

Assignment 9

CS532-s16: Web Sciences

Spring 2016

John Berlin

Generated on April 21, 2016

List of Figures

List of Tables

1	Blog Structure Count	4
2	Genre Category Count	4
3	Blog Structure Trained On For 100 Entries	5
4	Genres Trained On For 100 Entries	5
5	Blog Structure Trained On For All Entries	6
6	Genres Trained On For All Entries	6
7	Dr, Nelson Results	6
8	Genres Results	7
9	Dr Nelson 10 fold Results	13

Index

Question 1, 3
Question 2, 5
Question 3, 8
Question 4, 10
Question 5, 12

Question 1

1. Choose a blog or a newsfeed (or something similar with an Atom or RSS feed). Every student should do a unique feed, so please "claim" the feed on the class email list (first come, first served). It should be on a topic or topics of which you are qualified to provide classification training data. Find something with at least 100 entries (or items if RSS).

Create between four and eight different categories for the entries in the feed:

examples:

work, class, family, news, deals

liberal, conservative, moderate, libertarian

sports, local, financial, national, international, entertainment

metal, electronic, ambient, folk, hip-hop, pop

Download and process the pages of the feed as per the week 12 class slides.

Be sure to upload the raw data (Atom or RSS) to your github account.

Answer

The blog selected for this assignment is [f-measure](#). As a fan of music and currently looking to broaden my musical horizons, choosing f-measure was a no brainer. Another reason was that I noticed something about the blog when I first checked it out. The author Dr. Nelson, adds categories to his own blog titles. Take for instance his latest post *Merle Haggard - "Mama Tried" (forgotten song)*, with the category in parentheses being *forgotten song*. This self categorization of the posts lead me to use them for this assignment. But mainly because I wondered if his own tagging could be enough classify each post.

But when I went to count the total number of posts for each category I noticed that the category and spotlight labels were not used enough. This deficiency lead me to combine the two into a single category *concert/spotlight*. The total counts of these categories can be found in table 1. As seen in the table *lp review* and *forgotten song* dominate the categories of his blog but as seen in table 3 the categories trained on, only *the song remains the same* was less than ten and the others were well within a "healthy" number.

Now since the question asked specifically for us to create four to eight categories, I choose to classify each entry on the genre of the artist the blog post is about. To obtain the genres I created an account to access the Gracenote Music [web api](#) and utilized the library [pygn](#) to query the api for the genre of each artist. Each query of the api returned at most the top three genres for each artist, of which I choose the first one as the genre. The genres returned by each query can be seen in the file *artistInfo.json* in the datafiles folder accompanying this report.

Some of the artists in the blog posts did not have an entry in Gracenote Music or I did not agree with the top genre assigned to them. These artists namely David Bowie who was labled glam and The Nerves, Blacktask whom had no genre. I know that David Bowie and The Nerves are rock and Blacktask is metal/punk. Doing a count of the top genres showed that all the artists represented in this blog covered a vast number of sub-genres, which lead to categories that were sparse. To correct this represented my own personal logic for condensing sub-genres to a single one in a method `determine_genre` seen in the code for this assignment which can be found in listing 10. This method also combines genres into composite genres such that the categories used for this assignment are have enough data. The total count for these composite genres can be seen in table 2 and the number of composite genres trained on can be seen in table 4.

class	count
the song remains the same	28
concert/spotlight	28
forgotten song	60
lp review	81

Table 1: Blog Structure Count

class	count
Rock	42
R&B/Jazz/Mowtown/Country/Other	35
Indie Rock/Alternative	37
Metal/Punk/Hardcore	30
Pop/Electronic/Hip-Hop	53

Table 2: Genre Category Count

Question 2

2. Manually classify the first 50 entries, and then classify (using the fisher classifier) the remaining 50 entries.

Create a table with the title, predicted category, actual category, and `cprob()` and `fisherprob()` for the actual category.

Answer

In order to use `docclass.py` for python three I had to modify it slightly to remove html content using beautiful soup and it can be seen in listing 11. I also modified its usage of the databases to ensure I had a clean db each time I ran a classifier by adding a schema file(listing 9) that is read in and executed and moved the queries executed into a file called *queries.txt* which is also read in. Both files can be found in the datafiles folder.

As discussed in the answer section for question one, the manual classification portion can be automated and is done so in the code used to answer these questions which can be seen in listing 10. Per usual please consult the comments in the code for further details as in this report I will discuss the results rather than details of the code used to generate the answers.

I used both categories when running the classifiers and the number of categories trained on per each category group can be seen in tables 3 for Dr. Nelson's self categorization and table 4 for genre. The results can be seen in table 7 for Dr. Nelsons self categories and in table 8 for genres. The tables had to be shrunk to fit into this report.

I also trained and ran the classifier on a 50/50 split of the data which I explain the purpose of this in the answer to question 3. The numbers of the categories for this can be seen in tables 5 for Dr. Nelsons categories and 6 for genres.

class	count
the song remains the same	6
concert/spotlight	15
forgotten song	16
lp review	13

Table 3: Blog Structure Trained On For 100 Entries

class	count
Rock	11
R&B/Jazz/Mowtown/Country/Other	11
Indie Rock/Alternative	9
Metal/Punk/Hardcore	11
Pop/Electronic/Hip-Hop	8

Table 4: Genres Trained On For 100 Entries

class	classCount	class	classCount
lp review	29	R&B/Jazz/Mowtown/Country/Other	22
concert/spotlight	27	Metal/Punk/Hardcore	16
the song remains the same	13	Rock	23
forgotten song	29	Indie Rock/Alternative	14
		Pop/Electronic/Hip-Hop	23

Table 5: Blog Structure Trained On For All Entries

Table 6: Genres Trained On For All Entries

title	predicted_cat	acat_fprob	actual_cat
The Beatles - "I Want You (She's So Heavy)" (spotlight)	lp review	0.000900145	concert/spotlight
The Everly Brothers - "Cathy's Clown" (spotlight)	lp review	0.0110278	concert/spotlight
Peter Seeger - "Turn! Turn! Turn! (to Everything There Is a Season)" (spotlight)	lp review	0.0106328	concert/spotlight
Squeeze - "Up The Junction" (forgotten song)	lp review	0.0414196	forgotten song
Camera Obscura - "Biggest Bluest Hi-Fi" (LP Review)	lp review	0.40074	lp review
Andy Stott - "Moogfest 2012" (concert)	lp review	0.00466945	concert/spotlight
The Beastie Boys - "No Sleep Till Brooklyn" (spotlight)	lp review	0.0933825	concert/spotlight
Pink Floyd - "Live At Pompeii" (concert)	lp review	0.0449999	concert/spotlight
Negativland - "Live At Lewis's, Norfolk VA, November 21, 1992" (concert)	lp review	0.027483	concert/spotlight
Red Rider - "Lunatic Fringe" (forgotten song)	lp review	0.202943	forgotten song
The Green Pajamas - "Kim The Waitress" (forgotten song)	lp review	0.00309035	forgotten song
The Naked and Famous - "Passive Me, Aggressive You" (LP Review)	lp review	0.38523	lp review
Rachel Goswell - "Waves Are Universal" (LP Review)	lp review	0.652459	lp review
The Brains - "Money Changes Everything" (the song remains the same)	lp review	0.000158907	the song remains the same
The Beastie Boys - "The Mix-Up" (LP Review)	lp review	0.495966	lp review
Houndmouth - "Houndmouth" (LP Review)	lp review	0.129944	lp review
Husker Du - "Candy Apple Grey" (LP Review)	lp review	0.685206	lp review
Stanley Jordan - "Stairway to Heaven" (the song remains the same)	lp review	0.0893994	the song remains the same
Discharge - "Protest and Survive" (the song remains the same)	lp review	0.205744	the song remains the same
Galaxie 500 - "Peel Sessions" (LP Review)	lp review	0.191561	lp review
My Bloody Valentine - "Loveless" (LP Review)	lp review	0.604418	lp review
Sonic Youth - "Diamond Sea" (forgotten song)	lp review	0.0292781	forgotten song
Slayer - "Haunting The Chapel" (LP Review)	lp review	0.564059	lp review
Hank Williams Jr. - "All My Rowdy Friends (Have Settled Down)" (forgotten song)	the song remains the same	0.0029125	forgotten song
Unkle - "Do Androids Dream of Electric Beats?" (LP Review)	lp review	0.674837	lp review
Pink Floyd - "Cymbaline" (forgotten song)	lp review	0.00153018	forgotten song
The Crips - "Payola" (LP Review)	lp review	0.514895	lp review
Dale Watson - "Quick Quick, Slow Slow" (spotlight)	lp review	0.0399479	concert/spotlight
The Rave Ups - "Positively Lost Me" (forgotten song)	lp review	0.03228	forgotten song
Damian Marley - "Welcome To Jamrock" (spotlight)	lp review	0.00157387	concert/spotlight
Mariachi El Bronx - "Cell Mates" (spotlight)	lp review	0.0377341	concert/spotlight
Beyonce - "Single Ladies (Put a Ring on It)" (the song remains the same)	lp review	0.0105216	the song remains the same
Ass Ponys - "Little Bastard" (forgotten song)	forgotten song	0.0470319	forgotten song
This Mortal Coil - "Song to the Siren" (the song remains the same)	lp review	0.0249224	the song remains the same
School of Seven Bells - "Ghostory" (LP Review)	lp review	0.511663	lp review
DJ Shadow - "The Less You Know, The Better" (LP review)	lp review	0.994952	lp review
Waxing Poetics - "Blue-Eyed Soul" (forgotten song)	forgotten song	0.0416508	forgotten song
Zones - "Earth Grid" (LP Review)	lp review	0.885645	lp review
Matt and Kim - "Daylight" (spotlight)	lp review	0.0228493	concert/spotlight
The Dave Brubeck Quartet - "Time Out" (LP Review)	lp review	0.100313	lp review
The Beach Boys - "Heroes and Villains" (forgotten song)	lp review	0.0018678	forgotten song
Saxon - "Princess of the Night" (forgotten song)	lp review	0.082235	forgotten song
Ph Balance - "Ph Balance" (LP Review)	lp review	0.826309	lp review
Sirah - "Double Yellow Lines" (spotlight)	lp review	0.00635146	concert/spotlight
Bow Wow Wow - "I Want Candy" (forgotten song)	lp review	0.00979798	forgotten song
The Cure - "High" (forgotten song)	lp review	0.0160724	forgotten song
The Beach Boys - "Good Vibrations" (the song remains the same)	lp review	0.000164846	the song remains the same
The Equals - "Police On My Back" (the song remains the same)	lp review	0.0578675	the song remains the same
The Clash - "I Fought The Law" (The Song Remains The Same)	lp review	0.200442	the song remains the same
Catherine Wheel - "Ferment" (LP Review)	lp review	0.425535	lp review

