A Lyrical Analysis of Popular Songs in the Last Three Decades

Dan Berenberg
University of Vermont

14 December 2016

Abstract

Has popular music notably changed in anyway since 1980? A 2012 Smithsonian article, citing scientific evidence (Eveleth), tends to agree with old people everywhere: it has definitely gotten worse. The article reports on a post-doctoral scholar who studied homogeneity of the timbre of music – how does it sound? But does the sound agree with the syntax?

After building a data set composed of song metadata including the name, artist, production year, lyrics, and genre an analysis of syntactical and syllabic patterns leads to some interesting information regarding the formation of music between 1980 and 2010. The simple pairwise combinations of lyrics, genre, and production year (name and artist are generally used as identifiers) are a spring of data that can be manipulated in order to produce important characteristics such as Yule coefficients and average syllabic count of a song that might lead to an avenue for (more) accurate inference.

Data

The dataset was built programmatically (all data collection code is located in data_xtraction.py) by accessing two different web APIs (Chartlyrics.com and Musixmatch.com) and a convenient repository on a website called jamrockentertainment.com and writing the information to a locally stored sqlite3 database called song_records.db (see build_db.py for database building as well as the database accessing section of data_xtraction.py [lines 251 – 297].

Each website contained crucial data for the dataset: Chartlyrics contains lyrics for hundreds of thousands of songs, Musixmatch has a wealth of metadata related to songs, namely the genre (though only allows 2000 requests per day), and jamrockentertainment.com was a source to find the top 100 song names for each year between 1980 and 2010.

Thus, the data were collected in the following way: each of the song and artist names for a given year were extracted off of the jamrockentertainment repository and the names were stored in a dictionary as key values to a dictionary associated with that song's metadata. After extracting the names, using the Chartlyrics API, each of the lyrics for each of the three thousand (100 songs * 30 years) names was requested. At this point, several data points were imputed away because unsuccessful requests lead to lyric-less data points, which are far less useful. After taking the lyrics, requests for the remaining data points were sent to the Musixmatch API to obtain the genres of each song.

After the data were extracted, there were 2421 complete data points, an 80.7% yield. Several sanitization methods took place afterwards, namely the filter_genres method [data_xtraction.py, lines 427 – 573, about 85 of those being the explanation to the algorithm], which guaranteed every song in the database had exactly one genre (described fully in data_xtraction.py).

Results

To begin, the data were extracted and cleaned as described in the Data section of this report. Genres were filtered down, and lyrics were cleaned of trailing punctuation and characters. Unfortunately, there were still 491 data points left genre-less by Musixmatch (probably due to the 2000 request limit). These data points were manually completed painstakingly using the hand_enter function inside of [data_xtraction.py, lines 574 – 600].

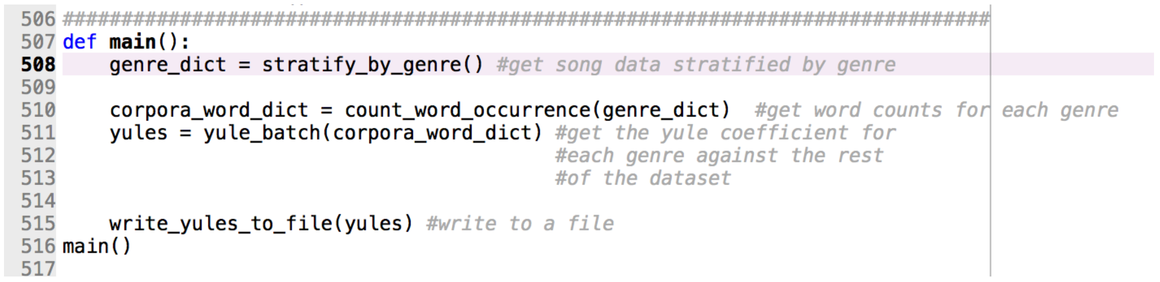
After cleaning through the data set, the analysis of the data began. It is important, however, to note, or clarify, the objective of this analysis. The analysis was meant to use the Yule coefficients across each genre in the data set (what words are common in one genre and simultaneously uncommon in the others?) and year, as well as the syllabic average of each song in each genre between 1980-2010, to make inferences, or spot changes in music over the course of that time period.

