Looking for Similar Artists by Analyzing Lyrics

One of my favorite music groups is Kings of Convenience. They are categorized as indie/pop musicians. In this Notebook I will be getting the lyrics of ten of their songs as well as 10 songs from 9 other musicians that are also considered indie/pop.

The focus of this notebook will be to try to identify which other musicians (from the ones chosen) are most similar to Kings of Convenience based on their lyrics. This will help me decide which other artists I might also be interested in hearing. This notebook will also allow others to figure out which artists are similar to their own favorite musicians.

First you will need to install Genius library and create a Token to access lyrics from Genius.com

To create the token go to https://genius.com/developers and choose the create an api client box. If you are having trouble installing the library, this site might help you https://pypi.org/project/lyricsgenius/.

In [1]:
```
from lyricsgenius import Genius
import pandas as pd
import os
import re
import string


## inside the token string copy the token that was generated. You might want to erase it afterwards so
 that nobody else can access it.
token=''
```

Let's see how this works

We will start by trying to get 2 songs by Kings of Convenience to make sure this library is working the way we expect it to. I will be displaying the lyrics of the first song.

In [2]:
```
genius = Genius(token)
artist = genius.search_artist("Kings of Convenience", max_songs=2, sort="title")
print(artist.songs[0])
```
```
Searching for songs by Kings of Convenience...

Song 1: "24-25"
Song 2: "Angel"

Reached user-specified song limit (2).
Done. Found 2 songs.
"24-25" by Kings of Convenience:
    She'll be gone soon
    You can have me for yourself
    She'll be gone soon
    You can have me for yourself

    R...
```

Now let's create a function that will take as its input an artist's name and a list of their songs and will output a dictionary that contains their lyrics (which will have been cleaned so that they are in a format we can analyze)

I will test the function with kings of convenience and a list of their most popular songs. I also want to add that they way I obtained the 10 songs for each artist by looking for them in Spotify and then going to their 'This is (Name of Artist)' playlist and from there choosing the first 10 songs (if there was an error with the lyrics of a song, I just switched with another song from that playlist). Also, I used this site as a reference to find other artists considered indie/pop https://www.last.fm/tag/indie+pop/artists.

In [3]:
```
## Function works: First it creates an empty dictionary. Then for each song in the list of songs it
 gets its lyrics and cleans them up. After that it adds the lyrics to the empty dictionary: Using the n
ame of the song as the key and the lyrics as the value.


def artist_lyrics(artist, songs):
    d = {}
    for song in songs:
        s = genius.search_song(song, artist)
        lyrics = s.lyrics
        lyrics = lyrics.lower() ##makes lyrics lowercase
        lyrics = re.sub(r'[\(\[].*?[\)\]]', '', lyrics) ##removes anything in brackets (such as (Choru
s))
        lyrics = re.sub('[%s]' % re.escape(string.punctuation), '', lyrics) ##removes punctuation
        lyrics = os.linesep.join([s for s in lyrics.splitlines() if s]) ##removes blank lines
        d[song] = lyrics
    return d

songs = ['Misread', 'Rocky Trail', 'Mrs. Cold', 'Comb My Hair', "I'd Rather Dance With You", 'Boat Beh
ind', 'Rumours', 'Know How', 'Toxic Girl', '24-25']
artist = "Kings of Convenience"
koc_d = artist_lyrics(artist, songs)
koc_d['Misread']
```
```
Searching for "Misread" by Kings of Convenience...
Done.
Searching for "Rocky Trail" by Kings of Convenience...
Done.
Searching for "Mrs. Cold" by Kings of Convenience...
Done.
Searching for "Comb My Hair" by Kings of Convenience...
Done.
Searching for "I'd Rather Dance With You" by Kings of Convenience...
Done.
Searching for "Boat Behind" by Kings of Convenience...
Done.
Searching for "Rumours" by Kings of Convenience...
Done.
Searching for "Know How" by Kings of Convenience...
Done.
Searching for "Toxic Girl" by Kings of Convenience...
Done.
Searching for "24-25" by Kings of Convenience...
Done.
```
Out[3]:
```
'if you want to be my friend\nyou want us to get along\nplease do not expect me to\nwrap it up and ke
ep it there\nthe observation in doing could\neasily be understood\nas cynical demeanor\nbut one of us
 misread\nwhat do you know\nit happened again\na friend is not a means\nyou utilize to get somewhere\n
somehow i didnt notice\nfriendship is an end\nwhat do you know\nit happened again\nhow come no one to
ld me\nall throughout history\nthe loneliest people\nwere the ones who always spoke the truth\nthe on
es who made a difference\nby withstanding the indifference\ni guess its up to me now\nshould i take t
hat risk or just smile\nwhat do you know\nit happened again\nwhat do you know4embedshare urlcopyembed
copy'
```

