Big Data Analytics Final Report

Luyuan Shi
Statistics at GSAS
Columbia University
Ls3301@columbia.edu

Abstract— **In our final project, we build an item-based music recommendation system. We use the million song data set. First, we abstract the music information, and construct the feature matrix. Then conducting statistical and machine learning method to reduce the features. Next, we put the data into spark to train K-means clustering. After this, we use the music tag to check the accuracy of our clustering. Finally, we could find a method to define song similarity function and build the recommendation system.**

Keywords-component; million-song, item-based, feature selection, recommendation system, Spark, big data

I. INTRODUCTION

We all listen to music, music have become an essential part of our life. We are wondering what type of music is the main trend and how music develop for the past decades. Fortunately, Columbia EE laboratory have the million dataset. This dataset contain all the information we need. At first, I did data exploration. Then I use the data features to do clustering, the clustering results are quiet make sense. So this motivate me to make a music recommendation system.

Music recommendation has always been a hot topic. Most recommendation systems are user based. They use users' score and music review to give recommendation. However, more often, there are many songs that are "less popular" so that they don't have enough user review, in which case, these good songs would often fade away. What's more, people always keen to certain type of music, such as rock music that are noisy or folk music that are soft and slow. So we build recommendation system using the music data and recommend music have similar features, such as type, musical rhythm.

The first problem we would like to solve is that what features would best represent a song. Also, we want to find out how to put the giant data into Spark System to conduct very complicate clustering calculating work. What's more, we should find a way to judge whether our clustering is appropriate. At last, we need to define music similarity function thus we can recommend songs.

II. RELATED WORKS

At first, I write programs to load all the HDF5 files for the music and make some summary files. With these files, we are able to do Data Exploration.

For the recommendation system, the first step is to load the data and tried to parse the song into a same dimension. For instance, if we have second-record data such as pitches, different music have different length that the dimension of these song is different. In our data set, there are so many data related to a song, such as max loudness during each segment, chroma features for each segment and confidence of the key estimation. All the features are recorded as lists and have different length for different songs.

For these features, I find a way to process the features into a same dimension. I will explain this part in detail in ALGORITHM part.

More importantly, I set up Spark environment for python and thus could make the difficult machine learning calculating work faster.

III. SYSTEM OVERVIEW

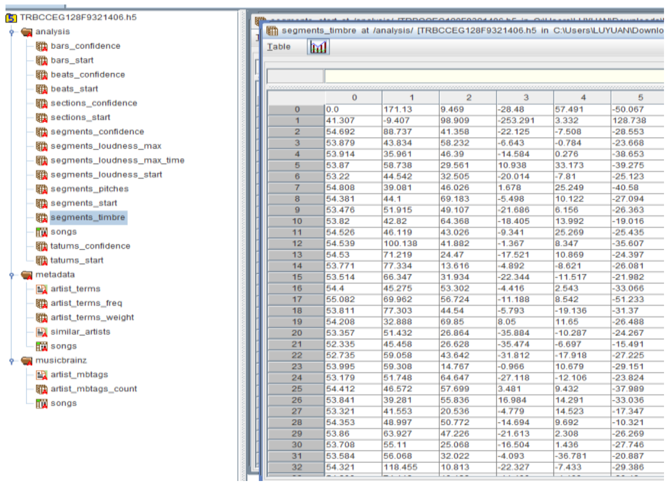
The ultimate goal of our problem is to build a recommendation system. This is input-output system. When you input a song name which is in our dataset. After similarity calculation, the system return you top three most similar songs related to your input.

The dataset we use is the million song data set that are constructed by Echo Nest and Columbia EE Laboratory.
This is a sample sub dataset for the entire dataset:
http://labrosa.ee.columbia.edu/millionsong/pages/getting-dataset#subset
For this subset, it is 2.8GB and contain exactly 10,000 songs.