Yule Analysis

Hence, in data_analysis.py we see the code to find the Yule coefficients across each genre [data_analysis.py, lines 171 – 361]. The following code resulted in the following output:

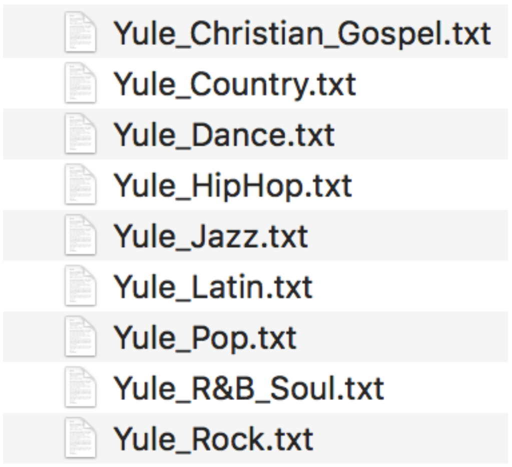
Figure (1)

IN: *Computing Yule code*



```
###########################################################################
def main():
    genre_dict = stratify_by_genre() #get song data stratified by genre

    corpora_word_dict = count_word_occurrence(genre_dict)  #get word counts for each genre
    yules = yule_batch(corpora_word_dict) #get the yule coefficient for
                                          #each genre against the rest
                                          #of the dataset

    write_yules_to_file(yules) #write to a file
main()

```

OUT: *The output files containing Yule coefficients*

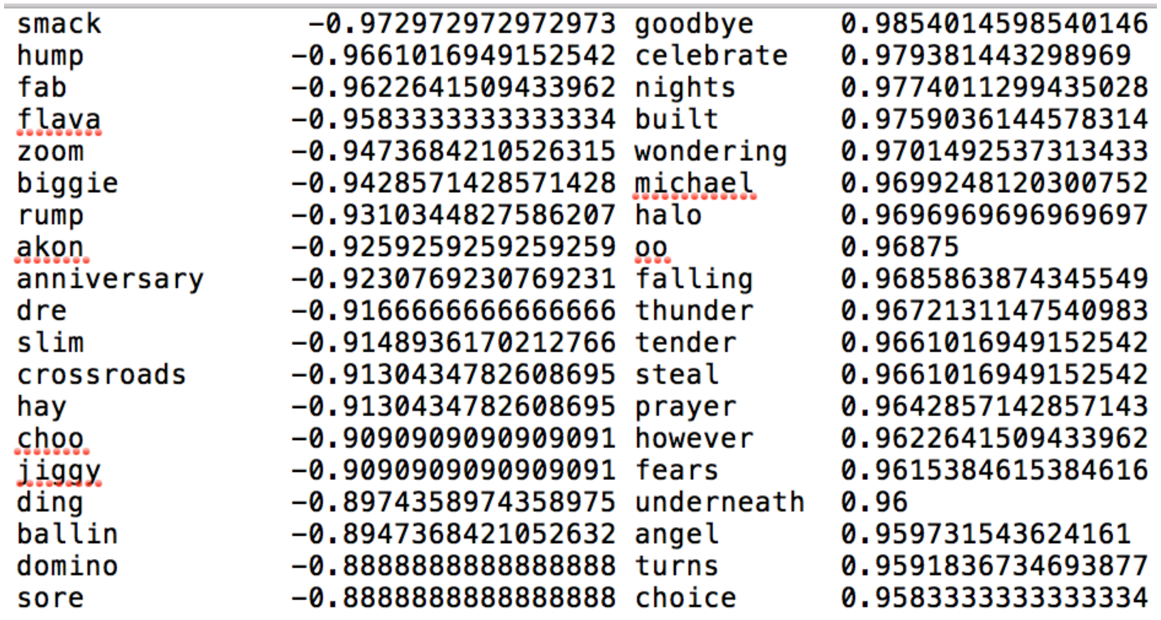


Yule_Christian_Gospel.txt
Yule_Country.txt
Yule_Dance.txt
Yule_HipHop.txt
Yule_Jazz.txt
Yule_Latin.txt
Yule_Pop.txt
Yule_R&B_Soul.txt
Yule_Rock.txt