In the cells below I will be getting 10 popular songs from the other 9 artists

If you are using different artists or songs, I advice you to display the full dictionary for each artist as there are some times you will get something different than the lyrics (instead of the lyrics you will get a really long block of text, the length of the text is usually the hint, so if you see a long block of text double check and compare it with the real lyrics).

In [4]:
```
## Belle and Sebastian
songs = ['The Boy With The Arab Strap', 'Another Sunny Day', 'Piazza', 'New York Catcher', 'The Party L
ine', "Get Me Away From Here, I'm Dying", 'Lazy Line Painter Jane', "Your Cover's Blown", 'Poor Boy',
"There's Too Much Love"]
artist = "Belle and Sebastian"
belle_and_s =  artist_lyrics(artist, songs)
```
```
Searching for "The Boy With The Arab Strap" by Belle and Sebastian...
Done.
Searching for "Another Sunny Day" by Belle and Sebastian...
Done.
Searching for "Piazza" by Belle and Sebastian...
Done.
Searching for "New York Catcher" by Belle and Sebastian...
Done.
Searching for "The Party Line" by Belle and Sebastian...
Done.
Searching for "Get Me Away From Here, I'm Dying" by Belle and Sebastian...
Done.
Searching for "Lazy Line Painter Jane" by Belle and Sebastian...
Done.
Searching for "Your Cover's Blown" by Belle and Sebastian...
Done.
Searching for "Poor Boy" by Belle and Sebastian...
Done.
Searching for "There's Too Much Love" by Belle and Sebastian...
Done.
```

In [5]:
```
## The Shins
songs = ['Girl Inform Me', 'New Slang', 'Name For You', 'Phantom Limb', 'Australia', 'Pink Bullets', 'S
imple Song', 'Painting a Hole', 'Caring is Creepy', 'Kissing the Lipless']
artist = "The Shins"
the_shins =  artist_lyrics(artist, songs)
```
```
Searching for "Girl Inform Me" by The Shins...
Done.
Searching for "New Slang" by The Shins...
Done.
Searching for "Name For You" by The Shins...
Done.
Searching for "Phantom Limb" by The Shins...
Done.
Searching for "Australia" by The Shins...
Done.
Searching for "Pink Bullets" by The Shins...
Done.
Searching for "Simple Song" by The Shins...
Done.
Searching for "Painting a Hole" by The Shins...
Done.
Searching for "Caring is Creepy" by The Shins...
Done.
Searching for "Kissing the Lipless" by The Shins...
Done.
```

In [6]:
```
## Feist
songs = ['1234', 'Fire In The Water', 'Mushaboom', 'The Limit To Your Love', 'How Come You Never Go The
re', 'One Evening', 'I Feel It All', 'Train Song', "I'm Not Running Away", 'Love Is A Lonely Thing']
artist = "Feist"
feist =  artist_lyrics(artist, songs)
```
```
Searching for "1234" by Feist...
Done.
Searching for "Fire In The Water" by Feist...
Done.
Searching for "Mushaboom" by Feist...
Done.
Searching for "The Limit To Your Love" by Feist...
Done.
Searching for "How Come You Never Go There" by Feist...
Done.
Searching for "One Evening" by Feist...
Done.
Searching for "I Feel It All" by Feist...
Done.
Searching for "Train Song" by Feist...
Done.
Searching for "I'm Not Running Away" by Feist...
Done.
Searching for "Love Is A Lonely Thing" by Feist...
Done.
```

