Emotion Identification of Songs

K. Prem Nishanth
S. Rahul
N. Ramanathan
Guide : Dr. R.S. Milton

SSN College of Engineering, Chennai

April 1, 2014

Overview

Introduction

- Music is an integral part of everyone's lives and every song portrays a different emotion
- To develop a system that classifies songs in real time by considering various features
- Selection of songs based on emotions can have wide applications in the music industry

Problem statement

To design a system that classifies emotion of songs as happy or sad, by analyzing audio-related features and lyrics

Input

Dataset containing audio-related features like tempo, energy, mode, key, loudness, harmony and lyrics

Output

Predicted emotion of song

Objective

To identify emotions of songs
To increase accuracy of prediction

Existing works and novel approach

- Most systems use *audio-related features* while some use *lyrics* for classification
- Very few have incorporated both

Novel approach

1. Extra audio feature *loudness* included
2. New method to calculate *harmony*
3. New approach to combine audio and lyrics into single feature space and perform classification

Design

Used 3 approaches to identify song's emotion

1. Audio-related features of the song
2. Lyrics of the song
3. Combination of both

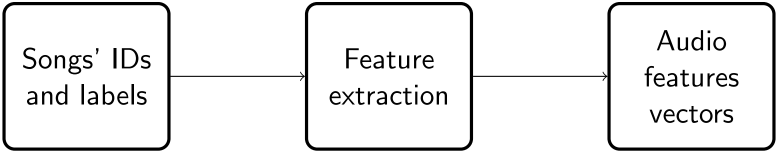
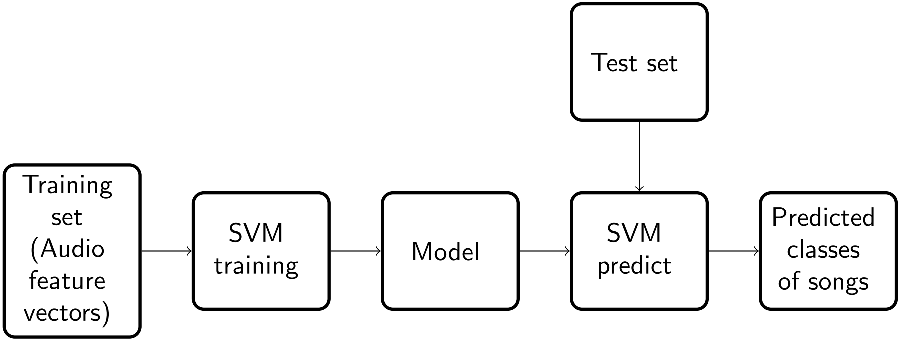


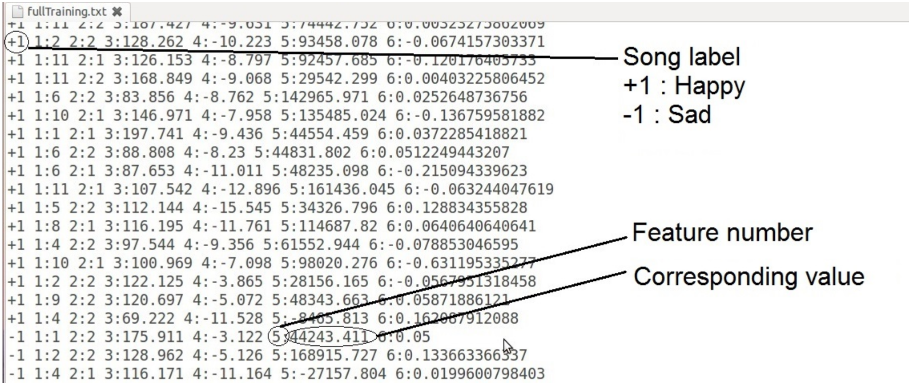
Figure: Audio Features Module 1

Audio features module

LIBSVM is an existing implementation of support vector machine


Test set
Training set (Audio feature vectors)
SVM training
Model
SVM predict
Predicted classes of songs

Audio features module



```
+1 1:11 2:2 3:187.427 4:-9.631 5:74442.752 6:0.00323275862069
+1 1:2 2:2 3:128.262 4:-10.223 5:93458.078 6:-0.0674157303371
+1 1:11 2:1 3:126.153 4:-8.797 5:92457.685 6:-0.120176405733
+1 1:11 2:2 3:168.849 4:-9.068 5:29542.299 6:0.00403225806452
+1 1:6 2:2 3:83.856 4:-8.762 5:142965.971 6:0.0252648736756
+1 1:10 2:1 3:146.971 4:-7.958 5:135485.024 6:-0.136759581882
+1 1:1 2:1 3:197.741 4:-9.436 5:44554.459 6:0.0372285418821
+1 1:6 2:2 3:88.808 4:-8.23 5:44831.802 6:0.0512249443207
+1 1:6 2:1 3:87.653 4:-11.011 5:48235.098 6:-0.215094339623
+1 1:11 2:1 3:107.542 4:-12.896 5:161436.045 6:-0.063244047619
+1 1:5 2:2 3:112.144 4:-15.545 5:34326.796 6:0.128834355828
+1 1:8 2:1 3:116.195 4:-11.761 5:114687.82 6:0.0640640640641
+1 1:4 2:2 3:97.544 4:-9.356 5:61552.944 6:-0.078853046595
+1 1:10 2:1 3:100.969 4:-7.098 5:98020.276 6:-0.631195335277
+1 1:2 2:2 3:122.125 4:-3.865 5:28156.165 6:-0.0567951318458
+1 1:9 2:2 3:120.697 4:-5.072 5:48343.663 6:0.05871886121
+1 1:4 2:2 3:69.222 4:-11.528 5:-8465.813 6:0.162087912088
-1 1:1 2:2 3:175.911 4:-3.122 5:44243.411 6:0.05
-1 1:2 2:2 3:128.962 4:-5.126 5:168915.727 6:0.133663366337
-1 1:4 2:1 3:116.171 4:-11.164 5:-27157.804 6:0.0199600798403
```