Table 7: Dr. Nelson Results

title	predicted_cat	acat_fprob	actual_cat
The Beatles - "I Want You (She's So Heavy)" (spotlight)	Indie Rock/Alternative	5.81592e-05	Rock
The Everly Brothers - "Cathy's Clown" (spotlight)	Indie Rock/Alternative	0.00348602	Rock
Peter Seeger - "Turn! Turn! Turn! (to Everything There Is a Season)" (spotlight)	Metal/Punk/Hardcore	0.00042724	Rock
Squeeze - "Up The Junction" (forgotten song)	Indie Rock/Alternative	0.0127401	Indie Rock/Alternative
Camera Obscura - "Biggest Bluest Hi-Fi" (LP Review)	Indie Rock/Alternative	0.02337	Indie Rock/Alternative
Andy Stott - "Moogfest 2012" (concert)	Indie Rock/Alternative	3.31013e-05	Pop/Electronic/Hip-Hop
The Beastie Boys - "No Sleep Till Brooklyn" (spotlight)	Metal/Punk/Hardcore	2.08461e-07	Pop/Electronic/Hip-Hop
Pink Floyd - "Live At Pompeii" (concert)	Indie Rock/Alternative	0.000290808	Rock
Negativland - "Live at Lewis's, Norfolk VA, November 21, 1992" (concert)	Indie Rock/Alternative	0.0707163	Indie Rock/Alternative
Red Rider - "Lunatic Fringe" (forgotten song)	Indie Rock/Alternative	0.0778946	Rock
The Green Pajamas - "Kim The Waitress" (forgotten song)	Indie Rock/Alternative	0.237983	Indie Rock/Alternative
The Naked and Famous - "Passive Me, Aggressive You" (LP Review)	Indie Rock/Alternative	0.0514449	Indie Rock/Alternative
Rachel Goswell - "Waves Are Universal" (LP Review)	Indie Rock/Alternative	1.0004e-05	Rock
The Brains - "Money Changes Everything" (the song remains the same)	Indie Rock/Alternative	0.0405007	Indie Rock/Alternative
The Beastie Boys - "The Mix-Up" (LP Review)	Indie Rock/Alternative	4.6062e-09	Pop/Electronic/Hip-Hop
Houndmouth - "Houndmouth" (LP Review)	Indie Rock/Alternative	0.000640891	R&B/Jazz/Mowtown/Country/Other
Husker Du - "Candy Apple Grey" (LP Review)	Indie Rock/Alternative	0.12089	Indie Rock/Alternative
Stanley Jordan - "Stairway to Heaven" (the song remains the same)	Indie Rock/Alternative	0.0133826	R&B/Jazz/Mowtown/Country/Other
Discharge - "Protest and Survive" (the song remains the same)	Indie Rock/Alternative	0.0523455	Metal/Punk/Hardcore
Galaxie 500 - "Peel Sessions" (LP Review)	Indie Rock/Alternative	0.0964701	Indie Rock/Alternative
My Bloody Valentine - "Loveless" (LP Review)	Indie Rock/Alternative	0.0219168	Metal/Punk/Hardcore
Sonic Youth - "Diamond Sea" (forgotten song)	Indie Rock/Alternative	0.0844261	Indie Rock/Alternative
Slayer - "Haunting The Chapel" (LP Review)	Indie Rock/Alternative	0.0157278	Metal/Punk/Hardcore
Hank Williams Jr. - "All My Rowdy Friends (Have Settled Down)" (forgotten song)	Indie Rock/Alternative	0.000494957	R&B/Jazz/Mowtown/Country/Other
Unkle - "Do Androids Dream of Electric Beats?" (LP Review)	Indie Rock/Alternative	7.18402e-07	Pop/Electronic/Hip-Hop
Pink Floyd - "Cymbaline" (forgotten song)	Indie Rock/Alternative	6.06206e-06	Rock
The Cribs - "Payola" (LP Review)	Indie Rock/Alternative	0.254918	Indie Rock/Alternative
Dale Watson - "Quick Quick, Slow Slow" (spotlight)	Indie Rock/Alternative	0.000510025	R&B/Jazz/Mowtown/Country/Other
The Rave Ups - "Positively Lost Me" (forgotten song)	Indie Rock/Alternative	1.17354e-06	Pop/Electronic/Hip-Hop
Damian Marley - "Welcome To Jamrock" (spotlight)	Indie Rock/Alternative	0.00014292	R&B/Jazz/Mowtown/Country/Other
Mariachi El Bronx - "Cell Mates" (spotlight)	Indie Rock/Alternative	0.114283	Indie Rock/Alternative
Beyonce - "Single Ladies (Put a Ring on It)" (the song remains the same)	Indie Rock/Alternative	0.00324419	R&B/Jazz/Mowtown/Country/Other
Ass Pony - "Little Bastard" (forgotten song)	Indie Rock/Alternative	0.0387984	Indie Rock/Alternative
This Mortal Coil - "Song to the Siren" (the song remains the same)	Indie Rock/Alternative	0.301464	Indie Rock/Alternative
School of Seven Bells - "Ghlostory" (LP Review)	Indie Rock/Alternative	0.0352295	Indie Rock/Alternative
DJ Shadow - "The Less You Know, The Better" (LP review)	Indie Rock/Alternative	1.18196e-08	Pop/Electronic/Hip-Hop
Waxing Poetics - "Blue-Eyed Soul" (forgotten song)	Indie Rock/Alternative	5.96e-05	Rock
Zones - "Earth Grid" (LP Review)	Indie Rock/Alternative	0.150787	Indie Rock/Alternative
Matt and Kim - "Daylight" (spotlight)	Indie Rock/Alternative	0.230078	Indie Rock/Alternative
The Dave Brubeck Quartet - "Time Out" (LP Review)	Indie Rock/Alternative	2.78336e-05	R&B/Jazz/Mowtown/Country/Other
The Beach Boys - "Heroes and Villains" (forgotten song)	Indie Rock/Alternative	0.00348093	Rock
Saxon - "Princess of the Night" (forgotten song)	Indie Rock/Alternative	0.00458431	Metal/Punk/Hardcore
Ph Balance - "Ph Balance" (LP Review)	Indie Rock/Alternative	3.61317e-06	Pop/Electronic/Hip-Hop
Sirah - "Double Yellow Lines" (spotlight)	Indie Rock/Alternative	0.000252111	Metal/Punk/Hardcore
Bow Wow Wow - "I Want Candy" (forgotten song)	Indie Rock/Alternative	0.16264	Indie Rock/Alternative
The Cure - "High" (forgotten song)	Indie Rock/Alternative	0.161629	Indie Rock/Alternative
The Beach Boys - "Good Vibrations" (the song remains the same)	Indie Rock/Alternative	5.23178e-05	Rock
The Equals - "Police On My Back" (the song remains the same)	Indie Rock/Alternative	0.0135199	R&B/Jazz/Mowtown/Country/Other
The Clash - "I Fought The Law" (The Song Remains The Same)	Metal/Punk/Hardcore	0.193105	Metal/Punk/Hardcore
Catherine Wheel - "Ferment" (LP Review)	Indie Rock/Alternative	0.0266902	Indie Rock/Alternative

Table 8: Genres Results

Question 3

3. Assess the performance of your classifier in each of your categories by computing precision, recall, and F-measure.

Answer

The results of the classification for Dr. Nelson categorization can be seen in listing 1 and for genres in listing 2. These reports were generated by scikit-learn *classification_report*. Using Dr. Nelson's categories did not give us great results nor did using the genres. The precision for forgotten song was 1.0 with as f1-score of .27. Recall for lp review was also 1.0 with a f1-score of .53 as its precision was .36 whereas the rest were very low. As seen in the table showing the blog structure counts both of these were the most present. For genres Indie Rock/Alternative had a recall of 1.0, precision of .36 and f1-score of .53. Metal/Punk/Hardcore had precision of 0.33, recall of 0.17 and f1-score of 0.22. I can only attribute this to the limited amount of training data and how that each post does not necessarily talk about the genre or self categorization that much. To test this I re-ran the classification for both using a 50,50 split of the data and the results can be seen in listings 3 for Dr. Nelsons self categories and 4 for genres.