In [7]:
```
## The Decemberists
songs = ['We All Die Young', "Don't Carry It All", 'Severed', 'The Crane Wife 3', 'Make You Better', 'O
nce In My Life', 'The Sporting Life', "I'll Be Your Girl", 'Red Right Ankle', 'O Valencia!']
artist = "The Decemberists"
decemberists =  artist_lyrics(artist, songs)
```
```
Searching for "We All Die Young" by The Decemberists...
Done.
Searching for "Don't Carry It All" by The Decemberists...
Done.
Searching for "Severed" by The Decemberists...
Done.
Searching for "The Crane Wife 3" by The Decemberists...
Done.
Searching for "Make You Better" by The Decemberists...
Done.
Searching for "Once In My Life" by The Decemberists...
Done.
Searching for "The Sporting Life" by The Decemberists...
Done.
Searching for "I'll Be Your Girl" by The Decemberists...
Done.
Searching for "Red Right Ankle" by The Decemberists...
Done.
Searching for "O Valencia!" by The Decemberists...
Done.
```

In [8]:
```
## MGMT
songs = ['Time to Pretend', 'Kids', 'Electric Feel', 'When You Die', 'Little Dark Age', 'Me and Michae
l', 'Congratulations', 'Weekend Wars', 'In The Afternoon', 'The Youth']
artist = "MGMT"
mgmt =  artist_lyrics(artist, songs)
```
```
Searching for "Time to Pretend" by MGMT...
Done.
Searching for "Kids" by MGMT...
Done.
Searching for "Electric Feel" by MGMT...
Done.
Searching for "When You Die" by MGMT...
Done.
Searching for "Little Dark Age" by MGMT...
Done.
Searching for "Me and Michael" by MGMT...
Done.
Searching for "Congratulations" by MGMT...
Done.
Searching for "Weekend Wars" by MGMT...
Done.
Searching for "In The Afternoon" by MGMT...
Done.
Searching for "The Youth" by MGMT...
Done.
```

In [9]:
```
## Vampire Weekend
songs = ['This Life', 'Jerusalem, New York, Berlin', 'Harmony Hall', 'Flower Moon', 'Married in a Gold
 Rush', 'Unbearably White', 'Bambina', 'Sunflower', 'Big Blue', 'Hold You Now']
artist = "Vampire Weekend"
vampire_weekend =  artist_lyrics(artist, songs)
```
```
Searching for "This Life" by Vampire Weekend...
Done.
Searching for "Jerusalem, New York, Berlin" by Vampire Weekend...
Done.
Searching for "Harmony Hall" by Vampire Weekend...
Done.
Searching for "Flower Moon" by Vampire Weekend...
Done.
Searching for "Married in a Gold Rush" by Vampire Weekend...
Done.
Searching for "Unbearably White" by Vampire Weekend...
Done.
Searching for "Bambina" by Vampire Weekend...
Done.
Searching for "Sunflower" by Vampire Weekend...
Done.
Searching for "Big Blue" by Vampire Weekend...
Done.
Searching for "Hold You Now" by Vampire Weekend...
Done.
```

In [14]:
```
## of Montreal
songs = ["The Party's Crushing Us", 'Gronlandic Edit', 'Gallery Piece', 'Lysergic Bliss', "It's differe
nt for girls", "Bunny Ain't No Kind Of Rider", 'We Will Commit Wolf Murder', 'Aries Equals Good Trash',
 "Tim I Wish You Were Born a Girl", 'The Past Is A Grotesque Animal']
artist = "of Montreal"
of_montreal =  artist_lyrics(artist, songs)
```
```
Searching for "The Party's Crushing Us" by of Montreal...
Done.
Searching for "Gronlandic Edit" by of Montreal...
Done.
Searching for "Gallery Piece" by of Montreal...
Done.
Searching for "Lysergic Bliss" by of Montreal...
Done.
Searching for "it's different for girls" by of Montreal...
Done.
Searching for "Bunny Ain't No Kind Of Rider" by of Montreal...
Done.
Searching for "We Will Commit Wolf Murder" by of Montreal...
Done.
Searching for "Aries Equals Good Trash" by of Montreal...
Done.
Searching for "Tim I Wish You Were Born a Girl" by of Montreal...
Done.
Searching for "The Past Is A Grotesque Animal" by of Montreal...
Done.
```