Figure: LIBSVM input format

- 10-fold Cross-validation performed on dataset

Audio features module

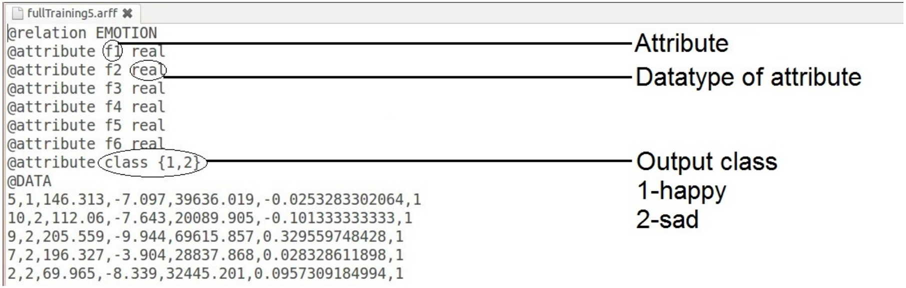
LIBLINEAR

- Input format same as LIBSVM
- Used for larger datasets
- Faster than LIBSVM

WEKA toolkit

- Input file in `arff` format
- Experimented with different algorithms
- Random Forests produced best results

Audio features module



```
@relation EMOTION
@attribute f1 real
@attribute f2 real
@attribute f3 real
@attribute f4 real
@attribute f5 real
@attribute f6 real
@attribute class {1,2}
@DATA
5,1,146.313,-7.097,39636.019,-0.0253283302064,1
10,2,112.06,-7.643,20089.905,-0.101333333333,1
9,2,205.559,-9.944,69615.857,0.329559748428,1
7,2,196.327,-3.904,28837.868,0.028328611898,1
2,2,69.965,-8.339,32445.201,0.0957309184994,1
```

Figure: WEKA input format

Audio features module

Random forests

- Ensemble learning method for classification
- Constructs multitude of decision trees
- Output is class that is most frequently occurring
- Implemented it in Java

Lyrics module

Naive Bayes classifier used for lyrics classification

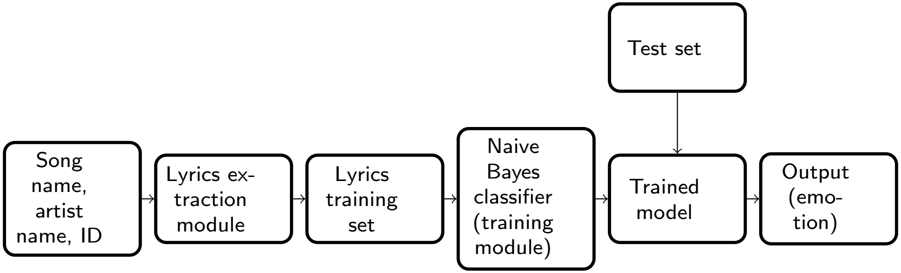


Figure: Lyrics Module

- Python program to extract lyrics from lyrics websites, `www.azlyrics.com` and `www.metrolyrics.com`
- Lyrics kept in 2 folders, `Happy` and `Sad`

Lyrics module

Basic Model

- Probabilistic classifier based on applying Bayes theorem with strong independent assumptions
- Presence or absence of a feature unrelated to the presence or absence of other features
- Example: Lyrics of a song contains title, introduction, verse and chorus and each contributes independently to classification

Significance of chorus and title

- Title and chorus depict the main theme
- Add more weight to words from title and chorus

Lyrics module

NLTK

- Provides interfaces along with text processing libraries for classification
- Implementation of Naive Bayes in NLTK used to classify based on lyrics