Sadly the numbers for the aforementioned categories only went up. The only conclusion from this is that using only the words of the blog is not good enough to classify them.

	precision	recall	f1-score	support
forgotten song	1.00	0.15	0.27	13
concert/spotlight	0.00	0.00	0.00	12
lp review	0.36	1.00	0.53	17
the song remains the same	0.00	0.00	0.00	8
avg / total	0.38	0.38	0.25	50

Listing 1: Scores Dr. Nelson Categorization

	precision	recall	f1-score
Rock	0.00	0.00	0.00
R&B/Jazz/Mowtown/Country/Other	0.00	0.00	0.00
Metal/Punk/Hardcore	0.33	0.17	0.22
Indie Rock/Alternative	0.40	1.00	0.58
Pop/Electronic/Hip-Hop	0.00	0.00	0.00
avg / total	0.19	0.40	0.25

Listing 2: Scores Genre Categorization

	precision	recall	f1-score	support

3	forgotten song	1.00	0.06	0.12	31
4	concert/spotlight	0.00	0.00	0.00	1
5	lp review	0.54	1.00	0.70	52
6	the song remains the same	0.00	0.00	0.00	15
7					
8	avg / total	0.60	0.55	0.41	99

Listing 3: Scores Dr. Nelson Categorization All

1		precision	recall	f1-score
2				
3	Rock	0.00	0.00	0.00
4	R&B/Jazz/Mowtown/Country/Other	0.00	0.00	0.00
5	Metal/Punk/Hardcore	1.00	0.08	0.15
6	Indie Rock/Alternative	0.41	1.00	0.58
7	Pop/Electronic/Hip-Hop	1.00	0.05	0.09
8				
9	avg / total	0.50	0.42	0.27

Listing 4: Scores Genre Categorization All

Question 4

4. Redo the questions above, but with the extensions on slide 27 and pp. 136--138.

Answer

The results of the classification for Dr. Nelson categorization can be seen in listing 5 and for genres in listing 6. I re-ran the classification for both using a 50,50 split of the data for extended as well and the results can be seen in listings 7 for Dr. Nelsons self categories and 8 for genres.

	precision	recall	f1-score	support
forgotten song	1.00	0.23	0.38	13
concert/spotlight	0.00	0.00	0.00	12
lp review	0.39	1.00	0.56	17
the song remains the same	0.33	0.12	0.18	8
avg / total	0.44	0.42	0.32	50

Listing 5: Scores Dr. Nelson Categorization Extended

	precision	recall	f1-score
Rock	0.00	0.00	0.00
R&B/Jazz/Mowtown/Country/Other	0.00	0.00	0.00
Metal/Punk/Hardcore	0.00	0.00	0.00
Indie Rock/Alternative	0.40	1.00	0.57
Pop/Electronic/Hip-Hop	0.00	0.00	0.00
avg / total	0.15	0.38	0.22

Listing 6: Scores Genre Categorization Extended

	precision	recall	f1-score	support
forgotten song	1.00	0.26	0.41	31
concert/spotlight	0.00	0.00	0.00	1
lp review	0.58	1.00	0.73	52
the song remains the same	0.00	0.00	0.00	15
avg / total	0.62	0.61	0.51	99

Listing 7: Scores Dr. Nelson Categorization All Extended

	precision	recall	f1-score
Rock	0.00	0.00	0.00
R&B/Jazz/Mowtown/Country/Other	0.00	0.00	0.00
Metal/Punk/Hardcore	1.00	0.08	0.15
Indie Rock/Alternative	0.42	1.00	0.59
Pop/Electronic/Hip-Hop	1.00	0.10	0.17

9	avg / total	0.50	0.43	0.29
---	-------------	------	------	------

Listing 8: Scores Genre Categorization All Extended

Question 5

5. A 1:1 split for training:test data typically not a good split; 5:1 or even 10:1 is preferable. We also typically use something called "10-fold cross validation" to make sure we spread the training out and don't "overfit" on a particular sequence of training data.

Rerun questions 2 & 3, but manually classifying all 100 documents, then using 90 for training and 10 for testing. Use 10-fold cross validation and generate the table from Q2, but this time with the average of all 10 values. What was the change, if any, in precision and recall (and thus F-Measure)?

