Exploring Music Genres' Most Popular Words

This is the full execution of our final project for ENGR122 at Drexel University. This code was written entirely by Nick Widmann and Ebed Jarrell. It was last edited on 2/25/16.

Here we define the genres cell array. This includes strings representing the 5 different genres we will be exploring

```
genres = {'country', 'hiphop', 'pop', 'r&b', 'rock'};
```

The lyric_importer() function imports all of the song lyrics' words into a nested cell array called lyrics and one big cell array called all_words. lyrics is sorted by genre with lyrics{3}{1} corresponding to the first word taken from the file for genres{3} i.e. 'pop'. The file name and location for GENRE is assumed to be 'lyrics/lyrics_GENRE.dat'. This file holds all of the lyrics for that genre. These were all taken from genius.com using a webscraper written in python using robobrowser.

```
[lyrics, all_words] = lyric_importer(genres);
```

The function create_unique_words_list() takes in all_words and sorts through to create a single list where each word only occurs once

```
unique_words = create_unique_words_list(all_words);
```

The word_percent_by_genre() function creates a matrix percent_mat that holds the percentages of each word in unique_words in each of genres. e.g. The percentage of unique_words{500} in genre{4} can be found at percent_mat(4,500). This percentage is given as a decimal with 0.5 = 50%.

```
percent_mat = word_percent_by_genre(unique_words, lyrics, genres);
```

This script, called full_execution_demo.m, executes all of the functions used in this project. However, this is not the ideal situation because getting to this point takes 33 minutes of execution. This is because we needed to sort through the words of over 10 thousand songs. At this point, we saved our workspace to a file called data.mat that we could load each time we opened MATLAB. For demonstration purposes, we have included everything in this one file.

The next four lines of code attempt to categorize an unknown song into a genre based on how its word percentages compare to percent_mat. The unknown songs are all saved at location 'unknown_songs/SONG_FILE.dat'. For this example, we will use 'song4' which is "Rude Boy" by Rhianna.

```
filename = 'song4';
```

This function called read_song_from_dat() takes in the file name (e.g. 'song4') and returns the song's words as a cell array called song_word_list. It converts all the words to uppercase and removes any characters that are not letters to simplify categorization.

```
song_word_list = read_song_from_dat(filename);
```

The function filter_song() filters out the n most popular words in the English language from song_word_list. For some songs, a majority of the words could be "THE", "AND", or other popular words that aren't characteristic of a genre. For those songs, to correctly categorize a song, it is necessary to filter those words out. In this example, filtering is not necessary so n=0 and this function is essentially unused.

```
song_word_list = filter_song(song_word_list, 0);
```

The function find_genre() first creates a vector the same size as unique_words full of zeros and then fills in each entry i with the percentage of unique_words{i} in song_word_list. It then treats each row in percent_mat as a vector representing the percentages of unique_words in a genre. Finally, it compares those vectors to the percentage vector for song_word_list and finds the geometric distance between each genre and the unknown song. The closest genre to the song is the one with the most similar word makeup and theoretically should be the unknown song's genre.

```
genre = find_genre(song_word_list, unique_words, percent_mat, genres)
```

```
Genre: country   Distance: 0.196478
Genre: hiphop    Distance: 0.194604
Genre: pop       Distance: 0.190323
Genre: r&b       Distance: 0.185676
Genre: rock      Distance: 0.193496

genre =

    'r&b'
```

We also created a GUI that allows a user to enter a word and compare it's percentages across genres. For demonstration purposes, we have programmatically entered a word ('baby') to be compared. In actual usage though, the user would be able to type a word into the text entry box and then press enter to see a bar graph comparing its percentage in each genre.

```
compare_word_gui_demo(unique_words, genres, percent_mat)
```

Enter a word here to compare its percentage across genres

baby