Combination of audio features and lyrics

Using consensus from multi-layer graphs

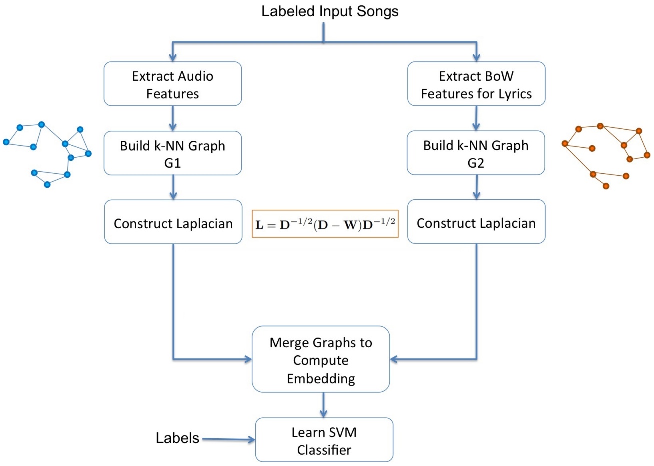


Figure: Training

Combination of audio features and lyrics

Using consensus from multi-layer graphs

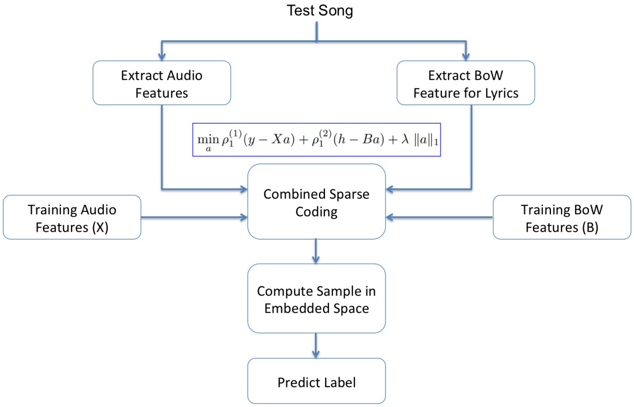

Test Song
Extract Audio Features
Extract BoW Feature for Lyrics
$\min_{a} \rho_1^{(1)}(y - Xa) + \rho_1^{(2)}(h - Ba) + \lambda \|a\|_1$
Training Audio Features (X)
Combined Sparse Coding
Training BoW Features (B)
Compute Sample in Embedded Space
Predict Label

Figure: Testing

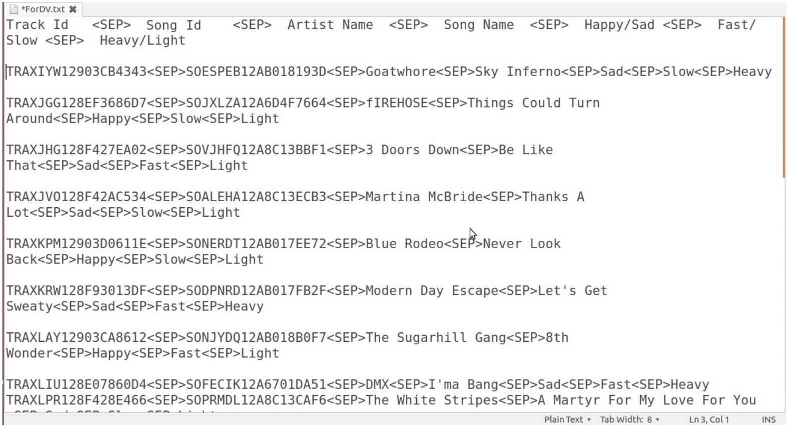
Implementation

Audio features extraction from songs

- Tried to create dataset by extracting features from songs
- Audio feature extraction tools used:
  1. **YAAFE** (Yet Another Audio Feature Extraction): MFCC feature of one song was extracted but required features were not extractable
  2. **MusicBrainz Server**: Local musicbrainz server was set up but connection to its database failed
- We attempted to use above mentioned tools but for now we used the available Million Song Dataset

Implementation

Manual labelling of song



*ForDV.txt

```
Track Id   <SEP>  Song Id    <SEP>  Artist Name  <SEP>  Song Name  <SEP>  Happy/Sad <SEP>  Fast/
Slow <SEP>  Heavy/Light

TRAXIYW12903CB4343<SEP>SOESPEB12AB018193D<SEP>Goatwhore<SEP>Sky Inferno<SEP>Sad<SEP>Slow<SEP>Heavy

TRAXJGG128EF3686D7<SEP>SOJXLZA12A6D4F7664<SEP>fIREHOSE<SEP>Things Could Turn
Around<SEP>Happy<SEP>Slow<SEP>Light

TRAXJHG128F427EA02<SEP>SOVJHFQ12A8C13BBF1<SEP>3 Doors Down<SEP>Be Like
That<SEP>Sad<SEP>Fast<SEP>Light

TRAXJVO128F42AC534<SEP>SOALEHA12A8C13ECB3<SEP>Martina McBride<SEP>Thanks A
Lot<SEP>Sad<SEP>Slow<SEP>Light

TRAXKPM12903D0611E<SEP>SONERDT12AB017EE72<SEP>Blue Rodeo<SEP>Never Look
Back<SEP>Happy<SEP>Slow<SEP>Light

TRAXKRW128F93013DF<SEP>SODPNRD12AB017FB2F<SEP>Modern Day Escape<SEP>Let's Get
Sweaty<SEP>Sad<SEP>Fast<SEP>Heavy

TRAXLAY12903CA8612<SEP>SONJYDQ12AB018B0F7<SEP>The Sugarhill Gang<SEP>8th
Wonder<SEP>Happy<SEP>Fast<SEP>Light

TRAXLIU128E07860D4<SEP>SOFECIK12A6701DA51<SEP>DMX<SEP>I'ma Bang<SEP>Sad<SEP>Fast<SEP>Heavy
TRAXLPR128F428E466<SEP>SOPRMDL12A8C13CAF6<SEP>The White Stripes<SEP>A Martyr For My Love For You
```