However, for my project, I challenged myself, I used a larger dataset. Which is 8.95GB and contain 31000 songs.
You can download the dataset from the link below:

http://colinraffel.com/projects/lmd/#get

For each song, there is a HDF5 file that record various data for a song. HDF5 file is a tree-type file. We can have a look:



```
TRBCCEG128F9321406.h5
  analysis
    bars_confidence
    bars_start
    beats_confidence
    beats_start
    sections_confidence
    sections_start
    segments_confidence
    segments_loudness_max
    segments_loudness_max_time
    segments_loudness_start
    segments_pitches
    segments_start
    segments_timbre
    songs
    tatums_confidence
    tatums_start
  metadata
    artist_terms
    artist_terms_freq
    artist_terms_weight
    similar_artists
    songs
  musicbrainz
    artist_mbtags
    artist_mbtags_count
    songs
```

segments_timbre at /analysis/ [TRBCCEG128F9321406.h5 in C:\Users\LUYUAN\Downlo

	0	1	2	3	4	5
0	0.0	171.13	9.469	-28.48	57.491	-50.067
1	41.307	-9.407	98.909	-253.291	3.332	128.738
2	54.692	88.737	41.358	-22.125	-7.508	-28.553
3	53.879	43.834	58.232	-6.643	-0.784	-23.668
4	53.914	35.961	46.39	-14.584	0.276	-38.653
5	53.87	58.738	29.561	10.938	33.173	-39.275
6	53.22	44.542	32.505	-20.014	-7.81	-25.123
7	54.808	39.081	46.026	1.678	25.249	-40.58
8	54.381	44.1	69.183	-5.498	10.122	-27.094
9	53.476	51.915	49.107	-21.686	6.156	-26.363
10	53.82	42.82	64.368	-18.405	13.992	-19.016
11	54.526	46.119	43.026	-9.341	25.269	-25.435
12	54.539	100.138	41.882	-1.367	8.347	-35.607
13	54.53	71.219	24.47	-17.521	10.869	-24.397
14	53.771	77.334	13.616	-4.892	-8.621	-26.081
15	53.514	66.347	31.934	-22.344	-11.517	-21.982
16	54.4	45.275	53.302	-4.416	2.543	-33.066
17	55.082	69.962	56.724	-11.188	8.542	-51.233
18	53.811	77.303	44.54	-5.793	-19.136	-31.37
19	54.208	32.888	69.85	8.05	11.65	-26.488
20	53.357	51.432	26.864	-35.884	-10.287	-24.267
21	52.335	45.458	26.628	-35.474	-6.697	-15.491
22	52.735	59.058	43.642	-31.812	-17.918	-27.225
23	53.995	59.308	14.767	-0.966	10.679	-29.151
24	53.179	51.748	64.647	-27.118	-12.106	-23.824
25	54.412	46.572	57.699	3.481	9.432	-37.989
26	53.841	39.281	55.836	16.984	14.291	-33.036
27	53.321	41.553	20.536	-4.779	14.523	-17.347
28	54.353	48.997	50.772	-14.694	9.692	-10.321
29	53.86	63.927	47.226	-21.613	2.308	-26.269
30	53.708	55.11	25.068	-16.504	1.436	-27.746
31	53.584	56.068	32.022	-4.093	-36.781	-20.887
32	54.321	118.455	10.813	-22.327	-7.433	-29.386

Some of the feature description is designed as below:
bars_confidence: confidence value (between 0 and 1) associated with each bar by The Echo Nest.

bars_start: start time of each bar according to The Echo Nest.

beats_confidence: confidence value (between 0 and 1) associated with each beat by The Echo Nest.

segments_pitches: chroma features for each segment (normalized so max is 1.

segments_timbre: MFCC-like features for each segment.

For more detailed description:
http://labrosa.ee.columbia.edu/millionsong/pages/example-track-description

IV. Algorithm

This part, I will introduce in details how I deal with the data. And I will take a song as an example.

For the song file:

file_path	A/A/G/TRAAGMC128F4292D0F.h5
song_name	My Love
artist_name	LITTLE TEXAS

For the analysis data in the HDF5 file, here we summarize the data dimension:

Song Information	dimension
bars_confidence	(95,)
bars_start	(95,)
beats_confidence	(383,)
beats_start	(383,)
sections_confidence	(13,)
sections_start	(13,)
segments_confidence	(647,)
segments_loudness_max	(647,)
segments_loudness_max_time	(647,)
segments_loudness_start	(647,)
segments_pitches	(647,12)
segments_start	(647,)
tatums_confidence	(766,)
tatums_start	(766,)

We can see that for the features dimensions are of different length. Most of the features are of hundreds. However, the length of sections_confidence and sections_start only have lists of 13 numbers, so we will not use these two information.