Answer

metric	mean
precision	0.303765
f1	0.301631
recall	0.386904

metric	mean
precision	0.254458
f1	0.265861
recall	0.357708

acet_fprob	actual_cat	title	predicted_cat
0.00829925	forgotten song	Merle Haggard - "Mama Tried" (forgotten song)	the song remains the same
0.000586942	the song remains the same	INXS - "Don't Change" (the song remains the same)	lp review
1.42309e-05	forgotten song	The Time - "Jungle Love" (forgotten song)	lp review
0.000142731	concert/spotlight	The Eagles - "Seven Bridges Road" (spotlight)	lp review
1.17538e-05	concert/spotlight	David Bowie - "Blackstar" (spotlight)	lp review
7.6536e-08	forgotten song	Hawkwind - "Silver Machine" (forgotten song)	lp review
0.885591	lp review	Waxahatchee - "Cerulean Salt" (LP Review)	lp review
0.942373	lp review	Balam Acab - "See Birds" (LP Review)	lp review
0.025489	concert/spotlight	Gina Chavez - Live NPR Tiny Desk Concert 2015-09-22 (concert)	lp review
0.00238738	forgotten song	The Robbin Thompson Band - "Candy Apple Red" (forgotten song)	lp review
0.00148934	concert/spotlight	Avett Brothers - "Kick Drum Heart" (spotlight)	lp review
0.00358476	concert/spotlight	Waxahatchee - Live KEXP 2015-05-03 (concert)	lp review
4.12992e-06	the song remains the same	Galaxie 500 - "Rain" (the song remains the same)	lp review
1.02912e-08	the song remains the same	LCD Soundsystem - "All My Friends" (the song remains the same)	lp review
0.0228689	forgotten song	Translator - "Everywhere That I'm Not" (forgotten song)	forgotten song
0.858726	lp review	Andy Stott - "Luxury Problems" (LP Review)	lp review
0.536274	lp review	Connan Mockasin - "Caramel" (LP Review)	lp review
0.3225539	lp review	Minutemen - "Dumb Nicks on the Dime" (LP review)	lp review
0.0118787	forgotten song	Split Enz - "History Never Repeats" (forgotten song)	forgotten song
0.00017994	forgotten song	Hum - "Stars" (forgotten song)	lp review
0.453167	lp review	Celtic Frost - "Morbid Tales" (LP Review)	lp review
0.0551286	lp review	The Beach Boys - "Wild Honey" (LP Review)	lp review
0.0453624	the song remains the same	Lesley Gore - "You Don't Own Me" (the song remains the same)	the song remains the same
9.82159e-10	concert/spotlight	Avett Brothers - Austin, TX 2014-10-11 (concert)	lp review
7.2537e-05	forgotten song	Hill Sobule - "I Kissed A Girl" (forgotten song)	lp review
0.102236	lp review	Iron Maiden - "Iron Maiden" (LP Review)	lp review
3.31497e-05	concert/spotlight	The Avett Brothers - Raleigh, NC 2014-12-31 (concert)	lp review
0.0029604	forgotten song	States - "My Latest Girl" (forgotten song)	lp review
0.001140737	forgotten song	Utopia - "Feet Don't Fail Me Now" (forgotten song)	lp review
0.00377996	forgotten song	Jim Ruffin - "What Becomes of the Broken Hearted?" (forgotten song)	lp review
0.883463	lp review	Waxahatchee - "American Weekend" (LP Review)	lp review
0.0169882	the song remains the same	Blackstreet - "No Diggity" (the song remains the same)	lp review
0.000582417	forgotten song	Ani DiFranco - "32 Flavors" (forgotten song)	lp review
2.00316e-05	forgotten song	Let's Active - "Every Word Means No" (forgotten song)	lp review
0.000150813	concert/spotlight	The Specials - "Rock Goes to College (1979)" (concert)	lp review
3.15345e-06	concert/spotlight	St. Paul & The Broken Bones - Live KEXP 2014-04-19 (concert)	lp review
3.11289e-06	concert/spotlight	Horseback - "Live at Nightlights 2011-11" (concert)	lp review
0.000203459	concert/spotlight	Motorhead - "Ace of Spades" (spotlight)	lp review
2.85102e-07	concert/spotlight	Motorhead - "R.A.M.O.N.E.S." (spotlight)	lp review
1.43843e-05	concert/spotlight	We Were Promised Jetpacks - "Live in Nashville, 2012-03-29" (concert)	lp review
0.30056	lp review	Camera Obscura - "My Manolin Career" (LP Review)	lp review
0.530964	lp review	Camera Obscura - "My Manolin Career" (LP Review)	lp review
0.00522835	forgotten song	Ultravox - "Vienna" (forgotten song)	forgotten song
0.00276339	the song remains the same	Queen & David Bowie - "Under Pressure" (the song remains the same)	lp review
0.56915	lp review	Times New Viking - "Dancer Equired!" (LP Review)	lp review
4.81931e-06	concert/spotlight	Wire - "On The Box: 1979" (concert)	lp review
1.82821e-07	concert/spotlight	GWAR - "Phallus in Wonderland" (spotlight)	lp review
0.215787	lp review	Mission of Burma - "Signals, Cabs, and Marches" (LP Review)	lp review
0.000241497	forgotten song	Neil Young and Devo - "Hey Hey, My My (Into the Black)" (forgotten song)	lp review
2.95192e-05	forgotten song	Stevie Wonder - "Higher Ground" (forgotten song)	lp review
1.72047e-07	concert/spotlight	The Beatles - "I Want You (She's So Heavy)" (spotlight)	lp review
2.22158e-05	concert/spotlight	The Everly Brothers - "Cathy's Clown" (spotlight)	lp review
5.23079e-06	concert/spotlight	Peter Seeger - "Turn! Turn! Turn! (to Everything There Is a Season)" (spotlight)	lp review
0.000115866	forgotten song	Squeeze - "Up The Junction" (forgotten song)	lp review
0.16036	lp review	Camera Obscura - "Biggest Bluest Hi-Fi" (LP Review)	lp review
5.23624e-05	concert/spotlight	Andy Stott - "Moogfest 2012" (concert)	lp review
7.69054e-05	concert/spotlight	The Beastie Boys - "No Sleep Till Brooklyn" (spotlight)	lp review
2.52318e-07	concert/spotlight	Pink Floyd - "Live At Pompeii" (concert)	lp review
1.23764e-07	concert/spotlight	Negativland - "Live at Lewis's, Norfolk VA, November 21, 1992" (concert)	lp review
0.0127229	forgotten song	Red Rider - "Lunatic Fringe" (forgotten song)	lp review
1.86416e-09	forgotten song	The Green Pajamas - "Kim The Waitress" (forgotten song)	lp review
0.317351	lp review	The Naked and Famous - "Passive Me, Aggressive You" (LP Review)	lp review
0.761564	lp review	Rachel Goswell - "Waves Are Universal" (LP Review)	lp review
0.00139494	the song remains the same	The Brains - "Money Changes Everything" (the song remains the same)	lp review
0.769315	lp review	The Beastie Boys - "The Mix-Up" (LP Review)	lp review
0.0756969	lp review	Houndmouth - "Houndmouth" (LP Review)	lp review
0.699192	lp review	Husker Du - "Candy Apple Grey" (LP Review)	lp review
0.024669	the song remains the same	Stanley Jordan - "Stairway to Heaven" (the song remains the same)	lp review
0.142757	the song remains the same	Discharge - "Protest and Survive" (the song remains the same)	lp review
0.286404	lp review	Galaxie 500 - "Feel Sessions" (LP Review)	lp review
0.712191	lp review	My Bloody Valentine - "Loveless" (LP Review)	lp review
1.34057e-05	forgotten song	Sonic Youth - "Diamond Sea" (forgotten song)	lp review
0.270206	lp review	Slayer - "Haunting The Chapel" (LP Review)	lp review
1.27901e-05	forgotten song	Hank Williams Jr. - "All My Rowdy Friends (Have Settled Down)" (forgotten song)	lp review
0.86955	lp review	Uncle - "Do Androids Dream of Electric Beats?" (LP Review)	lp review
2.3542e-07	forgotten song	Pink Floyd - "Cymbaline" (forgotten song)	lp review
0.858767	lp review	The Cribz - "Pavlov" (LP Review)	lp review
7.90432e-08	concert/spotlight	Dale Watson - "Quick Quick, Slow Slow" (spotlight)	lp review
0.00204828	forgotten song	The Rave Ups - "Positively Lost Me" (forgotten song)	lp review
6.37284e-12	concert/spotlight	Damian Marley - "Welcome To Jamrock" (spotlight)	lp review
0.000374377	concert/spotlight	Mariachi El Bronx - "Cell Mates" (spotlight)	lp review
0.00515597	the song remains the same	Byrnes - "Single Ladies (Put a Ring on It)" (the song remains the same)	lp review
0.00289288	forgotten song	As Pony - "Little Bastard" (forgotten song)	lp review
0.000952902	the song remains the same	This Mortal Coil - "Song to the Siren" (the song remains the same)	lp review
0.214369	lp review	School of Seven Bells - "Ghostory" (LP Review)	lp review
0.964615	lp review	DJ Shadow - "The Less You Know, The Better" (LP review)	lp review
4.95668e-05	forgotten song	Waxing Poetics - "Blue-Eyed Soul" (forgotten song)	lp review
0.945019	lp review	Zones - "Earth Grid" (LP Review)	lp review
0.000120524	concert/spotlight	Matt and Kim - "Daylight" (spotlight)	lp review
0.0989495	lp review	The Dave Brubeck Quartet - "Time Out" (LP Review)	lp review
6.15297e-06	forgotten song	The Beach Boys - "Heroes and Villains" (forgotten song)	lp review
0.000598057	forgotten song	Saxon - "Princess of the Night" (forgotten song)	lp review
0.827962	lp review	Ph Balance - "Ph Balance" (LP Review)	lp review
2.58567e-05	concert/spotlight	Sirah - "Double Yellow Lines" (spotlight)	lp review
0.000137517	forgotten song	Bow Wow Wow - "I Want Candy" (forgotten song)	lp review
0.000341937	forgotten song	The Cure - "High" (forgotten song)	lp review
1.23812e-05	the song remains the same	The Beach Boys - "Good Vibrations" (the song remains the same)	lp review
0.0195592	the song remains the same	The Equals - "Police On My Back" (the song remains the same)	lp review
0.06920798	the song remains the same	The Clash - "I Fought The Law" (The Song Remains The Same)	lp review
0.293363	lp review	Catherine Wheel - "Ferment" (LP Review)	lp review
0.0205435	lp review	Velocity Girl - "Velocity Girl" (LP Review)	lp review
0.299915	lp review	Yngwie Malmsteen - "Rising Force" (LP Review)	lp review
6.73502e-05	forgotten song	Deep Purple - "Child In Time" (forgotten song)	lp review
0.945474	lp review	The Caretaker - "An Empty Bliss Beyond This World" (LP Review)	lp review
0.340475	lp review	Husker Du - "Land Speed Record" (LP Review)	lp review
0.490925	lp review	Black Sabbath - "Born Again" (LP Review)	lp review
0.82963	lp review	The Magnetic Fields - "Distortion" (LP Review)	lp review
0.59083	lp review	The Magnetic Fields - "69 Love Songs" (LP Review)	lp review
0.267212	lp review	Mazzy Star - "Common Burn/Lay Myself Down" (LP Review)	lp review
0.773815	lp review	DJ Shadow & Out Chemist - "Product Placement" (LP Review)	lp review
0.891037	lp review	DJ Shadow - "Preemptive Strike" (LP Review)	lp review
0.000314281	forgotten song	Jim Carroll - "People Who Died" (forgotten song)	the song remains the same
1.71236e-08	the song remains the same	Townes Van Zandt - "Pancho and Lefty" (the song remains the same)	lp review
4.95283e-06	forgotten song	Montrose - "I Got The Fire" (forgotten song)	lp review
0.593258	lp review	Rainbow - "Rising" (LP Review)	lp review
0.843885	lp review	Autichart - "Pool Session" (LP Review)	lp review
0.00120013	forgotten song	The Clash - "Straight to Hell" (forgotten song)	lp review
1.42811e-05	forgotten song	Roseanne Cash - "Seven Year Ache" (forgotten song)	lp review
0.632481	lp review	Negativland - "U2" (LP Review)	lp review
4.41345e-05	the song remains the same	Ice-T - "99 Problems" (the song remains the same)	lp review
0.493556	lp review	Liz Phair - "Juvenilia" (LP Review)	lp review
0.896297	lp review	Zones - "Zones" (LP Review)	lp review
0.773329	lp review	The Cribz - "Ignore the Ignorant" (LP Review)	lp review
0.0132366	forgotten song	States - "Picture Me With You" (forgotten song)	lp review
0.637912	lp review	Mayer Hawthorne - "A Strange Arrangement" (LP Review)	lp review
0.7293	lp review	Deathprod - "Treetop Drive 1-3, Towboat" (LP Review)	lp review
1.03802e-05	forgotten song	Blink-182 - "Josie" (forgotten song)	lp review
3.0826e-05	the song remains the same	R.E.M. - "Superman" (the song remains the same)	lp review
0.0806372	the song remains the same	Brubeck Stringquartet - "The Ghost of Tom Joad" (the song remains the same)	lp review
0.801063	lp review	Lissy Trullie - "Self-Taught Learner" (LP Review)	lp review
0.732728	lp review	Perfume Tree - "Tides' Out" (LP Review)	lp review
0.560927	lp review	Ultra Orange & Emmannuelle - "Ultra Orange & Emmannuelle" (LP Review)	lp review
0.421679	lp review	Arctic Monkeys - "Whatever People Say I Am, That's What I'm Not" (LP Review)	lp review

```

1 DROP TABLE IF EXISTS feed;
2 DROP TABLE IF EXISTS feature_count;
3 DROP TABLE IF EXISTS category_count;
4
5 CREATE TABLE IF NOT EXISTS feed(
6   num integer,
7   entry text,
8   feature text,
9   predicted text,
10  actual text,
11  cprob decimal
12 );
13
14 CREATE TABLE IF NOT EXISTS feature_count(
15   feature text,
16   category text,
17   count integer
18 );
19
20 CREATE TABLE IF NOT EXISTS category_count(
21   category text,
22   count integer
23 );
24
25 delete from feed;
26 delete from feature_count;
27 delete from category_count;