Figure: Classified songs

Implementation

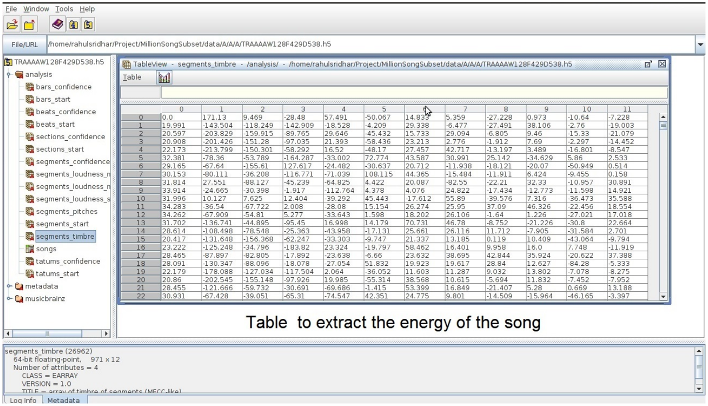
Audio features used

- **Key:** Identifies which of the 12 keys the song has been played in
- **Mode:** Song can be played in either minor mode or major mode
- **Tempo:** Speed of the song (measured in beats per minute)
- **Energy:** Work done to produce a tone at a particular frequency
- **Loudness:** Refers to general loudness of song. Perception of amplitude
- **Harmony:** Combination of simultaneously sounded musical notes to produce a pleasing effect

Implementation

Analyzing the features and their combinations

- Features automatically retreived from dataset using a Python script
- Input is song name and output is stored as training set



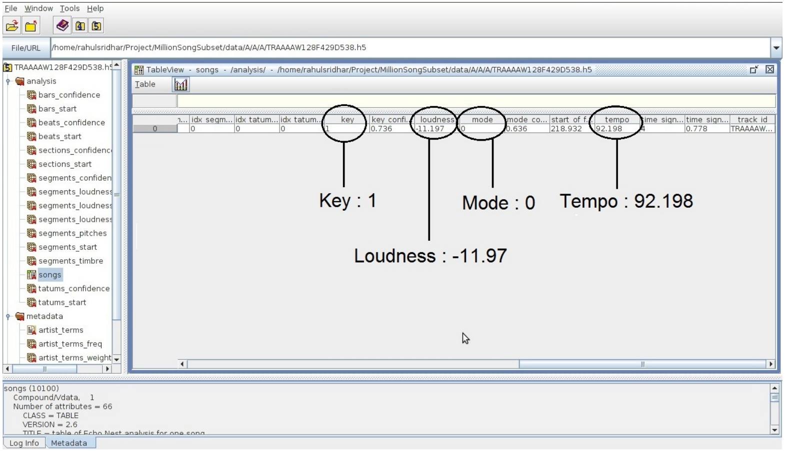
	0	1	2	3	4	5	6	7	8	9	10	11
0	0.0	171.13	9.469	-28.48	57.491	-50.067	14.833	5.359	-27.228	0.973	-10.64	-7.228
1	19.991	-143.504	-118.249	-142.909	-18.528	-4.209	29.338	-6.477	-27.491	38.106	-2.76	-19.003
2	20.597	-203.829	-159.915	-89.765	29.646	-45.432	15.733	29.094	-6.805	9.46	-15.33	-21.079
3	20.908	-201.426	-151.28	-97.035	21.393	-58.436	23.213	2.776	-1.912	7.69	-2.297	-14.452
4	22.173	-213.799	-150.301	-58.292	16.52	-48.17	27.457	42.717	-13.197	3.489	-16.801	-8.547
5	32.381	-78.36	-53.789	-164.287	-33.002	72.774	43.587	30.991	25.142	-34.629	5.86	2.533
6	29.165	-67.64	-155.61	127.617	-24.482	-30.637	20.712	-11.938	-18.121	-20.07	-50.949	0.514
7	30.153	-80.111	-36.208	-116.771	-71.039	108.115	44.365	-15.484	-11.911	6.424	-9.455	0.158
8	31.814	27.551	-88.127	-45.239	-64.825	4.422	20.087	-82.55	-22.21	32.33	-10.957	30.891
9	33.914	-24.665	-30.398	-1.917	-112.764	4.378	4.076	24.822	-17.434	-12.773	-11.598	14.921
10	31.996	10.127	7.625	12.404	-39.292	45.443	-17.612	55.89	-39.576	7.316	-36.473	35.588
11	34.283	-36.54	-67.722	2.008	-28.08	15.154	26.274	25.95	37.09	46.326	-22.456	18.554
12	34.262	-67.909	-54.81	5.277	-33.643	1.598	18.202	26.106	-1.64	1.226	-27.021	17.018
13	31.702	-136.741	-44.895	-95.45	16.998	14.179	70.731	46.78	-8.752	-21.226	-30.8	22.664
14	28.614	-108.498	-78.548	-25.363	-43.958	-17.131	25.661	26.116	11.712	-7.905	-31.584	2.701
15	20.417	-131.648	-156.368	-62.247	-33.303	-9.747	21.337	13.185	0.119	10.409	-43.064	-9.794
16	23.222	-125.248	-34.796	-183.82	23.324	-19.797	58.462	16.401	9.958	16.0	7.748	-11.919
17	28.465	-87.897	-82.805	-17.892	-23.638	-6.66	23.632	38.695	42.844	35.924	-20.622	37.388
18	28.091	-130.347	-88.096	-18.078	-27.054	51.832	19.923	19.617	28.84	12.627	-84.28	-5.333
19	22.179	-178.088	-127.034	-117.504	2.064	-36.052	11.603	11.287	9.032	13.802	-7.078	-8.275
20	20.86	-202.545	-155.148	-97.926	19.985	-55.314	38.568	10.615	-5.694	11.832	-7.452	-7.952
21	28.455	-121.666	-59.732	-30.691	-69.686	-1.415	53.399	16.849	-21.407	5.28	0.669	13.188
22	30.931	-67.428	-39.051	-65.31	-74.547	42.351	24.775	9.801	-14.509	-15.964	-46.165	-3.397