Different songs have different duration time, so the dimensions of these features would be different. In order to make comparison. We calculate some statistics of these features. I calculated 7 statistics: mean, median, standard deviation, 25 percent quantile, 75 percent quantile, skewness, and Kurtosis.

The reason that I choose these statistics is that these statistics would best describe the distribution of certain number array. If the mean and median would not differ much, than it is more likely that the data is symmetrical distributed. In probability theory and statistics, skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The skewness value can be positive or negative. For a unimodal distribution, negative skew indicates that the tail on the left side of the probability density function is longer or fatter than the right side. Conversely, positive skew indicates that the tail on the right side is longer or fatter than the left side. For Kurtosis, it is a measure of the "tailedness" of the probability distribution of a real-valued random variable. The kurtosis of any univariate normal distribution is 3. Distributions with kurtosis less than 3 are said to be platykurtic and distributions with kurtosis greater than 3 are said to be leptokurtic. For differet quantile, it is also meaningful. Statistically, rock music is much loader than country music, thus bars and beats median of rock is much higher than country music.

So for a single feature, such as bars_confidence, we use the 7 statistics number to present it. However, the pitch information of this songs is 647 by 12. In music the pitch of a note means how high or low a note is and there are 12 notes needed in Western music. So for pitches data, it is always a N(segment number) by 12 matrix. So, for the pitch information, I would calculate the 7 statistics for each note. So we will have an 12*7=84 statistical numbers represent the pitches information.

At last, we have more than 150 features for a single song. The feature dimension is still high and might have collinearity problem. Then, I extract all the features for all the 30883 songs. I made a correlation matrix to check which features tend to have high correlation and I find that for most features, the mean, 25 percent quantile and 75 percent quantile are highly correlated. In order to solve the collinearity problem, we would choose only 5 statistics: mean, median, standard deviation, skewness, and Kurtosis.

After this, we have 115 features for a single song. If we want to do clustering or further analysis, there are two potential problems: the first is that the data have different scales, which would cause serious bias problem when we calculate the distance. The second problem is that the weight that represent the song should not be the same. For instance, should song loudness and song beats have the same weight?
If not, how should we assign the weights to each feature?

The solution for the above problem is to conduct PCA(Principle Component Analysis). To conduct PCA, the first step is to scale all the features so we don't have to worry different scale problem. Also, we could reduce the data dimension. For instance, the original feature dimension is n, after PCA, we could reduce the feature dimension to m. For each new feature, it is a linear combination of old n feature.

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \alpha_{21} & \cdots & \alpha_{n1} \\ \alpha_{12} & \alpha_{22} & \cdots & \alpha_{n2} \\ \vdots & \vdots & & \vdots \\ \alpha_{nn} & \alpha_{2n} & \cdots & \alpha_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Just as the formula above, α_{ij} is the weight assign for n-dimension feature x_j to m-dimension feature y_i.

After PCA, we will check how many principle component should I keep.

first n components	Variance Explained(%)	first n components	Variance Explained(%)
1	0.175	16	0.727
2	0.268	17	0.744
3	0.339	18	0.76
4	0.401	19	0.775
5	0.459	20	0.788
6	0.5	21	0.799
7	0.529	22	0.809
8	0.558	23	0.818
9	0.585	24	0.827
10	0.611	25	0.836
11	0.633	26	0.844
12	0.656	27	0.852
13	0.675	28	0.86
14	0.693	29	0.867
15	0.71	30	0.871

We see that the first 27 components will explain 85 percent of the total variance of the data. So I keep the first 27 component. Now, we successfully reduce our data dimension from 115 to 27 and we keep all the information in the 27 dimension new features.

Next, I would like to cluster to the songs into different groups based on the 27-dimensin features. I conducted hierarchical clustering to see how many clusters is the best. After hierarchical clustering, we can see that 4 clusters are the best.

Then I conduct K-means Clustering method and cluster the 30083 songs into 4 big groups. I will explain the result in details in Experiment Results part.

I define song distance as cosine distance between the two 27-dimension-array. The cosine distance is defined as below:

$$similarity = \cos(\theta) = \frac{A \bullet B}{\|A\|\|B\|} = \frac{\sum_{i=`}^{n} A_i B_i}{\sqrt{\sum_{i=`}^{n} A_i^2}\sqrt{\sum_{i=`}^{n} B_i^2}}$$

Where A_i and B_i are components of vector between A and B respectively. The reason why I choose cosine distance instead of euclidean distance is than when it comes to higher dimensions, cosine distance often works better and more appropriate.