```

Listing 9: Database Schema

```

1 import json
2 import os
3 import statistics
4 from collections import Counter, defaultdict
5
6 import feedparser
7 import requests
8 from feedgen.feed import FeedGenerator
9 from sklearn.cross_validation import KFold
10 from sklearn.metrics import classification_report
11 from sklearn.metrics import f1_score, precision_score, recall_score
12 from tabulate import tabulate
13
14 import pygn
15 from docclass import *
16
17 fmeasure = "http://f-measure.blogspot.com/feeds/posts/default?max-
    results=200"
18 # regex to capture the self labeled topic of the blog post
19 findClass = re.compile("^.+\(((+)\)\)$")
20
21 # extract the artist portion. Capture everything until our negative
    look ahead says we have a space - space "
22 artistsExtractor = re.compile("(?!\\s\\-\\s\\")([a-zA-Z0-9.&\\-']+\\s
    (?:[a-zA-Z0-9.&\\-']+\\s)*)"
23
24 # Gracenote Music Web API user id
25 # used for pygen in order to get the genre of the artists
26 gnmUID = "put yours here"
27
28
29 def check_next(text):
30     # check for the next button ie pagination of blog pages
31     soup = BeautifulSoup(text, "xml-xml")
32     next_page = soup.find_all('link', attrs={'type': 'application/
        atom+xml', 'rel': 'next'})
33     # if there is a next page our next_page list will always be 1
        otherwise its 0
34     # that means we have consumed all the pages for the blog
35     if len(next_page) > 0:
36         nl = next_page[0].attrs['href']
37         return True, nl
38     return False, None
39
40
41 def getDataFeed():
42     # have a useragent so we do not look like a robot
43     useragent = 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:45.0)
        Gecko/20100101 Firefox/45.0'
44     sesh = requests.Session() # type: requests.Session
45     sesh.headers.update({'User-Agent': useragent})
46     r = sesh.get(fmeasure)
47     # ok to make the gotten feed slimmer I will be building a new
        one
48     # containing only the title of the blog, the feed it
49     # and the entries so I am using feedgen library to do so
50     fg = FeedGenerator()

```



```

51 feed = feedparser.parse(r.text)
52 fg.title(feed.feed.title)
53 fg.id(feed.feed.id)
54 entries = []
55 # concatenate every entry from the current pagination
56 entries.extend(feed.entries)
57 # as usual check next and while good get next set of entries
58 # and extract the entries
59 good, nl = check_next(r.text)
60 while good:
61     r = sesh.get(nl)
62     feed = feedparser.parse(r.text)
63     entries.extend(feed.entries)
64     good, nl = check_next(r.text)
65     r.close()
66 # for each of the entries
67 for e in entries:
68     # create a new entry
69     fe = fg.add_entry()
70     # add the entry id, title and content
71     fe.id(e.id)
72     fe.title(e.title)
73     c = e.content[0]
74     fe.content(content=c.value, type=c.type)
75 # write the new feed file out
76 fg.atom_file("datafiles/f-measure.xml", pretty=True)
77 sesh.close()
78 # now to get the genres
79 get_genres()
80
81
82 def genre_sanity(it):
83     '''
84     This method is a quick overview of how
85     I reduce genres down to a single one
86     Yes its a mess but hey
87     '''
88     if "New Wave" in it:
89         return "Indie Rock/Alternative"
90     if "Indie Rock" in it:
91         return "Indie Rock/Alternative"
92     if "Punk" in it:
93         return "Metal/Punk/Hardcore"
94     if "Rock" in it:
95         return "Rock"
96     if "Pop" in it:
97         return "Pop/Electronic/Hip-Hop"
98     if "Techno" in it or "Electronica" in it \
99         or "Intelligent (IDM)" in it or "Downtempo, Lounge &
100         Ambient" in it \
101         or "Trip Hop" in it:
102         return "Pop/Electronic/Hip-Hop"
103     if "Electronic" in it:
104         return "Pop/Electronic/Hip-Hop"
105     if "Metal" in it:
106         return "Metal/Punk/Hardcore"
107     if "Emo" in it or "Hardcore" in it:

```

```

107         return "Metal/Punk/Hardcore"
108     if "Hip-Hop" in it:
109         return "Pop/Electronic/Hip-Hop"
110     if "Country" in it or "Comedy" in it or "Classical" in it or "
Americana" in it:
111         return "R&B/Jazz/Mowtown/Country/Other"
112     if "R&B" in it or "Jazz" in it or "Mowtown" in it or "Soul" in
it \
113         or "Reggae" in it or "Ska" in it or "Urban" in it or "
Funk" in it:
114         return "R&B/Jazz/Mowtown/Country/Other"
115     if "Lo-Fi" in it or "Shoegazer" in it or "Slowcore" in it or "
Neo-Psychedelic" in it or "Alternative" in it:
116         return "Indie Rock/Alternative"
117     if "Classic Prog" in it:
118         return "Rock"
119     if "Post-Modern Art" in it or "Bakersfield Sound":
120         return "R&B/Jazz/Mowtown/Country/Other"
121     return it
122
123
124 def determine_genre(artist, genreList):
125     # The genre is the first one in the list as it is the dominate
one
126     # I denote the first genre in the list as dominate as it is the
most relevant
127     # if the list is empty say none
128     genre = genreList[0] if len(genreList) != 0 else None
129     # this portion I by hand say what genre an artist is
130     # they have David Bowie as Glam technically yes
131     # he is fabulous but seriously Rock
132     if "My Bloody Valentine" in artist:
133         return "Metal/Punk/Hardcore"
134     if "Ultravox" in artist:
135         return "Rock"
136     if "DJ Shadow" in artist:
137         return "Pop/Electronic/Hip-Hop"
138     if "David Bowie" in artist:
139         return "Rock"
140     if genre is None:
141         # genre is none happens sometime for the various artists
blog entries
142         # but I know these artists happen with it
143         # otherwise I do not know how to classify the genre so
default
144         if "The Nerves" in artist:
145             return "Rock"
146         if "Blacktask" in artist:
147             return "Metal/Punk/Hardcore"
148         return "R&B/Jazz/Mowtown/Country/Other"
149     # sanitize the genre after everything
150     return genre_sanity(genre)
151
152
153 def get_genres():
154     # read the newly created feed
155     feed = feedparser.parse("datafiles/f-measure.xml")

```

```

156
157 # want the set of unique artists
158 uniqueArtists = set()
159
160 # get the artists from the feed
161 for e in feed.entries:
162     m = artistsExtractor.match(e.title)
163     if m is not None:
164         # my regex grabs the trailing space dash in some cases
165         # so remove it
166         uniqueArtists.add(m.group(1).rstrip(" -"))
167 # set up for getting the unique artist genre
168 userID = pygn.register(gnmUID)
169 artistsInfo = {} # store all associated genre information per
artist
170 artistsToGenre = {} # store the direct mapping of the chosen
genre
171 # for each artist
172 for ua in uniqueArtists:
173     # get their metadata
174     metaData = pygn.search(clientID=gnmUID, userID=userID,
artist=ua)
175     uaG = []
176     print(ua)
177     # if its not none then Gracenote Music has information on
them
178     if metaData is not None:
179         # extract the genre for the artists
180         for genre in map(lambda ge: ge['TEXT'], metaData['genre
'].values()):
181             uaG.append(genre)
182             artistsInfo[ua] = uaG
183             artistsToGenre[ua] = determine_genre(ua, uaG)
184 # write data to file
185 with open("datafiles/artistsInfo.json", "w+") as out:
186     out.write(json.dumps(artistsInfo, indent=1))
187 with open("datafiles/artistsToGenre.json", "w+") as out:
188     out.write(json.dumps(artistsToGenre, indent=1))
189
190
191 def labelCounter():
192     # I want to be able to know what how many labels i have
193     # for reporting and sanity checking
194     # load the artist genre information
195     with open("datafiles/artistsInfo.json", "r") as ai:
196         artistsData = json.load(ai)
197     gCounter = Counter()
198     artistsToGenre = {}
199     # count genre and map it to artist
200     for artist, genres in sorted(artistsData.items(), key=lambda x:
x[0]):
201         genre = determine_genre(artist, genres)
202         gCounter[genre] += 1
203         print(artist, genre, genres)
204         artistsToGenre[artist] = genre
205         print("_____")
206