In [11]:
```
## Tegan and Sara
songs = ['Closer', 'Where Does the Good Go', 'Back in Your Head', 'Boyfriend', 'Walking with a Ghost',
'I Was a Fool', 'The Con', 'Bad Ones', "You Wouldn't Like Me", 'Goodbye, Goodbye']
artist = "Tegan and Sara"
tegan_and_sara =  artist_lyrics(artist, songs)
```
```
Searching for "Closer" by Tegan and Sara...
Done.
Searching for "Where Does the Good Go" by Tegan and Sara...
Done.
Searching for "Back in Your Head" by Tegan and Sara...
Done.
Searching for "Boyfriend" by Tegan and Sara...
Done.
Searching for "Walking with a Ghost" by Tegan and Sara...
Done.
Searching for "I Was a Fool" by Tegan and Sara...
Done.
Searching for "The Con" by Tegan and Sara...
Done.
Searching for "Bad Ones" by Tegan and Sara...
Done.
Searching for "You Wouldn't Like Me" by Tegan and Sara...
Done.
Searching for "Goodbye, Goodbye" by Tegan and Sara...
Done.
```

In [12]:
```
## Lana Del Rey
songs = ["Blue Banisters", 'Wildflower Wildfire', 'Text Book', 'Summertime Sadness', 'Born To Die', 'Vi
deo Games', 'Chemtrail Over The Country Club', 'Young and Beautiful', 'Blue Jeans', 'Ride']
artist = "Lana Del Rey"
lana_dr =  artist_lyrics(artist, songs)
```
```
Searching for "Blue Banisters" by Lana Del Rey...
Done.
Searching for "Wildflower Wildfire" by Lana Del Rey...
Done.
Searching for "Text Book" by Lana Del Rey...
Done.
Searching for "Summertime Sadness" by Lana Del Rey...
Done.
Searching for "Born To Die" by Lana Del Rey...
Done.
Searching for "Video Games" by Lana Del Rey...
Done.
Searching for "Chemtrail Over The Country Club" by Lana Del Rey...
Done.
Searching for "Young and Beautiful" by Lana Del Rey...
Done.
Searching for "Blue Jeans" by Lana Del Rey...
Done.
Searching for "Ride" by Lana Del Rey...
Done.
```

Now that we have a dictionary for each of the artists, we will be creating a pandas data frame

I work frequently with pandas so this is the easiest way for me to work with data. The dataframe will display the song, the lyrics, and the name of the artist. I will be writing a function to create this dataframe.

In [15]:
```
##This function takes as an input a list of the artist's dictionary that we just created and a list of
 the names of the artists that is in the same sequence as the other list (that is to say the name of the
 artist is in the same position as its corresponding dictionary). First the function creates a data fra
me with the first dictionary, it names its lyrics column and then creates a new column with the name of
 the artist. Then for each other dictionary it does the same and concatenates it with the  first data f
rame.

def create_df(d_lst, artists):
    df = pd.DataFrame.from_dict(d_lst[0], orient="index")
    df.columns = ['lyrics']
    df['artist'] = artists[0]
    for i in range(1,len(d_lst)):
        df_2 = pd.DataFrame.from_dict(d_lst[i], orient="index")
        df_2.columns = ['lyrics']
        df_2['artist'] = artists[i]
        df = pd.concat([df, df_2])

    return df

lst = [koc_d, belle_and_s, the_shins, feist, decemberists, mgmt, vampire_weekend, of_montreal, tegan_an
d_sara, lana_dr]
artists = ['Kings of Convenience', 'Belle and Sebastian', 'The Shins', 'Feist', 'The Decemberists', 'MG
MT', 'Vampire Weekend', 'of Montreal', 'Tegan and Sara', 'Lana Del Rey']

df = create_df(lst, artists)
df
```
Out[15]:

	lyrics	artist
Misread	if you want to be my friend\nyou want us to ge...	Kings of Convenience
Rocky Trail	one more time\nlets say you give me one more t...	Kings of Convenience
Mrs. Cold	hey baby mrs cold\nacting so tough\ndidnt know...	Kings of Convenience
Comb My Hair	ive got nothing to say to you\nive got to look...	Kings of Convenience
I'd Rather Dance With You	id rather dance with you than talk with you\ns...	Kings of Convenience
...	...	...
Video Games	swingin in the backyard pull up in your fast c...	Lana Del Rey
Chemtrail Over The Country Club	im on the run with you my sweet love\ntheres n...	Lana Del Rey
Young and Beautiful	ive seen the world done it all\nhad my cake no...	Lana Del Rey
Blue Jeans	blue jeans white shirt\nwalked into the room y...	Lana Del Rey
Ride	oohoohoohooh oohoohooh\noohoohoohooh oohoohooh...	Lana Del Rey

100 rows × 2 columns

Now that we have the data frame we can start doing some exploration of the data

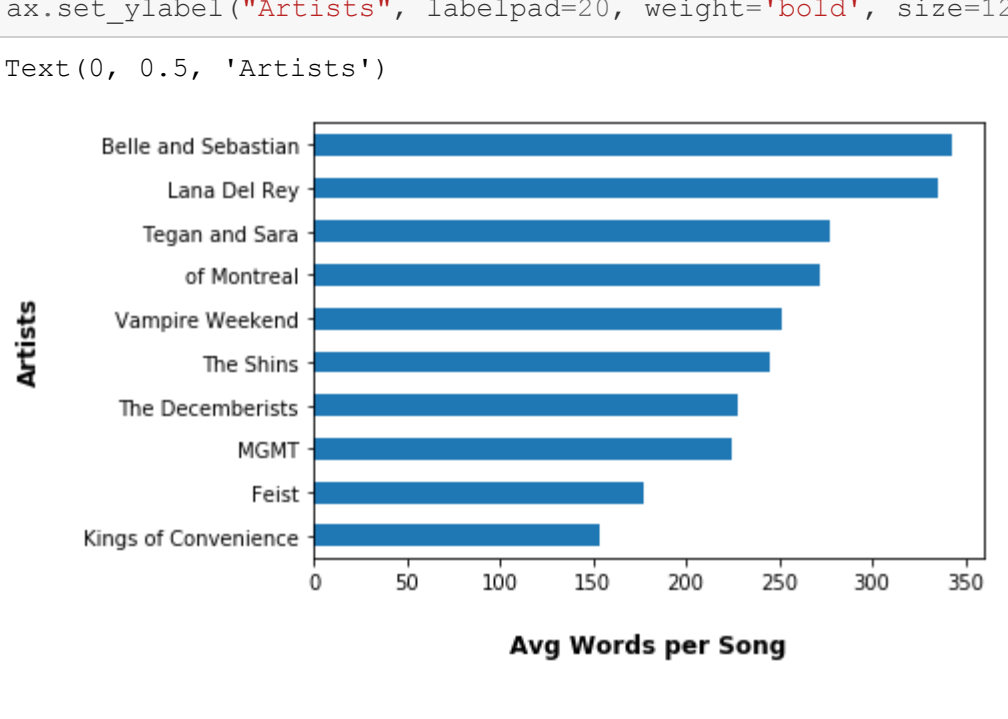
First let's see how many words each of their songs contain. I know most of the songs by Kings of Convenience contain less lyrics than most of the other songs I know of, this might give us an idea which of the other artists is similar to Kings of Convenience in this way.