When it comes to how to make recommendation, when a user input a song title, I will calculate the cosine distance for all the songs and return back the top 3 songs which have the smallest cosine distance.

Last but not least, I made word cloud plot for each cluster in part VI. I use terms' weight, instead simply add up, since weights are more accurate.

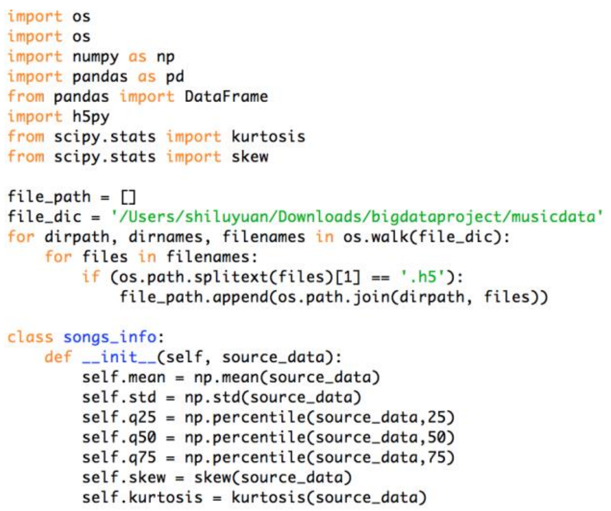
V. Software Package Description

I do main part of my project with python, for Spark, I also use python shell. For data visualization part, I also use Tableau.

Here I list all the packages I used in my project:

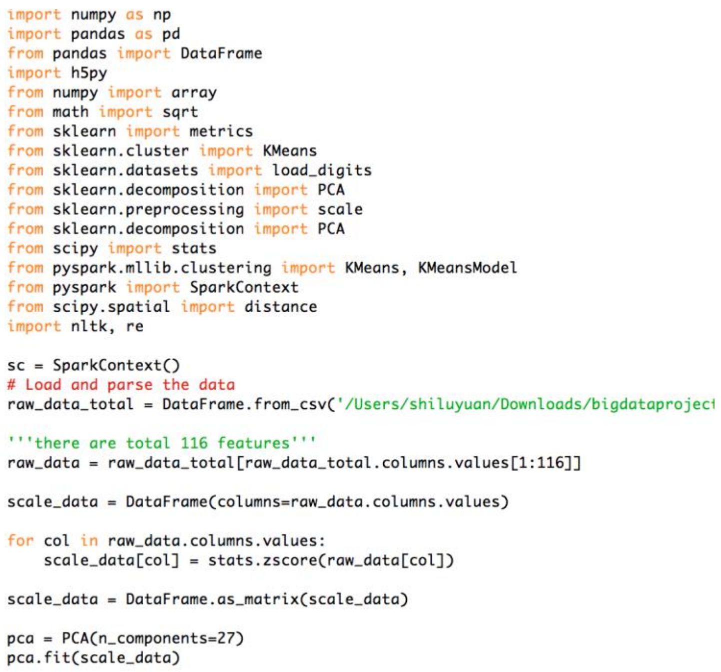
package	fucntion
os	use to operate files
numpy	get the data, easy to multipulate
pandas	easy to condunct the data
pandas.DataFrame	treat the data as dataframe
sklearn.decomposition.PCA	condunct PCA
h5py	read the HDF5 file
sklearn.preprocessing.scale	scale the data
scipy.stats.zscore	normalize the data
pyspark	parse the data
SparkContext	put the data into RDD
scipy.spatial.distance	calculate vector distance
pyspark.mllib.clustering	use spark to do clusting
scipy.stats.kurtosis	calculate data kurtosis
scipy.stats.skew	calculate data skewness
scipy.cluster.hierarchy	condunct hierachical clustering

I used some basic python package such as os, h5py, numpy and pandas to read and clean data. For the heavy clustering work, I use pyspark.mllib package.

