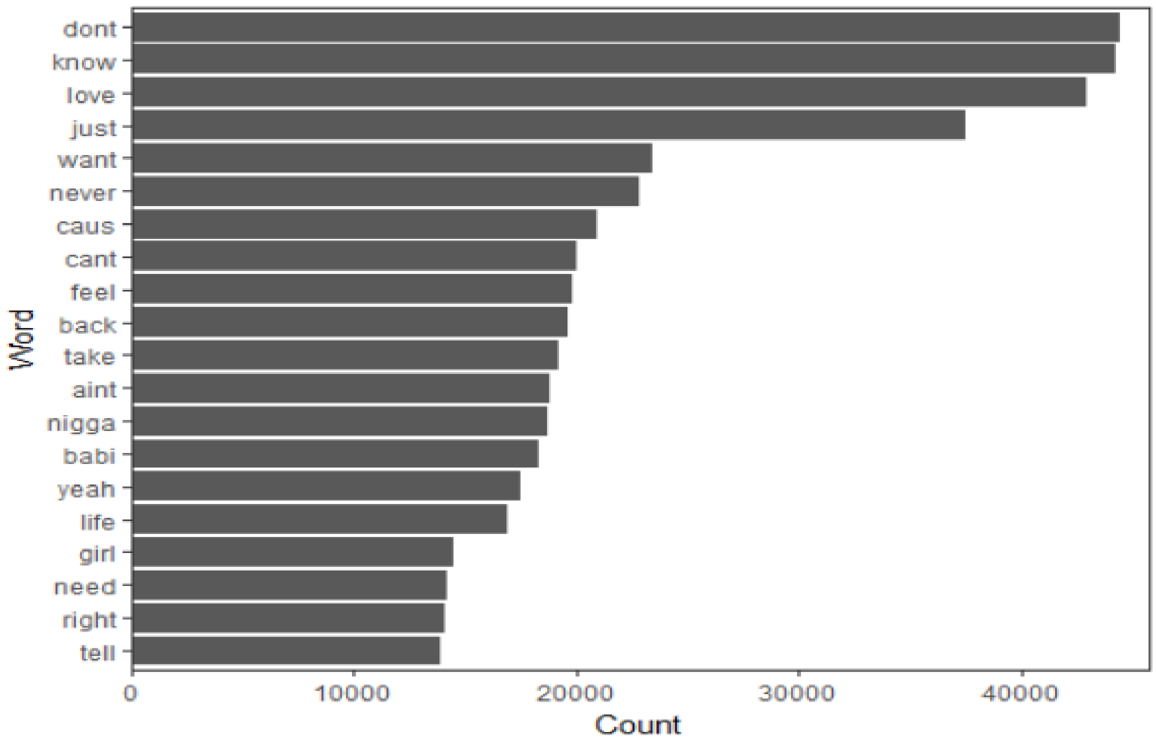

Figure 5: Top Words of Final Subset

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).

index	song	year	artist	genre	lyrics
72240	Ay Mama	2008	Chayanne	Pop	â¡esssso! â¡ahiiiii! â¡venga! â¡a gozar! â¡dale, dale! why es
73812	Feliz Navidad 6	2014	Arcangel	Rock	(â¿y quiã©n les dijo que esto se habâa acabado? feliz na
196614	Pois E	2015	Clda	Pop	ã<U+0083>â<U+0089> o caos, ã<U+0083>â© o caos,
40800	Felicitrk	2014	Al Bano	Pop	ã<U+0088> tenersi per mano andare lontano, la felicitã
345136	8 Grenudos Locos	2006	Brujeria	Metal	ã<U+0093>iganme bien esto es para ustedes hermanos
55446	Casa	2015	Emicida	Hip-Hop	ã<U+0094>ã´ã´ ã´ã´ lã¡ fora ã© selva a sã³s entre luz e tr
332812	Aa R Pce Way Home	2015	Andy Lau	Pop	å<U+009B><U+009E>å®¶ç<U+009A><U+0084>è·¯ æ
275943	Apar Meni	2015	Aygun Kaza Mova	Pop	å<U+009E>irin-å<U+009F>irin sã¶hbé<U+0099>t etdik
290635	Pink Stars In The Sky	2013	Annabelle	Pop	aaahhhhh make it shine like pink! (pink, pink) pink stars
145634	Persian Wine	2012	Asap Mob	Hip-Hop	a$ap murder clan, ride with the mob twinkle in the eye
215159	Babylon Vampires	2016	Avantasia	Metal	aaron: a lot of good advice nobody's keeping for thems
261547	Infester	2007	Exhumed	Metal	your abdomen punctured just beneath the chest to acc