```

```

207 genreToEntry = defaultdict(list)
208 feed = feedparser.parse("datafiles/f-measure.xml")
209 gc = Counter()
210 gcc = 0
211 # count number of genres for all feed entries
212 # and find out which genres we are training on
213 for e in feed.entries:
214     m = artistsExtractor.match(e.title)
215     g = None
216     if m is not None:
217         art = m.group(1).rstrip(" -")
218         genreToEntry[artistsToGenre[art]].append(e)
219         g = artistsToGenre[art]
220     else:
221         genreToEntry["R&B/Jazz/Mowtown/Country/Other"].append(e)
222
223     g = "R&B/Jazz/Mowtown/Country/Other"
224     if gcc < 50:
225         gc[g] += 1
226         gcc += 1
227
228 print(gc)
229 fgl = []
230 fcl = []
231 # for the first 50 training items table
232 for fg, fc in gc.items():
233     fgl.append(fg)
234     fcl.append(fc)
235
236 with open("datafiles/genreFirst50LatexTable.txt", "w+") as lout:
237     lout.write(tabulate({"class": fgl, "classCount": fcl},
238         headers="keys", tablefmt="latex"))
239
240 # for all items
241 genrel = []
242 genrecl = []
243 for g, gl in genreToEntry.items():
244     print(g, len(gl))
245     genrel.append(g)
246     genrecl.append(len(gl))
247
248 with open("datafiles/genreAllLatexTable.txt", "w+") as lout:
249     lout.write(tabulate({"class": genrel, "classCount": genrecl},
250         headers="keys", tablefmt="latex"))
251
252 # my original labels reporting section
253 oToE = defaultdict(list)
254 count = 0
255 cc = Counter()
256
257 for e2 in feed.entries:
258     clazz = findClass.match(e2.title).group(1).lower().strip()
259     if "concert" in clazz or 'spotlight' in clazz:
260         clazz = 'concert/spotlight'
261     oToE[clazz].append(e2)
262     if count < 50:

```

```

260         cc[clazz] += 1
261         count += 1
262     print("_____")
263     print(cc)
264
265     fl = []
266     cl = []
267     for fg, fc in cc.items():
268         fl.append(fg)
269         cl.append(fc)
270
271     with open("datafiles/allFirst50RegLatexTable.txt", "w+") as lout:
272         lout.write(tabulate({"class": fl, "classCount": cl},
273                             headers="keys", tablefmt="latex"))
274
275     clzzl = []
276     clzzcl = []
277     for clzz, cl in oToE.items():
278         print(clzz, len(cl))
279         clzzl.append(clzz)
280         clzzcl.append(len(cl))
281
282     with open("datafiles/allRegLatexTable.txt", "w+") as lout:
283         lout.write(tabulate({"class": clzzl, "classCount": clzzcl},
284                             headers="keys", tablefmt="latex"))
285
286 def do_classification(dbfile, startStop=(50, 100), extension=False)
287 :
288     # This is the original label classification method
289     # dbfile: the database file to be used
290     # startStop: are we doing the first 100 or all data, start train
291     # on count, stop is classification
292     # extension: are we doing using the extension for features
293     # parse the feed and create the classifier
294     feed = feedparser.parse("datafiles/f-measure.xml")
295     if extension:
296         cl = fisherclassifier(dbfile, getfeatures=entryfeatures)
297     else:
298         cl = fisherclassifier(dbfile)
299     # counter for training data
300     trainCount = 0
301     # for reporting
302     labs = ['forgotten song', 'concert/spotlight', 'lp review', '
303             the song remains the same']
304     actualLabels = []
305     predictedLabels = []
306     tabelOut = defaultdict(list)
307     trainedOut = defaultdict(list)
308
309     # I do not ask the user for a label as I let the structure of
310     # the blog be the user specified aspect
311     # ie Merle Haggard - "Mama Tried" (forgotten song): forgotten
312     # song is the label/category
313     for e in feed.entries:

```

```

309     print("_____")
310     # get the label
311     clazz = findClass.match(e.title).group(1).lower().strip()
312     # I combine concert and spotlight into a combined label
313     if "concert" in clazz or 'spotlight' in clazz:
314         clazz = 'concert/spotlight'
315     # get the data to be used depending on if we are using the
extension or not
316     text = "%s\n%s" % (e.title, BeautifulSoup(e.summary, "
html5lib").text) if not extension else e
317     # since train count starts at zero like any good cs person
we
318     if trainCount < startStop[0]:
319         # train and log what we trained
320         cl.train(text, clazz)
321         print("Training iteration %d for entry title: %s with
class %s" % (trainCount, e.title, clazz))
322         trainedOut['Title'].append(e.title)
323         trainedOut['Category'].append(clazz)
324         print("
_____")
325         trainCount += 1
326     else:
327         # now we classify
328         if startStop[0] <= trainCount < startStop[1]:
329             print("Classify iteration %d for entry title: %s
with class %s" % (trainCount, e.title, clazz))
330             # classify and store data for reporting
331             prediction = cl.classify(text, "indeterminable")
332             actualLabels.append(clazz)
333             predictedLabels.append(prediction)
334             cprob = cl.cprobLast
335             fprob = cl.fisherprob(text, clazz)
336             tlabelOut['title'].append(e.title)
337             tlabelOut['predicted_cat'].append(prediction)
338             tlabelOut['actual_cat'].append(clazz)
339             tlabelOut['acat_fprob'].append(fprob)
340             os = "Title: %s, predcat: %s, actcat: %s,
acat_cprob: %f, acat_fprob: %f" % (
341                 e.title, prediction, clazz, cprob, fprob)
342             print(os)
343             print("
_____")
344             trainCount += 1
345
346     # our file names are dependent on if we are using 100 entries
or all of them
347     if startStop[1] == 100:
348         ctable = "tables/regClassificationTable.txt" if not
extension else "tables/regClassificationTable-e.txt"
349         creport = "reports/regClassificationReport.txt" if not
extension else \
350             "reports/regClassificationReport-e.txt"
351         train = "tables/regTrainTable.txt" if not extension else "
tables/regTrainTable-e.txt"
352     else:
353         ctable = "tables/regClassificationTableAll.txt" if not

```

```

354     extension else \
355         "tables/regClassificationTableAll-e.txt"
356     creport = "reports/regClassificationReportAll.txt" if not
357     extension else \
358         "reports/regClassificationReportAll-e.txt"
359     train = "tables/regTrainTableAll.txt" if not extension else
360     "tables/regTrainTableAll-e.txt"
361
362 # write out information
363 with open(ctable, "w+") as tout:
364     tout.write(tabulate(tabelOut, headers="keys", tablefmt="
365     latex"))
366
367 with open(creport, "w+") as rout:
368     rout.write(classification_report(actualLabels,
369     predictedLabels, labels=labs, target_names=labs))
370
371 with open(train, "w+") as tout:
372     tout.write(tabulate(trainedOut, headers="keys", tablefmt="
373     latex"))
374 cl.close_con()
375
376 def myTest(dbfile, startStop=(50, 100), extension=False):
377     # works the same as the first one but we are classifying on
378     genre
379     with open("datafiles/artistsToGenre.json", "r") as agi:
380         aTg = json.load(agi)
381         feed = feedparser.parse("datafiles/f-measure.xml")
382         labs = ['Rock', 'R&B/Jazz/Mowtown/Country/Other', 'Metal/Punk/
383         Hardcore', 'Indie Rock/Alternative',
384         'Pop/Electronic/Hip-Hop']
385     if extension:
386         cl = fisherclassifier(dbfile, getfeatures=entryfeatures)
387     else:
388         cl = fisherclassifier(dbfile)
389         actualLabels = []
390         predictedLabels = []
391         tabelOut = defaultdict(list)
392         trainedOut = defaultdict(list)
393         trainCount = 0
394         for e in feed.entries:
395             m = artistsExtractor.match(e.title)
396             if m is not None:
397                 clazz = aTg[m.group(1).rstrip("-")]
398             else:
399                 clazz = "R&B/Jazz/Mowtown/Country/Other"
400
401             text = "%s\n%s" % (e.title, BeautifulSoup(e.summary, "
402             html5lib").text) if not extension else e
403             if trainCount < startStop[0]:
404                 cl.train(text, clazz)
405                 print("Training iteration %d for entry title: %s with
406                 class %s" % (trainCount, e.title, clazz))
407                 trainedOut['Title'].append(e.title)
408                 trainedOut['Category'].append(clazz)
409                 print("