Table to extract the energy of the song

```
segments_timbre (26962)
  64-bit floating-point,   971 x 12
  Number of attributes = 4
     CLASS = EARRAY
     VERSION = 1.0
     TITLE = array of timbre of segments (MFCC-like)
```

Figure: HDF dataset, SegmentsTimbre table

Implementation



	ı...	idx segm...	idx tatum...	idx tatum...	key	key confi...	loudness	mode	mode co...	start of f...	tempo	ime sign...	time sign...	track id
0		0	0	0	1	0.736	-11.197	0	0.636	218.932	92.198	4	0.778	TRAAAAW...

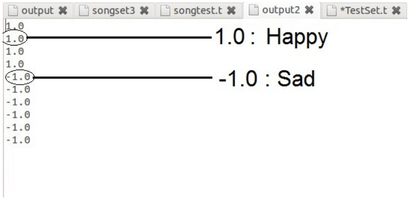
```
songs (10100)
  Compound/Vdata,   1
  Number of attributes = 66
    CLASS = TABLE
    VERSION = 2.6
    TITLE = table of Echo Nest analysis for one song
```

Figure: HDF dataset, Songs table

Audio features

LIBSVM

- Training set file given to SVMtrain
- Resultant model file, along with test set, given to SVMpredict
- Output file contains classified results
- 10-fold cross validation used for initial 110 songs set



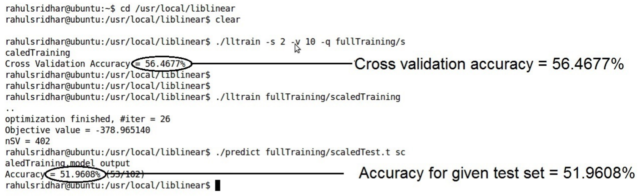
```
1.0
1.0
1.0
1.0
-1.0
-1.0
-1.0
-1.0
-1.0
-1.0
```

Figure: LIBSVM output

Audio features

LIBLINEAR

- When dataset was increased, results from LIBSVM were biased towards 'sad'.
- Training and test set format same as LIBSVM
- Accuracy obtained from cross validation was 56%



```
rahulsridhar@ubuntu:~$ cd /usr/local/liblinear
rahulsridhar@ubuntu:/usr/local/liblinear$ clear

rahulsridhar@ubuntu:/usr/local/liblinear$ ./lltrain -s 2 -v 10 -q fullTraining/s
caledTraining
Cross Validation Accuracy = 56.4677%
rahulsridhar@ubuntu:/usr/local/liblinear$
rahulsridhar@ubuntu:/usr/local/liblinear$
rahulsridhar@ubuntu:/usr/local/liblinear$ ./lltrain fullTraining/scaledTraining
..
optimization finished, #iter = 26
Objective value = -378.965140
nSV = 402
rahulsridhar@ubuntu:/usr/local/liblinear$ ./predict fullTraining/scaledTest.t sc
aledTraining.model output
Accuracy = 51.9608% (53/102)
rahulsridhar@ubuntu:/usr/local/liblinear$
```

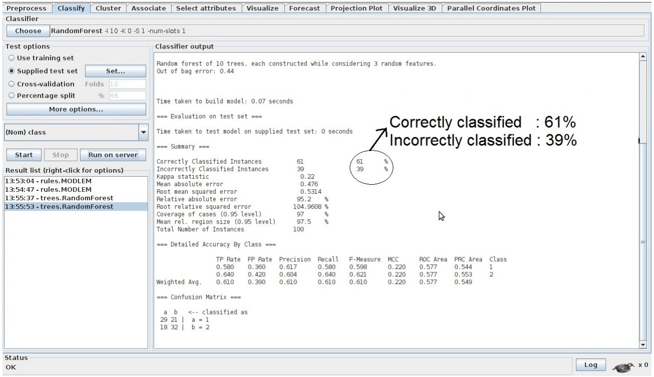
Cross validation accuracy = 56.4677%
Accuracy for given test set = 51.9608%

Figure: LIBLINEAR output

Audio features

WEKA Toolkit

- Input file in .arff format
- Cross validation sets created using a Python script
- Results noted after trying various classifiers