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.

it ain't nothin' for me to ball on you it ain't nothin' for me to spoil you if i adore you, ma give you that theory i wanna be with you, i wanna be with you i wanna be with you, i wanna be with you i wanna be with you, i wanna be with you i wanna be with you, i wanna be with you i wanna be where the commas will be but i need a hood nigga with the llama degree get the limited edition, audemars, it could be in a pivotal position, gotta pardon the fee 'cause he bought a couple bags and he sent a couple whips and he took a couple trips, then its dinner and a myx and he's never with no other bitch, fronting like he's slick 'cause it's levels to this shit and she could never be nic niggas be fallin' in love with this pussy mean stew chicken, and bake him a couple of cookies dick on veteran, ain't fucking with rookies saw the high school video now he wanna play hookie baddest bitch, the catalyst ain't never been done, bitch i added this nah, i ain't gotta shoot, i got mad assists 'bout to put a couple pieces on the mannequin got a big billboard out in madison at the trump, and you bitches at the radisson got the .22 on me, and it's thin shoot movies, jennifer aniston you decide you'll be mine you can come inside you the type that can make me prioritize hittin' my phone, it's alright hittin' my phone, it's alright you reply, what's your sign? you're a gemini you deny that you're shy, maybe we should slide? i wanna be with you i wanna be with you, baby ballin' on you too easy, splurgin' on you too easy buyin' purses too easy, payin' bills too easy i wanna be with you, i wanna be with you i wanna be with you, i wanna be with you i wanna be with you, i wanna be with you i wanna be with you, i wanna be with you ballin' on you too easy, splurgin' on you too easy buyin' cars too easy, poppin' bottles too easy i wanna be with you, i wanna be with you

aint nothin ball aint nothin spoil ador ma give theori wanna wanna wanna wanna wanna wanna wanna wanna wanna comma will need hood nigga llama degre get limit edit audemar pivot posit gotta pardon fee caus bought coupl bag sent coupl whip took coupl trip dinner myx hes never bitch front like hes slick caus level shit never nic nigga fallin love pussi mean stew chicken bake coupl cooki dick veteran aint fuck rooki saw high school video now wanna play hooki baddest bitch catalyst aint never done bitch ad nah aint gotta shoot got mad assist bout put coupl piec mannequin got big billboard madison trump bitch radisson got thin shoot movi jennif aniston decid youll mine can come insid type can make priorit hittin phone alright hittin phone alright repli what sign your gemini deni your shi mayb slide wanna wanna babi ballin easi splurgin easi buyin purs easi payin bill easi wanna wanna wanna wanna wanna wanna wanna wanna ballin easi splurgin easi buyin car easi poppin bottl easi wanna wanna everyth brand new make bad bitch shorti hit club throw forti hat bent like uhh chain drip like water car paint like tar ma sex harder bitch let go hand gave key soon bought voom voom oh yeah big bank real money right boy dead broke two year check look aint even know big deal site act right shop hard pack light ho chick get play talk cocain white tee rope chain blow roof back kurt cobain phantom cost like four dollar flo seat hoe holla underground pimp shit smoke one port arthur ballin easi splurgin easi buyin purs easi payin bill easi wanna wanna wanna wanna wanna wanna wanna wanna ballin easi splurgin easi buyin car easi poppin bottl easi wanna wanna everyth brand new suffer success wit great best ever wanna wanna wanna wanna wanna wanna wanna wanna wanna