Each of the files is a genre compared against the rest of the data set (not including that genre) and contains the top 100 Yule scoring words for each genre. Here is the inside of the Hip-Hop file:

Figure (2)

The first 19 entries of Yule_HipHop.txt (LEFT: Common in Hip-Hop)



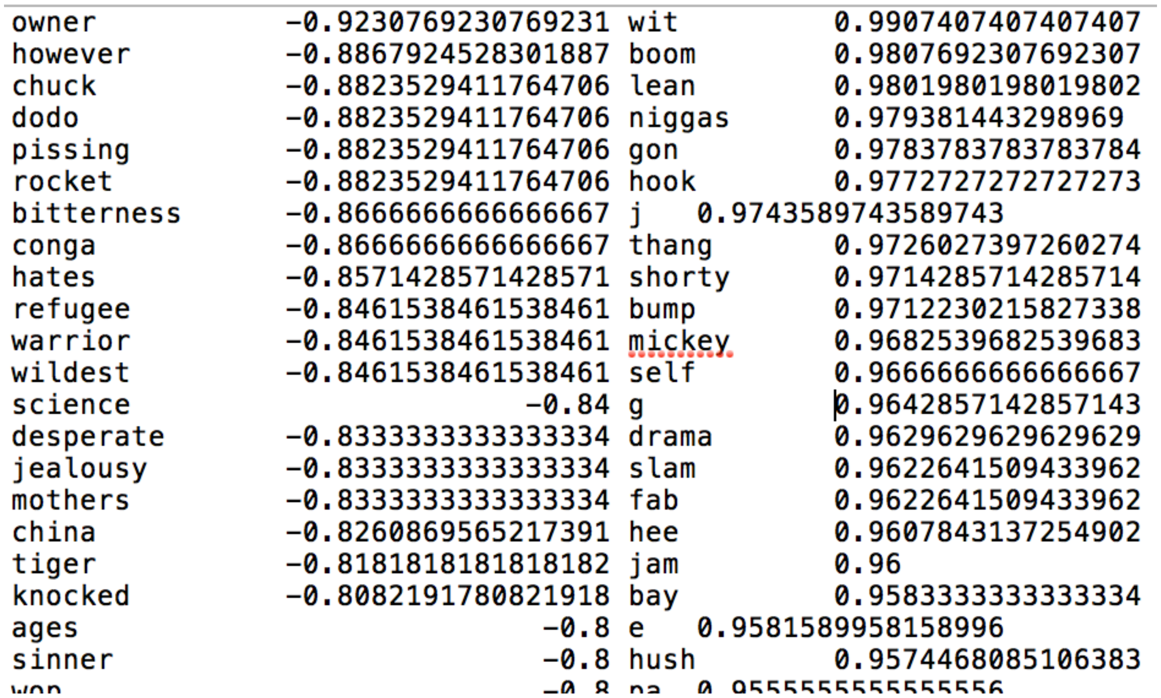
```
smack          -0.972972972972973  goodbye     0.9854014598540146
hump           -0.9661016949152542 celebrate   0.979381443298969
fab            -0.9622641509433962 nights      0.9774011299435028
flava          -0.9583333333333334 built       0.9759036144578314
zoom           -0.9473684210526315 wondering   0.9701492537313433
biggie         -0.9428571428571428 michael     0.9699248120300752
rump           -0.9310344827586207 halo        0.9696969696969697
akon           -0.9259259259259259 oo          0.96875
anniversary    -0.9230769230769231 falling     0.9685863874345549
dre            -0.9166666666666666 thunder     0.9672131147540983
slim           -0.9148936170212766 tender      0.9661016949152542
crossroads     -0.9130434782608695 steal       0.9661016949152542
hay            -0.9130434782608695 prayer      0.9642857142857143
choo           -0.9090909090909091 however     0.9622641509433962
jiggy          -0.9090909090909091 fears       0.9615384615384616
ding           -0.8974358974358975 underneath  0.96
ballin         -0.8947368421052632 angel       0.959731543624161
domino         -0.8888888888888888 turns       0.9591836734693877
sore           -0.8888888888888888 choice      0.9583333333333334
```