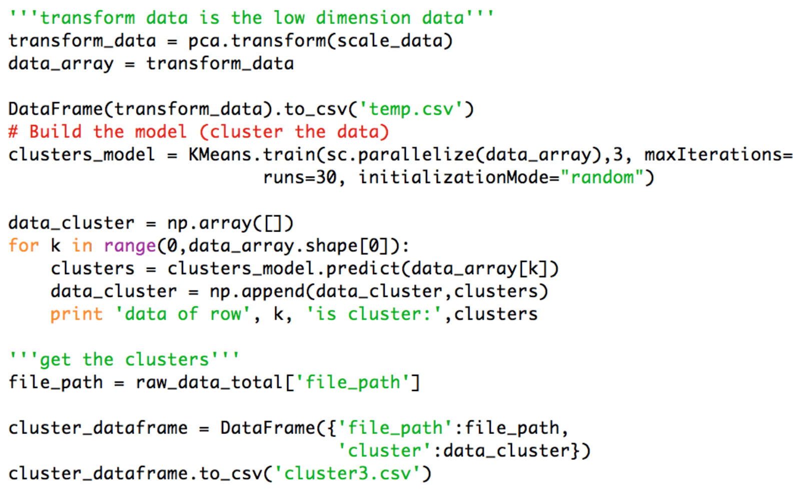


```
import os
import os
import numpy as np
import pandas as pd
from pandas import DataFrame
import h5py
from scipy.stats import kurtosis
from scipy.stats import skew

file_path = []
file_dic = '/Users/shiluyuan/Downloads/bigdataproject/musicdata'
for dirpath, dirnames, filenames in os.walk(file_dic):
    for files in filenames:
        if (os.path.splitext(files)[1] == '.h5'):
            file_path.append(os.path.join(dirpath, files))

class songs_info:
    def __init__(self, source_data):
        self.mean = np.mean(source_data)
        self.std = np.std(source_data)
        self.q25 = np.percentile(source_data,25)
        self.q50 = np.percentile(source_data,50)
        self.q75 = np.percentile(source_data,75)
        self.skew = skew(source_data)
        self.kurtosis = kurtosis(source_data)
```

In this simple case, I use package os, numpy, pandas and h5py to read the HDF5 file. For calculating skewness and kurtosis part, I use scipy.stats package.



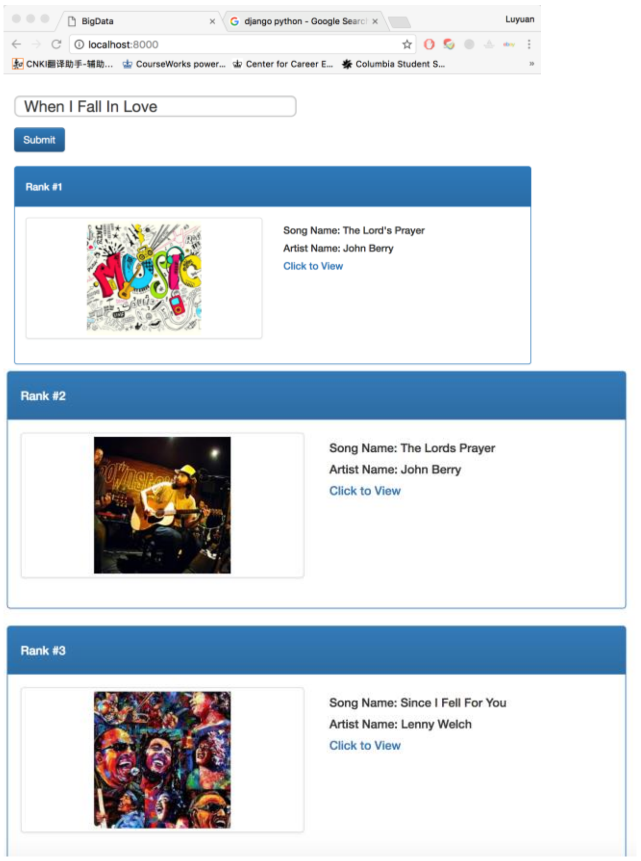
```
import numpy as np
import pandas as pd
from pandas import DataFrame
import h5py
from numpy import array
from math import sqrt
from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA
from scipy import stats
from pyspark.mllib.clustering import KMeans, KMeansModel
from pyspark import SparkContext
from scipy.spatial import distance
import nltk, re

sc = SparkContext()
# Load and parse the data
raw_data_total = DataFrame.from_csv('/Users/shiluyuan/Downloads/bigdataproject

'''there are total 116 features'''
raw_data = raw_data_total[raw_data_total.columns.values[1:116]]

scale_data = DataFrame(columns=raw_data.columns.values)

for col in raw_data.columns.values:
    scale_data[col] = stats.zscore(raw_data[col])

scale_data = DataFrame.as_matrix(scale_data)

pca = PCA(n_components=27)
pca.fit(scale_data)
```



```
'''transform data is the low dimension data'''
transform_data = pca.transform(scale_data)
data_array = transform_data

DataFrame(transform_data).to_csv('temp.csv')
# Build the model (cluster the data)
clusters_model = KMeans.train(sc.parallelize(data_array),3, maxIterations=
                        runs=30, initializationMode="random")

data_cluster = np.array([])
for k in range(0,data_array.shape[0]):
    clusters = clusters_model.predict(data_array[k])
    data_cluster = np.append(data_cluster,clusters)
    print 'data of row', k, 'is cluster:',clusters

'''get the clusters'''
file_path = raw_data_total['file_path']

cluster_dataframe = DataFrame({'file_path':file_path,
                               'cluster':data_cluster})
cluster_dataframe.to_csv('cluster3.csv')
```