```

```

401         trainCount += 1
402     else:
403         if startStop[0] <= trainCount < startStop[1]:
404             print("Classify iteration %d for entry title: %s
with class %s" % (trainCount, e.title, clazz))
405             prediction = cl.classify(text, "indeterminable")
406             actualLabels.append(clazz)
407             predictedLabels.append(prediction)
408             cprob = cl.cprobLast
409             fprob = cl.fisherprob(text, clazz)
410             tlabelOut['title'].append(e.title)
411             tlabelOut['predicted_cat'].append(prediction)
412             tlabelOut['actual_cat'].append(clazz)
413             tlabelOut['acat_fprob'].append(fprob)
414             os = "Title: %s, predcat: %s, actcat: %s,
acat_cprob: %f, acat_fprob: %f" % (
415                 e.title, prediction, clazz, cprob, fprob)
416             print(os)
417             print("
418         trainCount += 1
419
420     if startStop[1] == 100:
421         ctable = "tables/myTestClassificationTable.txt" if not
extension else \
422             "tables/myTestClassificationTable-e.txt"
423         creport = "reports/myTestClassificationReport.txt" if not
extension else \
424             "reports/myTestClassificationReport-e.txt"
425         train = "tables/myTestTrainTable.txt" if not extension else
"tables/myTestTrainTable-e.txt"
426     else:
427         ctable = "tables/myTestClassificationTableAll.txt" if not
extension else \
428             "tables/myTestClassificationTableAll-e.txt"
429         creport = "reports/myTestClassificationReportAll.txt" if
not extension else \
430             "reports/myTestClassificationReportAll-e.txt"
431         train = "tables/myTestTrainTableAll.txt" if not extension
else "tables/myTestTrainTableAll-e.txt"
432
433     with open(ctable, "w+") as tout:
434         tout.write(tabulate(tlabelOut, headers="keys", tablefmt="
latex"))
435
436     with open(creport, "w+") as rout:
437         rout.write(classification_report(actualLabels,
predictedLabels, labels=labs, target_names=labs))
438
439     with open(train, "w+") as tout:
440         tout.write(tabulate(trainedOut, headers="keys", tablefmt="
latex"))
441     cl.close_con()
442
443
444 def tf_train(cl, clmt, kf, aTg, feed, fname):

```



```

445 # ten fold cross validation method
446 # kfold metric scores for the blog structure classification
447 kfhMetrics = defaultdict(list)
448 # kfold metric scores for genre classification
449 kfhMetricsmt = defaultdict(list)
450 # output tables
451 tabelOut = defaultdict(list)
452 tabelOut2 = defaultdict(list)
453 count = 0
454 # loop through the validation indexes
455 for train_index, test_index in kf:
456     # labels for output for structure and genre predictions
457     actualLabels = []
458     predictedLabels = []
459     actualLabels2 = []
460     predictedLabels2 = []
461     print("Training for k=%d for %s" % (count + 1, fname))
462     # train
463     for i in train_index:
464         # var1 is for the structure and var2 is for genre
465         clazz1 = findClass.match(feed[i].title).group(1).lower
466         ().strip()
467         if "concert" in clazz1 or 'spotlight' in clazz1:
468             clazz1 = 'concert/spotlight'
469             text1 = "%s\n%s" % (feed[i].title, BeautifulSoup(feed[i
470 ].summary, "html5lib").text)
471             m = artistsExtractor.match(feed[i].title)
472             if m is not None:
473                 clazz2 = aTg[m.group(1).rstrip(" -")]
474             else:
475                 clazz2 = "R&B/Jazz/Mowtown/Country/Other"
476             text2 = "%s\n%s" % (feed[i].title, BeautifulSoup(feed
477 [1].summary, "html5lib").text)
478             cl.train(text1, clazz1)
479             clmt.train(text2, clazz2)
480             print("Classifying for k=%d for %s" % (count + 1, fname))
481             # classify
482             for i in test_index:
483                 clazz1 = findClass.match(feed[i].title).group(1).lower
484                 ().strip()
485                 if "concert" in clazz1 or 'spotlight' in clazz1:
486                     clazz1 = 'concert/spotlight'
487                     text1 = "%s\n%s" % (feed[i].title, BeautifulSoup(feed[i
488 ].summary, "html5lib").text)
489                     m = artistsExtractor.match(feed[i].title)
490                     if m is not None:
491                         clazz2 = aTg[m.group(1).rstrip(" -")]
492                     else:
493                         clazz2 = "R&B/Jazz/Mowtown/Country/Other"
494                     text2 = "%s\n%s" % (feed[i].title, BeautifulSoup(feed[i
495 ].summary, "html5lib").text)
496                     prediction = cl.classify(text1, "indeterminable")
497                     actualLabels.append(clazz1)
498                     predictedLabels.append(prediction)
499                     fprob = cl.fisherprob(text1, clazz1)
500
501                 tabelOut['title'].append(feed[i].title)

```

```

496         tabelOut[ 'predicted_cat' ].append(prediction)
497         tabelOut[ 'actual_cat' ].append(clazz1)
498         tabelOut[ 'acat_fprob' ].append(fprob)
499         prediction2 = clmt.classify(text2, "indeterminable")
500         actualLabels2.append(clazz2)
501         predictedLabels2.append(prediction2)
502         fprob2 = clmt.fisherprob(text2, clazz2)
503         tabelOut2[ 'title' ].append(feed[i].title)
504         tabelOut2[ 'predicted_cat' ].append(prediction2)
505         tabelOut2[ 'actual_cat' ].append(clazz2)
506         tabelOut2[ 'acat_fprob' ].append(fprob2)
507         # take mean of the values as the scikit learn scores
returns the values for each label
508         kfhMetrics[ 'precision' ].append(
509             statistics.mean(precision_score(actualLabels,
predictedLabels, average=None)))
510         kfhMetrics[ 'recall' ].append(statistics.mean(
recall_score(actualLabels, predictedLabels, average=None)))
511         kfhMetrics[ 'f1' ].append(statistics.mean(f1_score(
actualLabels, predictedLabels, average=None)))
512         kfhMetricsmt[ 'precision' ].append(
513             statistics.mean(precision_score(actualLabels2,
predictedLabels2, average=None)))
514         kfhMetricsmt[ 'recall' ].append(statistics.mean(
recall_score(actualLabels2, predictedLabels2, average=None)))
515         kfhMetricsmt[ 'f1' ].append(statistics.mean(f1_score(
actualLabels2, predictedLabels2, average=None)))
516         count += 1
517         # you gotta do it from scratch k times
518         cl.clear_db()
519         clmt.clear_db()
520     ret1 = []
521     ret2 = []
522     headers = [ "metric", "mean" ]
523     for m, v in kfhMetrics.items():
524         ret1.append([m, statistics.mean(v)])
525     for m, v in kfhMetricsmt.items():
526         ret2.append([m, statistics.mean(v)])
527
528     print(ret1)
529     print(ret2)
530
531     with open("tables/tenfold-classret-%s.txt" % fname, "w+") as
out:
532         out.write(tabulate(tabelOut, headers="keys", tablefmt="
latex"))
533     with open("tables/tenfold-classret-mytest-%s.txt" % fname, "w+")
as out:
534         out.write(tabulate(tabelOut2, headers="keys", tablefmt="
latex"))
535
536     with open("tables/tenfold-metrics-%s.txt" % fname, "w+") as out
:
537         out.write(tabulate(ret1, headers=headers, tablefmt="latex")
)
538     with open("tables/tenfold-metrics-mytest-%s.txt" % fname, "w+")
as out:

```

```

539         out.write(tabulate(ret2, headers=headers, tablefmt="latex")
540     )
541     cl.close_con()
542     clmt.close_con()
543
544     def ten_fold():
545         # do the ten fold validation
546         feed = feedparser.parse("datafiles/f-measure.xml")
547
548         with open("datafiles/artistsToGenre.json", "r") as agi:
549             aTg = json.load(agi)
550         # feall is for all data and fehundo is for the original test
551         feall = []
552         fehundo = []
553
554         count = 0
555         for e in feed.entries:
556             feall.append(e)
557             if count < 100:
558                 fehundo.append(e)
559                 count += 1
560
561         # create kfold index objects
562         # the take the length of the data and the number of folds
563         kfh = KFold(len(fehundo), 10)
564         kfa = KFold(len(feall), 10)
565         # create the classifiers
566         cl = fisherclassifier("dbs/fmeasure10f.sqlite3")
567         clmt = fisherclassifier("dbs/fmeasure-myTest10f.sqlite3")
568         cl2 = fisherclassifier("dbs/fmeasure10f-all.sqlite3")
569         clmt2 = fisherclassifier("dbs/fmeasure-myTest10f-all.sqlite3")
570         tf_train(cl, clmt, kfh, aTg, fehundo, "hundred")
571         tf_train(cl2, clmt2, kfa, aTg, feall, "all")
572
573
574     if __name__ == "__main__":
575         cwd = os.getcwd()
576         feedFile = "datafiles/f-measure.xml"
577         if not os.path.exists("%s/datafiles" % cwd):
578             os.makedirs("%s/datafiles" % cwd)
579         if not os.path.exists("%s/datafiles/f-measure.xml" % cwd):
580             getDataFeed()
581             labelCounter()
582
583         do_classification("dbs/fmeasure.sqlite3")
584         myTest("dbs/fmeasure-myTest.sqlite3")
585
586         do_classification("dbs/fmeasure-all.sqlite3", startStop=(98,
587             1000))
588         myTest("dbs/fmeasure-myTest-all.sqlite3", startStop=(98, 1000))
589
590         do_classification("dbs/fmeasure-extension.sqlite3", extension=
591             True)
592         myTest("dbs/fmeasure-myTest-extension.sqlite3", extension=True)
593
594         do_classification("dbs/fmeasure-all-extension.sqlite3",