Already we can see notable information. According to this dataset, hip-hop songs commonly contain slang words, homages to other hip-hop artists (slim, dre, biggie), as well as suggestions of material satisfaction. The above snapshot and related calculations actually show that those words are truly more common in hip-hop and its subgenres than any other genre in the data set. Given even those top 19 words, a statistically backed statement on hip-hop's message can be formed (though the statement may still seem shaky at best).

Taking a look at another file (Yule_Rock.txt):

Figure (3)

The top 20 uncommon words in Rock and top 20 uncommon words in the rest.



```
owner              -0.9230769230769231 wit         0.9907407407407407
however            -0.8867924528301887 boom        0.9807692307692307
chuck              -0.8823529411764706 lean        0.9801980198019802
dodo               -0.8823529411764706 niggas      0.979381443298969
pissing            -0.8823529411764706 gon         0.9783783783783784
rocket             -0.8823529411764706 hook        0.9772727272727273
bitterness         -0.8666666666666667 j   0.9743589743589743
conga              -0.8666666666666667 thang       0.9726027397260274
hates              -0.8571428571428571 shorty      0.9714285714285714
refugee            -0.8461538461538461 bump        0.9712230215827338
warrior            -0.8461538461538461 mickey      0.9682539682539683
wildest            -0.8461538461538461 self        0.9666666666666667
science                          -0.84 g           0.9642857142857143
desperate          -0.8333333333333334 drama       0.9629629629629629
jealousy           -0.8333333333333334 slam        0.9622641509433962
mothers            -0.8333333333333334 fab         0.9622641509433962
china              -0.8260869565217391 hee         0.9607843137254902
tiger              -0.8181818181818182 jam         0.96
knocked            -0.8082191780821918 bay         0.9583333333333334
ages                              -0.8 e   0.9581589958158996
sinner                            -0.8 hush        0.9574468085106383
won                               -0.8 na  0.9555555555555556
```

The Rock Yule coefficients show us an underlying trend between the diction of rockers and the diction of musicians from the rest of the genre pool: Rock is most unlike hip-hop. This is shown from the fact that several of the most common words of hip-hop (see Yule_HipHop.txt) are of the uncommon words in the rock corpus. Hence, hip-hop and rock, by the information provided by this data set, truly are extremely dissimilar, fitting with the common notion that rock and hip-hop are very different.

The other 7 Yule text file contain important information as well, however Rock and Hip-Hop seem to be the most ‘uncorrupted’ corpora of the data set. That is, these corpora are large enough, dispersed enough throughout the years, and have the least errors in their genre placement (not many songs that should be in a different genre are hiding in Rock or Hip-Hop).