RandomForest -I 10 -K 0 -S 1 -num-slots 1

Cross-validation Folds 10
Percentage split % 66

Status: OK

```
13:53:04 - rules.MODLEM
13:54:47 - rules.MODLEM
13:55:37 - trees.RandomForest
13:55:53 - trees.RandomForest
```

```
Random forest of 10 trees, each constructed while considering 3 random features.
Out of bag error: 0.44



Time taken to build model: 0.07 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances          61               61      %
Incorrectly Classified Instances        39               39      %
Kappa statistic                          0.22
Mean absolute error                      0.476
Root mean squared error                  0.5314
Relative absolute error                 95.2    %
Root relative squared error            104.9608 %
Coverage of cases (0.95 level)          97      %
Mean rel. region size (0.95 level)      97.5    %
Total Number of Instances              100

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.580    0.360    0.617      0.580    0.598      0.220    0.577     0.544     1
                 0.640    0.420    0.604      0.640    0.621      0.220    0.577     0.553     2
Weighted Avg.    0.610    0.390    0.610      0.610    0.610      0.220    0.577     0.549

=== Confusion Matrix ===

  a  b   <-- classified as
 29 21 |  a = 1
 18 32 |  b = 2
```

Correctly classified : 61%
Incorrectly classified : 39%

Figure: Result of Random forest

Audio features

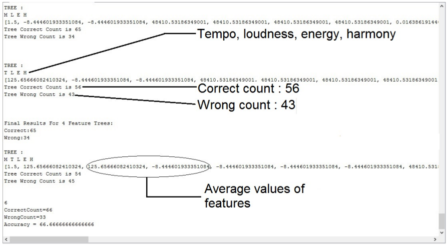
Random forests

- Coding done in Java with NetBeans as frontend
- Several decision trees created using all audio features
- Average value calculated based on following formula:

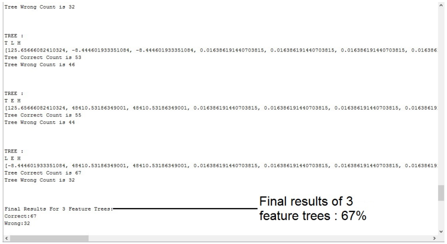
$$\text{avg} = \text{avgT} - \frac{|\text{avgH} - \text{avgS}| * |\text{countH} - \text{countS}|}{\text{countT}}$$

Insert decision tree picture (tikz image showing error)

Audio features



```
TREE :
M L E H
[1.5, -8.444601933351084, -8.444601933351084, 48410.53186349001, 48410.53186349001, 48410.53186349001, 48410.53186349001, 0.01638619144
Tree Correct Count is 65
Tree Wrong Count is 34


TREE :
T L E H
[125.65666082410324, -8.444601933351084, -8.444601933351084, 48410.53186349001. 48410.53186349001, 48410.53186349001, 48410.53186349001
Tree Correct Count is 56
Tree Wrong Count is 43


Final Results For 4 Feature Trees:
Correct:65
Wrong:34

TREE :
M T L E H
[1.5, 125.65666082410324, 125.65666082410324, -8.444601933351084, -8.444601933351084, -8.444601933351084, -8.444601933351084, 48410.531


6
CorrectCount=66
WrongCount=33
Accuracy = 66.66666666666666
```

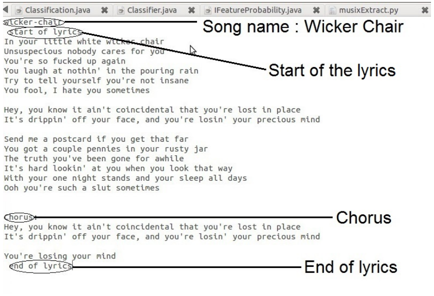


```
Tree Wrong Count is 32



TREE :
T L H
[125.65666082410324, -8.444601933351084, -8.444601933351084, 0.016386191440703815, 0.016386191440703815, 0.016386191440703815, 0.016386
Tree Correct Count is 53
Tree Wrong Count is 46



TREE :
T E H
[125.65666082410324, 48410.53186349001, 48410.53186349001, 0.016386191440703815, 0.016386191440703815, 0.016386191440703815, 0.01638619
Tree Correct Count is 55
Tree Wrong Count is 44



TREE :
L E H
[-8.444601933351084, 48410.53186349001, 48410.53186349001, 0.016386191440703815, 0.016386191440703815, 0.016386191440703815, 0.01638619
Tree Correct Count is 67
Tree Wrong Count is 32



Final Results For 3 Feature Trees:
Correct:67
Wrong:32
```

Figure: Output of Random forests with prediction accuracy

Lyrics

- Extraction of lyrics automated from 2 websites using Python
- Takes song name and artist name as input and generates dataset
  1. **AZlyrics.com:** "www.azlyrics.com/ArtistName/SongName.html"
  2. **Metrolyrics.com:** "www.metrolyrics.com/SongName-lyrics-ArtistName.html"



```
wicker-chair
 start of lyrics
In your little white wicker chair
Unsuspecious nobody cares for you
You're so fucked up again
You laugh at nothin' in the pouring rain
Try to tell yourself you're not insane
You fool, I hate you sometimes

Hey, you know it ain't coincidental that you're lost in place
It's drippin' off your face, and you're losin' your precious mind

Send me a postcard if you get that far
You got a couple pennies in your rusty jar
The truth you've been gone for awhile
It's hard lookin' at you when you look that way
With your one night stands and your sleep all days
Ooh you're such a slut sometimes


chorus:
Hey, you know it ain't coincidental that you're lost in place
It's drippin' off your face, and you're losin' your precious mind

You're losing your mind
 end of lyrics
```