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.

```
freq
    1     2     3     4     5     6     7     8
19906  5606  2681  1703  1172   933   671   565
    9    10
  463   410
```

Figure 7: Top Frequencies

```
 freq
 7139  7715  8373  9221  9976  9990 11430 11805
    1     1     1     1     1     1     1     1
11896 13021
    1     1
```

Figure 8: Bottom Frequencies

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

```
 like  love  dont  know   get
13021 11896 11805 11430  9990
 just   got   now   can  time
 9976  9221  8373  7715  7139
```

Figure 9: Top Words

```
zweihundertachtzig               zweit
                 1                   1
           zweiten    zweiundzwanzigst
                 1                   1
           zwicken             zwiegen
                 1                   1
     zwielichtigen            zwischen
                 1                   1
            zxulth              zydeco
                 1                   1
```

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:

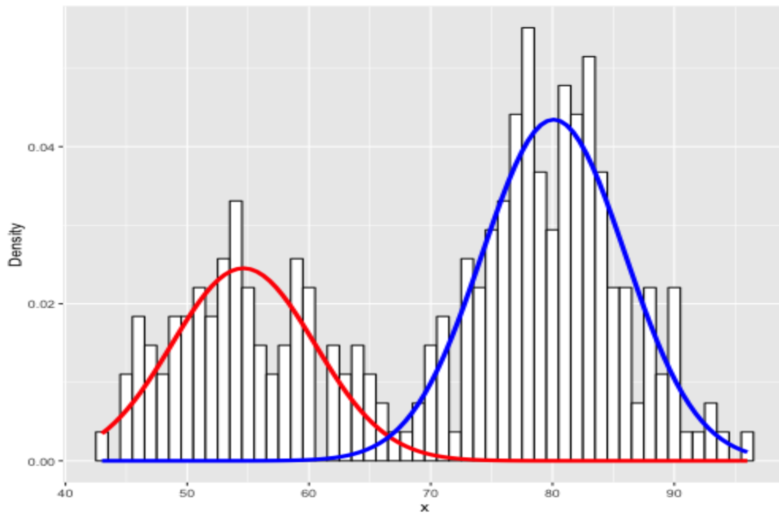

Figure 11: Normal Mixed Model Example

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.

	topic	term	beta
	<int>	<chr>	<dbl>
1	1	will	0.015398857
2	1	life	0.007608932
3	1	now	0.006583725
4	1	one	0.005916447
5	1	god	0.005752760

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.

	document	topic	gamma
	<chr>	<int>	<dbl>
1	1	1	0.1551601518
2	2	1	0.3482541264
3	3	1	0.0005080265
4	4	1	0.1330571501
5	5	1	0.0005337114
6	6	1	0.4546007943

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:

```
     1    2    3    4    5
1   71   95    5   43   93
2  680  922  385  556  177
3 1923 1518 2577  261  896
4  163   92  128 1965  127
5  235  163   60  104 1761

Train Accuracy Rate is 0.4864
Train Error Rate is  0.5136

[Rock] Accuracy Rate is 0.02311198
[Pop] Accuracy Rate is 0.3304659
[Country] Accuracy Rate is 0.8167987
[Hip-Hop] Accuracy Rate is 0.6708774
[Metal] Accuracy Rate is 0.5766208
```

```
     1    2    3    4    5
1   75  114    3   37   83
2  600  868  361  458  162
3 1721 1395 2237  231  892
4  162  107  123 1865  137
5  203  112   61   83 1575

Test Accuracy Rate is 0.4844493
Test Error Rate is  0.5155507

[Rock] Accuracy Rate is 0.02716407
[Pop] Accuracy Rate is 0.3343606
[Country] Accuracy Rate is 0.8032316
[Hip-Hop] Accuracy Rate is 0.697457
[Metal] Accuracy Rate is 0.5528256
```