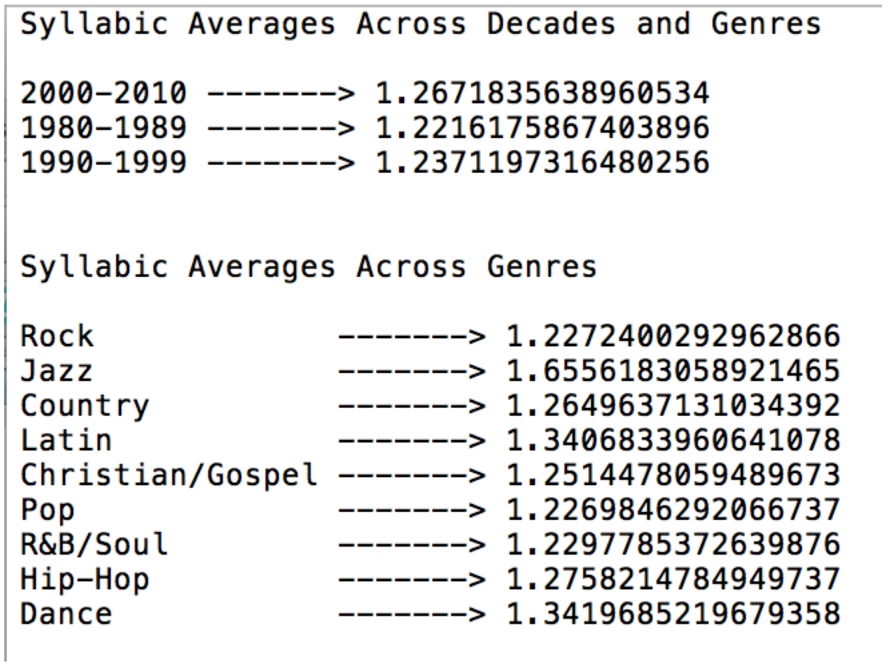
The slight corruption or tainting of the rest of the data set is generally due to misclassification of genre or specificity of genre. For example, given that Pop can be either simply a popular song or a song that has a ‘pop’ sound to it, many mistakes can be made there. In addition, since each of these songs comes from the top songs of that year, most songs (especially from 1980-1999) were simply listed under the ‘Pop’ genre, leading to an even more common case of misclassification.

Syllabic Average Analysis

The Yule coefficient analysis gave a nice insight into the narrative of some of the genres. The syllabic average of each song within each genre leads less to conclusions about the emotion or rhetoric of a song, and more towards the complexity. It is arguable that the more syllables that are used in a song imply that that song is more complex. Hence, analyzing the syllabic tendencies and the progression over the course of time could be a fruitful in gathering information about complexity differences among genres. All of this code is inside of data_analysis.py in the syllabic average section [data_analysis.py, lines 365 – 500, 643-697]. The syllabic average of each word was calculated using the nltk (Natural Language Tool Kit) and looking up words from the Carnegie Melon Pronunciation Dictionary (cmudict). If words were unsuccessfully found using cmudict, a simple and less accurate algorithm was used to calculate the syllables. See below for several figures detailing the output of the algorithm and plots associated with it.

Figure (4)

The output after running the algorithm both genre-wise and decade-wise:
(Taken from syllabic_tdncy.txt)

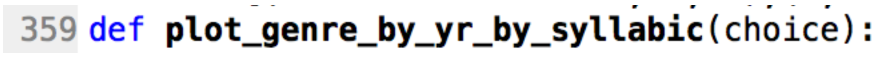


```
Syllabic Averages Across Decades and Genres

2000-2010 -------> 1.2671835638960534
1980-1989 -------> 1.2216175867403896
1990-1999 -------> 1.2371197316480256


Syllabic Averages Across Genres

Rock              -------> 1.2272400292962866
Jazz              -------> 1.6556183058921465
Country           -------> 1.2649637131034392
Latin             -------> 1.3406833960641078
Christian/Gospel  -------> 1.2514478059489673
Pop               -------> 1.226984629206673 7
R&B/Soul          -------> 1.2297785372639876
Hip-Hop           -------> 1.2758214784949737
Dance             -------> 1.3419685219679358
```

Figure (4) shows that there is marked difference between syllabic counts in the 2000-2010 decade versus the other two as well as important distinctions in average syllable count across genres. It is important to note that smaller corpora will have larger syllable counts if they potentially. In addition, since Spanish is commonly in Latin songs, those songs also have a naturally higher syllabic average. Hence, it suffices to apply the rule from before that pop may act as an amalgamate control, while rock, hip-hop, and even r&b/soul can be analyzed.