Figure: Song lyrics

Lyrics

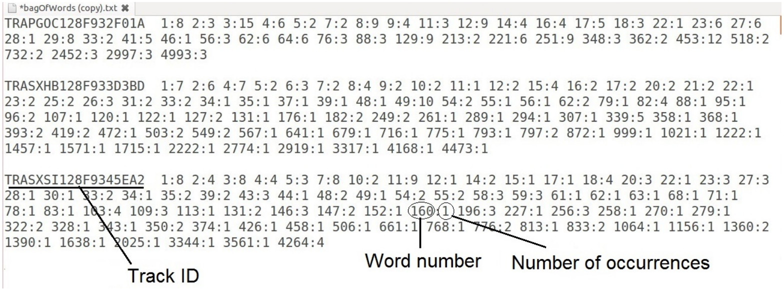
Word List

- 6800 strong word list containing two sets of words, positive and negative, was used
- Each song classified by counting number of times a word from each list appears in it
- Emotion predicted by comparing counts
- Obtained 58% accuracy

Lyrics

Bag of Words

- Used musiXMatch dataset
- Extracted bag of words from it for 500 songs using Python
- Classified using LIBLINEAR
- Obtained 53% accuracy



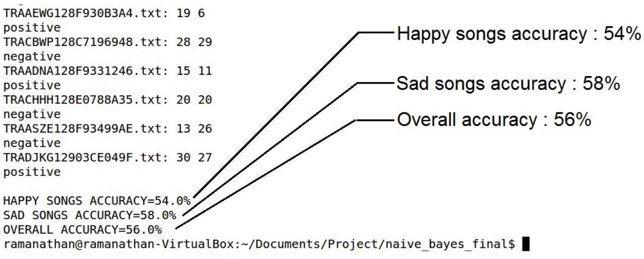
```
TRAPGOC128F932F01A  1:8 2:3 3:15 4:6 5:2 7:2 8:9 9:4 11:3 12:9 14:4 16:4 17:5 18:3 22:1 23:6 27:6
28:1 29:8 33:2 41:5 46:1 56:3 62:6 64:6 76:3 88:3 129:9 213:2 221:6 251:9 348:3 362:2 453:12 518:2
732:2 2452:3 2997:3 4993:3

TRASXHB128F933D3BD  1:7 2:6 4:7 5:2 6:3 7:2 8:4 9:2 10:2 11:1 12:2 15:4 16:2 17:2 20:2 21:2 22:1
23:2 25:2 26:3 31:2 33:2 34:1 35:1 37:1 39:1 48:1 49:10 54:2 55:1 56:1 62:2 79:1 82:4 88:1 95:1
96:2 107:1 120:1 122:1 127:2 131:1 176:1 182:2 249:2 261:1 289:1 294:1 307:1 339:5 358:1 368:1
393:2 419:2 472:1 503:2 549:2 567:1 641:1 679:1 716:1 775:1 793:1 797:2 872:1 999:1 1021:1 1222:1
1457:1 1571:1 1715:1 2222:1 2774:1 2919:1 3317:1 4168:1 4473:1

TRASXSI128F9345EA2  1:8 2:4 3:8 4:4 5:3 7:8 10:2 11:9 12:1 14:2 15:1 17:1 18:4 20:3 22:1 23:3 27:3
28:1 30:1 33:2 34:1 35:2 39:2 43:3 44:1 48:2 49:1 54:2 55:2 58:3 59:3 61:1 62:1 63:1 68:1 71:1
78:1 83:1 103:4 109:3 113:1 131:2 146:3 147:2 152:1 160:1 196:3 227:1 256:3 258:1 270:1 279:1
322:2 328:1 343:1 350:2 374:1 426:1 458:1 506:1 661:1 768:1 776:2 813:1 833:2 1064:1 1156:1 1360:2
1390:1 1638:1 2025:1 3344:1 3561:1 4264:4
```

Figure: Bag of Words

Lyrics

Naive Bayes classifier

- Java implementation of Naive Bayes Classifier used
- Training done on entire lyrics dataset



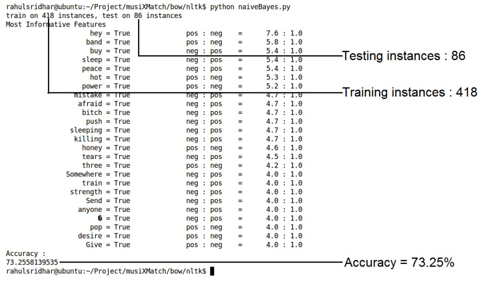
```
TRAAEWG128F930B3A4.txt: 19 6
positive
TRACBWP128C7196948.txt: 28 29
negative
TRAADNA128F9331246.txt: 15 11
positive
TRACHHH128E0788A35.txt: 20 20
negative
TRAASZE128F93499AE.txt: 13 26
negative
TRADJKG12903CE049F.txt: 30 27
positive

HAPPY SONGS ACCURACY=54.0%
SAD SONGS ACCURACY=58.0%
OVERALL ACCURACY=56.0%
ramanathan@ramanathan-VirtualBox:~/Documents/Project/naive_bayes_final$
```

Figure: Output of Naive Bayes classifier

Lyrics

NLTK

- Implementation of Naive Bayes found in NLTK used to classify songs
- Results obtained accurate up to 75% in certain cases