In this example, I used the SparkContext to put the data into Spark RDD and then I use pyspark.mllib.clustering.Kmeans to do clustering calculation.

I also Django to bulid my recommendation system.
This is this the app UI:


BigData
django python - Google Search
Luyuan
localhost:8000
CNKI翻译助手-辅助... CourseWorks power... Center for Career E... Columbia Student S...

When I Fall In Love

Submit

Rank #1

Song Name: The Lord's Prayer
Artist Name: John Berry
Click to View

Rank #2

Song Name: The Lords Prayer
Artist Name: John Berry
Click to View

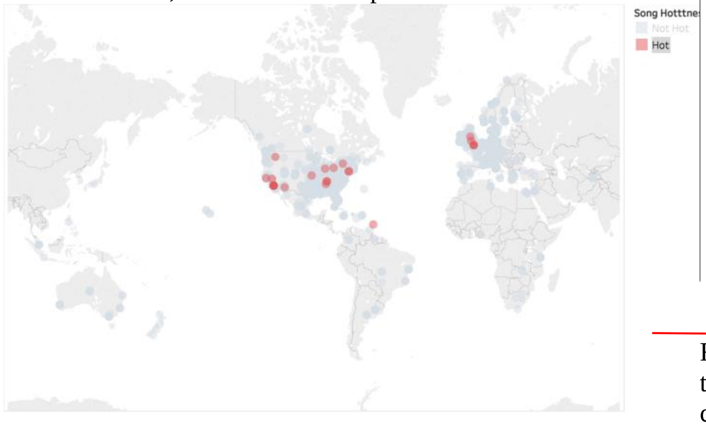
Rank #3

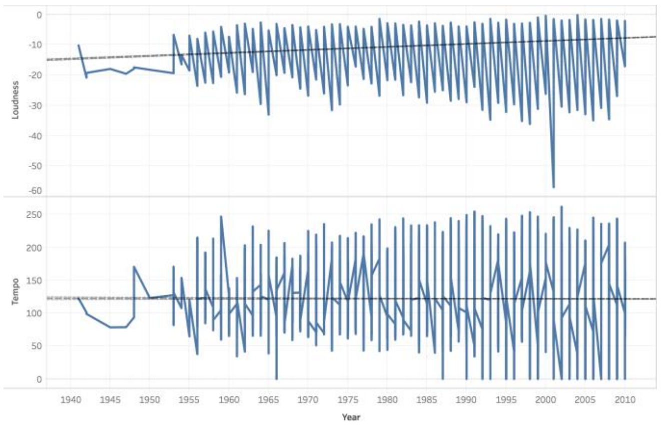
Song Name: Since I Fell For You
Artist Name: Lenny Welch
Click to View

As you can see, there is an input box, which the user input a song name. After calculation, the system return top 3 most similar songs and also pride the artist name and Youtube link. This recommendation system works is very conventient.

VI. Experiment Results

To begin with, let's explore the data. In the HDF5 file for each song, there are location data which is latitude and longitude. What's more, we also have song hotness data. With these data, I draw a word map.