Figure (5)

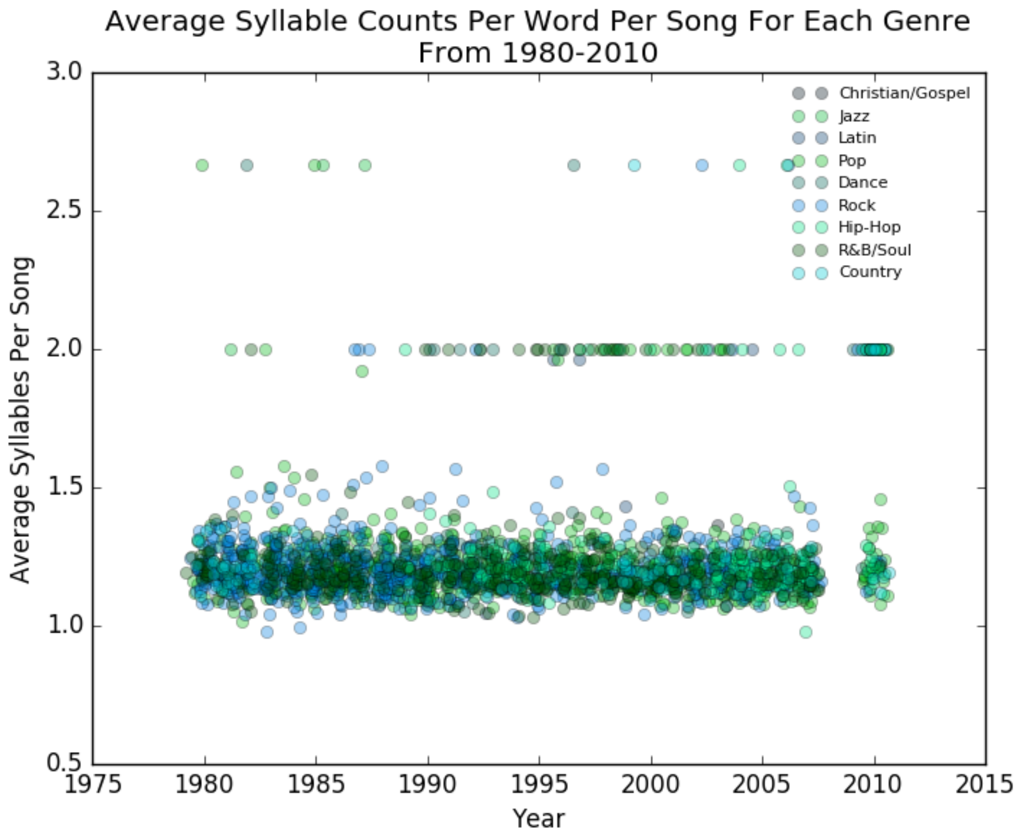
Inside of data_analysis.txt, a plotting function to graph syllabic trends over time



```
def plot_genre_by_yr_by_syllabic(choice):
```

Here is all of the data at once plotted as a time series:

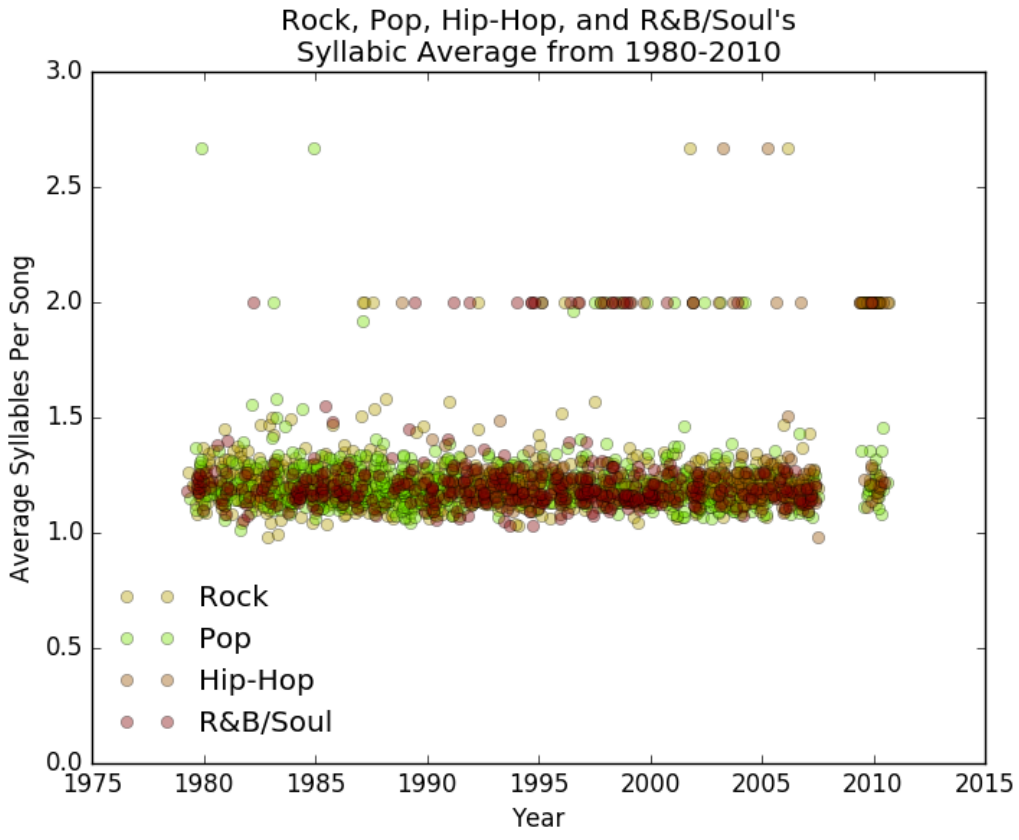
Figure (6)



As shown, the data conform to a general tendency of 1 – 1.5 syllables per song. This is expected, as songs must conform to the tempo, rhythm, et cetera in order to be melodic. We can note some strong outliers that hover at about 2 syllables per song and even some at about 2.7 syllables per song. Clearly, there are some interesting trends going on.

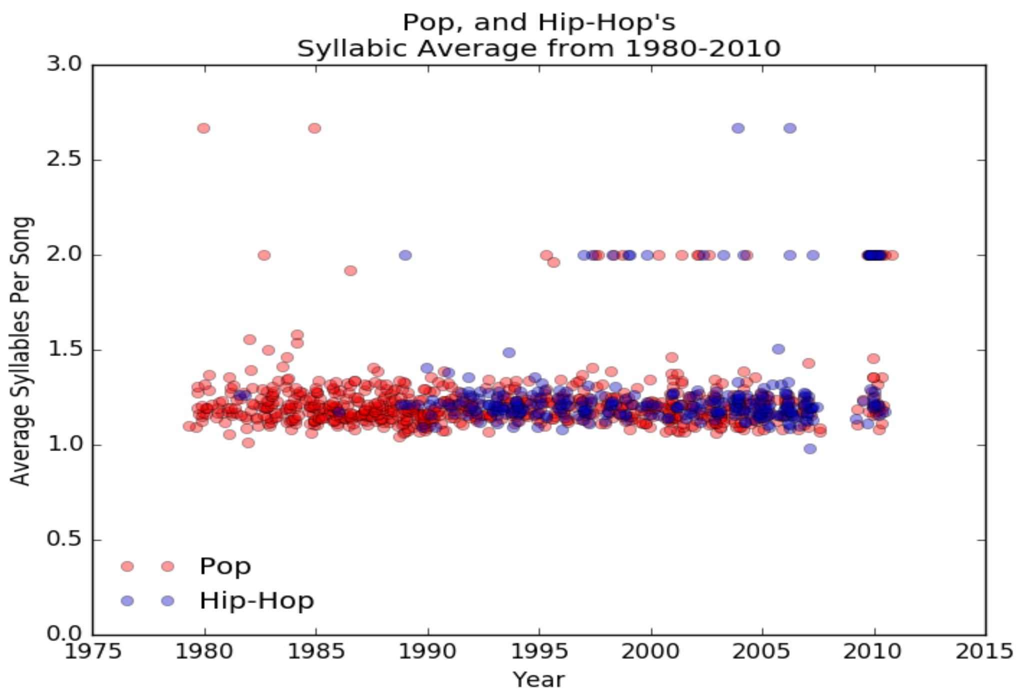
To get a better look, here is the time series, plotted in the same way for only hip-hop, rock, pop, and R&B/soul:

Figure (7)



This plot is also a bit crowded. Filtering it down to only hip-hop and pop:

Figure (8)



Using Figure (7), we find that hip-hop definitely has a strong presence in the same regions the range of syllabic tendency as pop (excluding where there is no data for hip-hop), even a possibly stronger presence at the 2.0 mark (this would require a lot more research and analysis.)

Overall, the syllabic average calculations tell us that the average syllable count for songs across most genres sits between 1.0 and 1.5 and debatably hip-hop has as higher syllabic tendency than pop. Mostly, however, we witnessed that there was truly an increase in syllabic tendency between 1980 and 2010. This may not speak to complexity of music but it surely does ensure the known statistic that hip-hop has become more prevalent given that in Figure (4) the 2000-2010 syllabic average nearly matches the hip-hop syllabic average. This means that hip-hop must be strongly influencing the syllabic average of 2010.

Conclusions

Overall, this project tested me most in my data analysis and data retrieval. It was challenging to get all the kinks of the database worked out, as well as analyzing the data once it was secured.

This project and these data shed light on very clear underlying trends in music. I cannot answer whether music has gotten ‘worse’ but it is easy to make inferences about the direction of music as far as genre and even word choice. An analysis I wish I had done is that of tracking the change in Yule coefficients across each year for each genre, this would give some more interesting information about the change in narrative of the lyrics for each genre.

Delving further into inference discussion, I attempted to build (and failed miserably) a machine-learning model based strictly off of Yule coefficients and syllabic tendency. The algorithm would run in a way that basically compared the entire data set against some input lyrics, calculated the Yule coefficients for the given song against the data set across each genre, using uncommon words of the lyrics as a model for guessing the actual genre of the song. In addition, the syllabic average would be calculated for the song and matched to the closest genre. Then, using the same type of technique, the song production year would be classified. I would have loved to analyze this model against a multinomial naïve bayes or some other mode and plot the success/failure data as a binomial distribution, but I did not have the time to do so.

Bibliography

Eveleth, R. (2012, July 27). Science Proves: Pop Music Has Actually Gotten Worse. Retrieved December 14, 12, from http://www.smithsonianmag.com/smart-news/science-proves-pop-music-has-actually-gotten-worse-8173368/

Additional Resources:

www.chartlyrics.com (API)

www.musixmatch.com (API)

http://stackoverflow.com/questions/5876040/number-of-syllables-for-words-in-a-text

http://www.nltk.org/genindex.html#S

http://stackoverflow.com/questions/14541303/count-the-number-of-syllables-in-a-word

http://matplotlib.org/users/legend_guide.html

http://stackoverflow.com/questions/18453176/removing-all-html-tags-along-with-their-content-from-text

http://stackoverflow.com/questions/1765848/remove-a-tag-using-beautifulsoup-but-keep-its-contents

https://www.crummy.com/software/BeautifulSoup/bs4/doc/#find

https://docs.python.org/2/library/urllib.html

https://docs.python.org/2/library/sqlite3.html

http://www.sqlitetutorial.net/sqlite-delete/

http://www.w3schools.com/sql/sql_select.asp

https://www.sqlite.org/lang_update.html

https://www.tutorialspoint.com/sql/sql-update-query.htm

http://stackoverflow.com/questions/305378/list-of-tables-db-schema-dump-etc-using-the-python-sqlite3-api