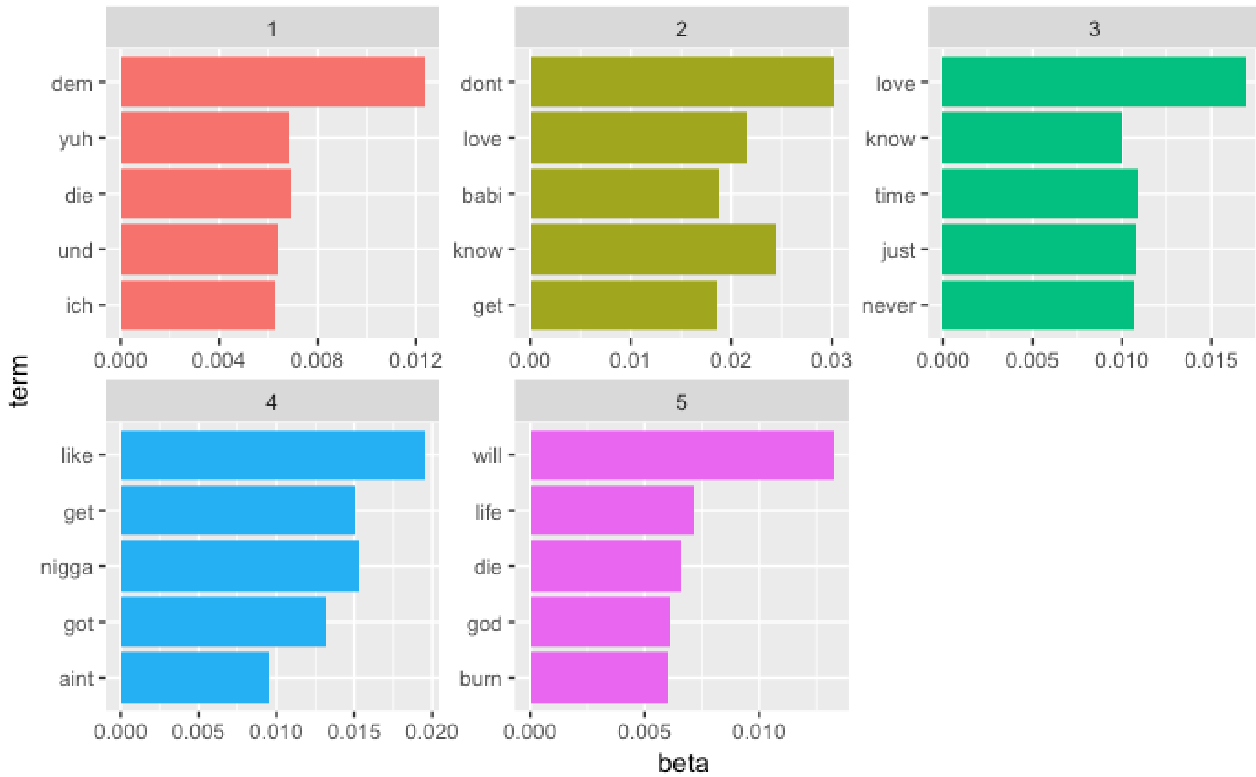

Figure 15: Top Words by Topic (Genre)

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.

```
     1    2
  1 2212  223
  2  368 2197

 Train Accuracy Rate is 0.8818

 Train Error Rate is  0.1182


 [Hip-Hop] Accuracy Rate is 0.8573643
 [Metal] Accuracy Rate is 0.9078512
```

```
      1    2
 1 2005  233
 2  254 2085

Test Accuracy Rate is 0.8935984
Test Error Rate is  0.1064016

[Hip-Hop] Accuracy Rate is 0.8875609
[Metal] Accuracy Rate is 0.8994823
```