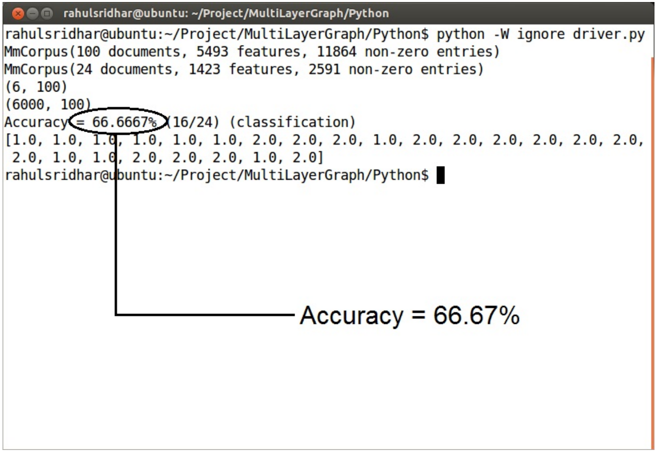
```
rahulsridhar@ubuntu:~/Project/musiXMatch/bow/nltk$ python naiveBayes.py
train on 418 instances, test on 86 instances
Most Informative Features
                     hey = True              pos : neg    =      7.6 : 1.0
                    band = True              pos : neg    =      5.8 : 1.0
                     buy = True              neg : pos    =      5.4 : 1.0
                   sleep = True              neg : pos    =      5.4 : 1.0
                   peace = True              neg : pos    =      5.4 : 1.0
                     hot = True              pos : neg    =      5.3 : 1.0
                   power = True              pos : neg    =      5.2 : 1.0
                 mistake = True              neg : pos    =      4.7 : 1.0
                  afraid = True              neg : pos    =      4.7 : 1.0
                   bitch = True              neg : pos    =      4.7 : 1.0
                    push = True              neg : pos    =      4.7 : 1.0
                sleeping = True              neg : pos    =      4.7 : 1.0
                 killing = True              neg : pos    =      4.7 : 1.0
                   honey = True              pos : neg    =      4.6 : 1.0
                   tears = True              neg : pos    =      4.5 : 1.0
                   three = True              pos : neg    =      4.2 : 1.0
               Somewhere = True              neg : pos    =      4.0 : 1.0
                   train = True              neg : pos    =      4.0 : 1.0
                strength = True              neg : pos    =      4.0 : 1.0
                    Send = True              neg : pos    =      4.0 : 1.0
                  anyone = True              neg : pos    =      4.0 : 1.0
                       � = True              neg : pos    =      4.0 : 1.0
                     pop = True              pos : neg    =      4.0 : 1.0
                  desire = True              pos : neg    =      4.0 : 1.0
                    Give = True              pos : neg    =      4.0 : 1.0
Accuracy :
73.2558139535
rahulsridhar@ubuntu:~/Project/musiXMatch/bow/nltk$ █
```

Figure: Output from the Naive Bayes classifier of NLTK for a training set

Using consensus from multi-layer graphs

- All modules implemented in Python
- Audio features and bag of words stored in separate matrices
- Given as input to main driver module and KNN graphs constructed
- *Compute embedding* module creates an embedding that represents training data
- *Sparse code* module creates sparse representation of test data
- Training and test data given as input to SVMTrain and SVMPredict

Using consensus from multi-layer graphs



```
rahulsridhar@ubuntu:~/Project/MultiLayerGraph/Python$ python -W ignore driver.py
MmCorpus(100 documents, 5493 features, 11864 non-zero entries)
MmCorpus(24 documents, 1423 features, 2591 non-zero entries)
(6, 100)
(6000, 100)
Accuracy = 66.6667% (16/24) (classification)
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 2.0, 2.0, 2.0, 1.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0,
 2.0, 1.0, 1.0, 2.0, 2.0, 2.0, 1.0, 2.0]
rahulsridhar@ubuntu:~/Project/MultiLayerGraph/Python$
```

Figure: Multi-layer graph

Conclusion and future work

- Different algorithms tried for emotion identification using audio features and lyrics
- Random Forests used for audio features
- Naive Bayes classifier in NLTK used for lyrics
- For combination, multi-layer graphs was used
- Can be tested on bigger datasets in future

References

1. K.N. Ramamurthy, A.Y. Aravkin and J.J. Thiagarajan. (2014) *Beyond L2 loss functions for learning sparse models*, ACM KDD. (Submitted)
2. X. Dong, P. Frossard, P. Vandergheynst and N. Nefedov. (2009) *Clustering on multi-layer graphs via subspace analysis on Grassmannian manifolds*
3. Dang Trung Thanh, Kiyoaki Shirai. (2009) *Machine Learning Approaches for Mood Classification of Songs toward Music Search Engine*, Japan Advanced Institute of Science and Technology.
4. Jose Padial, Ashish Goel. (2011) *Music Mood Classification*, Carnegie Mellon University.
5. A. Ng, M. Jordan, and Y. Weiss. (2001) *On spectral clustering - analysis and an algorithm*, Advances in NIPS.
6. Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman and Paul Lamere. (1983) *The Million Song Dataset*, In Proceedings of the 12th International Society for Music Information Retrieval Conference.

Thank You