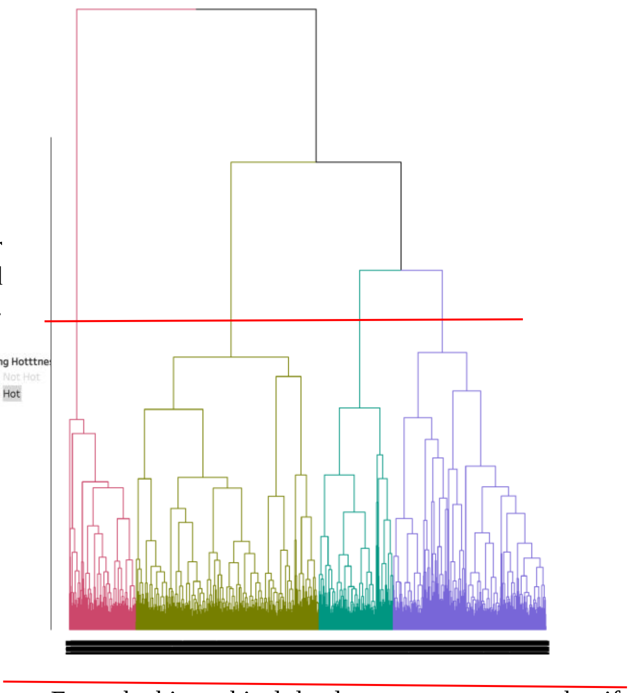




Also, I draw time trend plot for song loudness. We can see that in the past decades, songs become louder and louder. Maybe people are becoming more and more crazy.

From the map above we can see, most songs for this dataset are from USA and Europe. The red spots stand for hot songs and the blue spots represent not hot songs. We can see most hot songs in USA are from mid-east and almost all the hot songs in Europe are from The UK.

Next, I will illustrate why I cluster the data into 4 big groups. The dataset doesn't have a cluster or group tag for a single song, so I find out a way to check whether our cluster is meaning and make sense.



From the hierarchical dendrogram, we can see that if we cut the cluster at the red line, we could have 4 clusters and the clusters are quite clear.

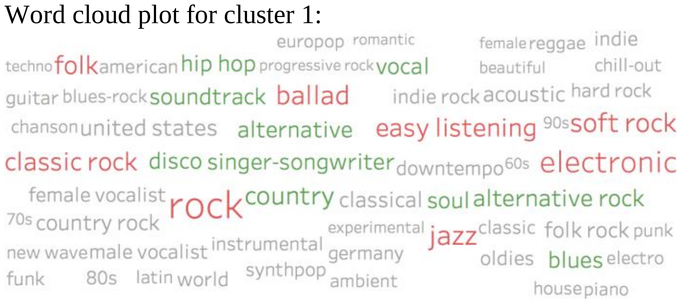
For each song, we have tags that symbol the songs. Take the song in part IV as an example:

terms	frequency	weight
honky tonk	1	1
folk-pop	0.87168881	0.89763935
country rock	0.82504395	0.86042827
classic country	0.78688528	0.82998708
blues-rock	0.76456389	0.81218012
ballad	0.76456389	0.80238024
pop rock	0.83353015	0.79982596
country	0.84191063	0.79744704
hard rock	0.86760907	0.79103079
soundtrack	0.85316557	0.7866795
anti-folk	0.72640523	0.78173893
folk	0.89179238	0.77711177
rockabilly	0.76456389	0.76919811
downtempo	0.81500691	0.76836437
world	0.82972724	0.76791766
oldies	0.78688528	0.75486426
easy listening	0.76456389	0.75441407
soft rock	0.72640523	0.75325093
techno	0.82504395	0.75046036
garage rock	0.72640523	0.7292917
folk rock	0.72640523	0.72567932
rock	0.87548696	0.72248191

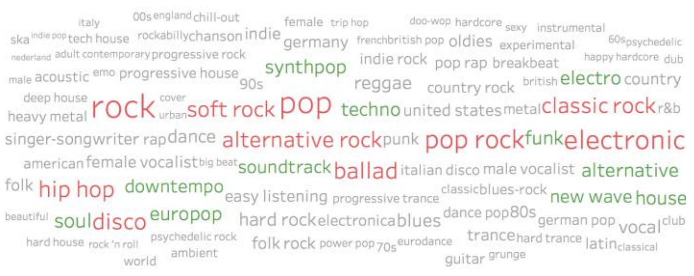
We can see that there are total 22 tags that describe this song. We can see word such as folk-pop, country rock and soft rock. Also, there are frequency and weight data that illustrate the weights for each term.