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.

```
glm.fit: algorithm did not converge
Call:
glm(formula = genre_code ~ ., family = binomial, data = genre.train)

Deviance Residuals:
      Min          1Q      Median          3Q         Max
-2.409e-06  -2.409e-06  -2.409e-06   2.409e-06   2.409e-06

Coefficients:
                               Estimate Std. Error z value Pr(>|z|)
(Intercept)                  -2.657e+01  5.171e+05       0        1
year2006                      4.473e-06  5.236e+05       0        1
year2008                      4.486e-06  5.539e+05       0        1
year2015                      4.476e-06  4.547e+05       0        1
year2011                      4.468e-06  4.626e+05       0        1
year2016                      4.474e-06  4.659e+05       0        1
year2007                      4.489e-06  4.569e+05       0        1
year2014                      4.544e-06  4.231e+05       0        1
year2013                      4.478e-06  4.453e+05       0        1
year1992                      4.504e-06  4.145e+05       0        1
year2004                      4.489e-06  3.872e+05       0        1
year2001                      4.476e-06  3.978e+05       0        1
year2005                      4.500e-06  4.355e+05       0        1
year1996                      4.493e-06  4.006e+05       0        1
year2003                      4.491e-06  3.813e+05       0        1
year2000                      4.498e-06  3.899e+05       0        1
year1995                      4.483e-06  3.969e+05       0        1
year1994                      4.853e-06  3.806e+05       0        1
year2002                      4.653e-06  3.817e+05       0        1
year1991                      4.390e-06  3.748e+05       0        1
year1999                      4.368e-06  3.733e+05       0        1
```

```
artistAmil                        -1.058e-07  5.081e+05       0        1
artistBrother Ali                  5.313e+01  3.907e+05       0        1
artistBrooke Valentine            -3.516e-08  5.094e+05       0        1
artistChingo Bling                -9.750e-08  4.413e+05       0        1
artistCannibal Ox                 -1.016e-07  4.413e+05       0        1
artistDavid Banner                 5.313e+01  5.088e+05       0        1
artistChevy Woods                  5.313e+01  3.864e+05       0        1
artistCali Swag District          -4.476e-08  4.176e+05       0        1
artistFuture Brown                -9.604e-08  5.081e+05       0        1
artistCraig Mack                  -9.416e-08  4.412e+05       0        1
artistDirt Nasty                  -1.103e-07  3.837e+05       0        1
artistAkala                        5.313e+01  5.100e+05       0        1
artistBg Knocc Out Dresta         -1.057e-07  4.413e+05       0        1
artistDiplo                        5.313e+01  4.421e+05       0        1
artistDreezy                      -9.104e-08  4.423e+05       0        1
artistDa Grym Reefer              -9.490e-08  3.904e+05       0        1
artistBlackalicious               -9.877e-08  3.905e+05       0        1
artist9Th Wonder                  -9.930e-08  5.081e+05       0        1
 [ reached getOption("max.print") -- omitted 954 rows ]

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6.9248e+03  on 4999  degrees of freedom
Residual deviance: 2.9008e-08  on 3796  degrees of freedom
AIC: 2408

Number of Fisher Scoring iterations: 25
```

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.

```
$love
     babi     chick cucaracha     grove    lllove  backseat  defibril      feel     heart      need
     0.18      0.14      0.13      0.13      0.12      0.11      0.11      0.11      0.11      0.11
   saffir       yup      girl
     0.11      0.11      0.10

$money
             cash              felecia              suckafre                  get             brooknon
             0.35                 0.28                  0.28                 0.27                 0.26
           robust              felicia                hustin               knotch             knowknow
             0.26                 0.25                  0.25                 0.25                 0.25
          shottin              swoopin       threefiftyseven         yknahimsayin          yknahmsayin
             0.25                 0.25                  0.25                 0.25                 0.25
           yoyoyo                  hoe                 nigga                 sung                  daz
             0.25                 0.23                  0.22                 0.22                 0.21
```

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
- 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://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)

...
````