```

```
593 startStop=(98, 1000), extension=True)
    myTest("dbs/fmeasure-myTest-all-extension.sqlite3", startStop
594         =(98, 1000), extension=True)
    ten_fold()
```

Listing 10: Code To Classify Blog Posts

```

1 import math
2 import re
3 from sqlite3 import dbapi2 as sqlite
4
5 import nltk
6 from bs4 import BeautifulSoup
7
8
9 def getwords(doc):
10     splitter = re.compile('\\W*')
11     # print doc
12     ## Remove all the HTML tags
13     doc = re.compile(r'<[^>+>').sub('', doc)
14     # Split the words by non-alpha characters
15     words = [s.lower().replace('\\', ' ') for s in nltk.
16               word_tokenize(doc)
17               if len(s) > 2 and len(s) < 20]
18
19     # Return the unique set of words only
20     return dict([(w, 1) for w in words])
21
22 def entryfeatures(entry):
23     splitter = re.compile('\\W*')
24     f = {}
25     titlewords = []
26     for s in nltk.word_tokenize(entry.title):
27         if 2 < len(s) < 20:
28             titlewords.append(s.lower().replace('\\', ' '))
29
30     for w in titlewords: f['Title:' + w] = 1
31     # Extract the summary words
32     summarywords = [s.lower().replace('\\', ' ') for s in
33                     nltk.word_tokenize(BeautifulSoup(entry.summary,
34                                                         "html5lib").text)
35                     if len(s) > 2 and len(s) < 20]
36
37     # Count uppercase words
38     uc = 0
39     for i in range(len(summarywords)):
40         w = summarywords[i]
41         f[w] = 1
42         if w.isupper(): uc += 1
43
44     # Get word pairs in summary as features
45     if i < len(summarywords) - 1:
46         twowords = ' '.join(summarywords[i:i + 1])
47         f[twowords] = 1
48
49     # Keep creator and publisher whole
50     # f['Publisher:' + entry['publisher']] = 1
51
52     # UPPERCASE is a virtual word flagging too much shouting
53     if float(uc) / len(summarywords) > 0.3: f['UPPERCASE'] = 1
54
55     return f

```

```

56
57 class classifier:
58     def __init__(self, dbfile, getfeatures=getwords):
59         self.fc = {}
60         self.cc = {}
61         self.getfeatures = getfeatures
62         # i moved the connection of the database here
63         # and we now have a set up
64         self.con = sqlite.connect(dbfile)
65         self.queries = {}
66         self._setupDB()
67
68     def _setupDB(self):
69         # build database and read in our queries
70         with open("datafiles/dbschema.sql", "r") as read:
71             self.con.executescript(read.read())
72         with open("datafiles/queries.txt", "r") as query:
73             for line in map(lambda x: x.rstrip("\n").split(':'),
74                             query):
75                 self.queries[line[0]] = line[1]
76
77     def manualClassdb(self, num, entry, feature, predicted, actual):
78         :
79         self.con.execute(self.queries['classEntry']
80                           % (num, entry, feature, predicted, actual,
81                             None))
82         self.con.commit()
83
84     def autoClassdb(self, num, entry, feature, predicted, actual,
85                     cp):
86         self.con.execute(self.queries['classEntry']
87                           % (num, entry, feature, predicted, actual,
88                             cp))
89         self.con.commit()
90
91     def incf(self, f, cat):
92         count = self.fcount(f, cat)
93         if count == 0:
94             self.con.execute(self.queries['insert_newFeature']
95                               % (f, cat))
96         else:
97             self.con.execute(
98                 self.queries['increment_feature']
99                 % (count + 1, f, cat))
100
101     def fcount(self, f, cat):
102         query = self.queries['count_Feature'] % (f, cat)
103         res = self.con.execute(query).fetchone()
104         if res is None:
105             return 0
106         else:
107             return int(res[0])
108
109     def incc(self, self, cat):

```

```

108         count = self.catcount(cat)
109         if count == 0:
110             self.con.execute(self.queries['insert_cat'] % cat)
111         else:
112             self.con.execute(self.queries['increment_cat'] % (count
+ 1, cat))
113
114
115
116     def catcount(self, cat):
117         res = self.con.execute(self.queries['count_cat'] % cat).
fetchone()
118         if res is None:
119             return 0
120         else:
121             return int(res[0])
122
123
124     def categories(self):
125         cur = self.con.execute(self.queries['get_cat'])
126         return [d[0] for d in cur]
127
128
129     def totalcount(self):
130         res = self.con.execute(self.queries['total_cat_count']).
fetchone()
131         if res is None: return 0
132         return res[0]
133
134     def train(self, item, cat):
135         features = self.getfeatures(item)
136         for f in features:
137             self.incf(f, cat)
138         self.incc(cat)
139         self.con.commit()
140
141     def fprob(self, f, cat):
142         if self.catcount(cat) == 0: return 0
143         return self.fcount(f, cat) / self.catcount(cat)
144
145     def weightedprob(self, f, cat, prf, weight=1.0, ap=0.5):
146         basicprob = prf(f, cat)
147
148         # Count the number of times this feature has appeared in
149         # all categories
150         totals = sum([self.fcount(f, c) for c in self.categories()
])
151
152         # Calculate the weighted average
153         bp = ((weight * ap) + (totals * basicprob)) / (weight +
totals)
154         return bp
155
156     def clear_db(self):
157         self.con.execute("DELETE FROM feature_count")
158         self.con.commit()
159         self.con.execute("DELETE FROM category_count")

```

```

160         self.con.commit()
161
162     def close_con(self):
163         self.con.commit()
164         self.con.close()
165
166
167 class naivebayes(classifier):
168     def __init__(self, dbfile, getfeatures=getwords):
169         classifier.__init__(self, dbfile, getfeatures)
170         self.thresholds = {}
171
172     def docprob(self, item, cat):
173         features = self.getfeatures(item)
174         p = 1
175         for f in features: p *= self.weightedprob(f, cat, self.
fprob)
176         return p
177
178     def prob(self, item, cat):
179         catprob = self.catcount(cat) / self.totalcount()
180         docprob = self.docprob(item, cat)
181         return docprob * catprob
182
183     def setthreshold(self, cat, t):
184         self.thresholds[cat] = t
185
186     def getthreshold(self, cat):
187         if cat not in self.thresholds: return 1.0
188         return self.thresholds[cat]
189
190     def classify(self, item, default=None):
191         probs = {}
192         max = 0.0
193         for cat in self.categories():
194             probs[cat] = self.prob(item, cat)
195             if probs[cat] > max:
196                 max = probs[cat]
197             best = cat
198
199         for cat in probs:
200             if cat == best: continue
201             if probs[cat] * self.getthreshold(best) > probs[best]:
202                 return default
203         return best
204
205 class fisherclassifier(classifier):
206     def __init__(self, dbfile, getfeatures=getwords):
207         classifier.__init__(self, dbfile, getfeatures)
208         self.minimums = {}
209         self.cprobLast = 0.0
210         self.fprobLast = 0.0
211
212     def cprob(self, f, cat):
213         # The frequency of this feature in this category
214         clf = self.fprob(f, cat)

```



```

215         if clf == 0: return 0
216
217     # The frequency of this feature in all the categories
218     freqsum = sum([self.fprob(f, c) for c in self.categories()
219 ])
220
221     # The probability is the frequency in this category divided
222     by
223     # the overall frequency
224     p = clf / freqsum
225
226     self.cprobLast = p
227
228     return p
229
230 def fisherprob(self, item, cat):
231     # Multiply all the probabilities together
232     p = 1
233     features = self.getfeatures(item)
234     for f in features:
235         p *= (self.weightedprob(f, cat, self.cprob))
236     # Take the natural log and multiply by -2
237     # I added this because we get a zero here in 10 fold
238     try:
239         fscore = -2 * math.log(p)
240     except ValueError:
241         fscore = -2 * math.log(self.fprobLast)
242     self.fprobLast = fscore
243     # Use the inverse chi2 function to get a probability
244     return self.invchi2(fscore, len(features) * 2)
245
246 def invchi2(self, chi, df):
247     m = chi / 2.0
248     sum = term = math.exp(-m)
249     for i in range(1, df // 2):
250         term *= m / i
251         sum += term
252     return min(sum, 1.0)
253
254 def setminimum(self, cat, min):
255     self.minimums[cat] = min
256
257 def getminimum(self, cat):
258     if cat not in self.minimums: return 0
259     return self.minimums[cat]
260
261 def classify(self, item, default=None):
262     # Loop through looking for the best result
263     best = default
264     max = 0.0
265     for c in self.categories():
266         p = self.fisherprob(item, c)
267         # Make sure it exceeds its minimum
268         if p > self.getminimum(c) and p > max:
269             best = c
270             max = p
271     return best

```

Listing 11: Python3 docclass file