In order to see whether our cluster make sense, I make word could for each cluster:

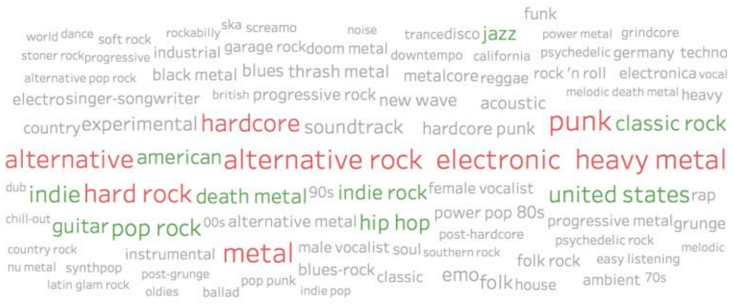
Word cloud plot for cluster 1:



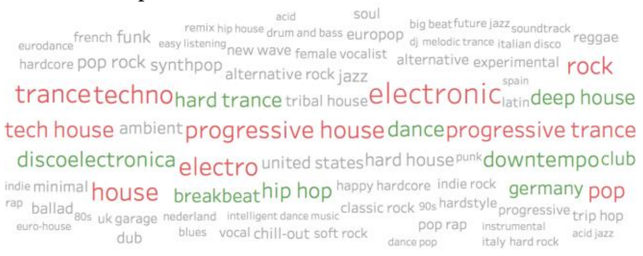
Word cloud plot for cluster 2:



Word cloud plot for cluster 3:



Word cloud plot for cluster 4:



For cluster 1, we can see that the top key words are folk, jazz, soft rock, classic rock. We call this type Jazz Fusion.

For cluster 2, the top words are pop, pop rock, hip hop, alternative rock and we call this group Pop Rock.

For cluster 3, punk, heavy metal, heavy, hard rock are top key words. Obviously, this cluster is Punk Rock.

For cluster 4, the top key words are trance, techno, house, tech house. This group is named as Trance music.

All these definitions are from Wikipedia. If you want to see more detailed definition, please click the keyword above.

As a conclusion, rock and pop are main trend. There are different type of rock and rock could be mixed with jazz and other type. Our clusters are quiet make sense and we could see very obvious difference between each group. All these suggest that the reduced dimension features could very well represent a song.

Here is the summary of clustering method:

	percentage	cluster name
cluster 1	30%	Jazz Fusion
cluster 2	14%	Pop Rock
cluster 3	40%	Punk Rock
cluster 4	16%	Trance music

For the recommendation part, for a given song, I have recommend 3 songs have the smallest cosine distance to the song using the 27 dimension data. I tried my web app recommendation system, it is very accurate most of the time. When I input a soft lyric song, it would recommend me similar type of songs, so the system does for rock type of music.
Here is an example.
Input song: Punk Rawk Show
Recommend song: Fame Infamy, Kingmaker and Dragostea Din Tei.
All the songs are punk rock, if you want to have a listen, please click on the song name.

VII. Conclusion

To begin with, we know that most of the songs for this data set is from USA and Europe. What's more, only a small part of songs are hot and most of the hot songs in USA are from mid-east and most hot songs are from England. Besides, the loudness of songs has a growing trend in the past few decades.
For the clustering result, judging from the word cloud, the clusters have significant difference to each other. This result suggest that the mean, median, standard deviation of all the features would very well represent a song and the PCA algorithm to reduce dimension would keep all the information and would reduce calculation work dramatically.

For the spark part, Spark RDD is real fast, for my project, it could run Hierarchical Clustering for 30883 songs with 27 features each within 10 seconds.

Team member contribution:
I did most part of this project. I gathered the data, did all the programming and wrote this final report on my own.
My teammate – Qianyun Zhang, made part of the presentation PPT and made presentaion.

Acknowledgment

I would like to thank Professor Lin. Professor introduced me to a whole new area which I had never heard of. He not only taught me much about big data inspire me the passion for big data analysis. What's more, I would also thank all my friends encourage me for this project.

Appendix

References

s