In [16]:
```
## This function splits the lyrics into a list with the different words in the lyrics it then returns t
he length of the list.

def length_song(lyrics):
    x = [word for word in lyrics.split()]
    return len(x)

df['number of words'] = df['lyrics'].apply(lambda x: length_song(x))
df
```
Out[16]:

	lyrics	artist	number of words
Misread	if you want to be my friend\nyou want us to ge...	Kings of Convenience	132
Rocky Trail	one more time\nlets say you give me one more t...	Kings of Convenience	262
Mrs. Cold	hey baby mrs cold\nacting so tough\ndidnt know...	Kings of Convenience	177
Comb My Hair	ive got nothing to say to you\nive got to look...	Kings of Convenience	103
I'd Rather Dance With You	id rather dance with you than talk with you\ns...	Kings of Convenience	163
...	...	...	...
Video Games	swingin in the backyard pull up in your fast c...	Lana Del Rey	359
Chemtrail Over The Country Club	im on the run with you my sweet love\ntheres n...	Lana Del Rey	301
Young and Beautiful	ive seen the world done it all\nhad my cake no...	Lana Del Rey	306
Blue Jeans	blue jeans white shirt\nwalked into the room y...	Lana Del Rey	384
Ride	oohoohoohooh oohoohooh\noohoohoohooh oohoohooh...	Lana Del Rey	274

100 rows × 3 columns

In [60]:
```
import matplotlib as plt

ax = df.groupby('artist')['number of words'].mean().sort_values().plot.barh()
ax.set_xlabel("Avg Words per Song", labelpad=20, weight='bold', size=12)
ax.set_ylabel("Artists", labelpad=20, weight='bold', size=12)
```
Out[60]: `Text(0, 0.5, 'Artists')`



From the graph above we can see that Feist is very similar to Kings of Convenience(KOC) as they tend to not use many lyrics (usually around 150 words per song). Interestingly, KOC has multiple songs in which they feature Feist.

Now let's see how polarizing and subjective each song is and get a visualization for that.

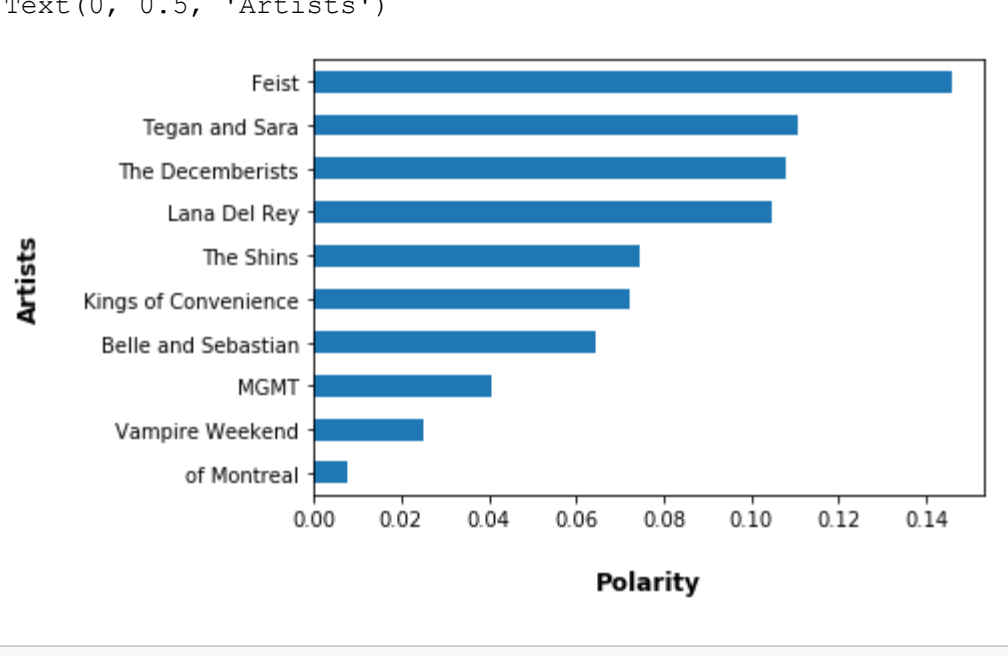
To get the sentiment for each song I will be using Textblob. For more info on how this algorithm works please see the following link https://textblob.readthedocs.io/en/dev/.

In [18]:
```
from textblob import TextBlob

pol = lambda x: TextBlob(x).sentiment.polarity
sub = lambda x: TextBlob(x).sentiment.subjectivity

df['polarity'] = df['lyrics'].apply(pol)
df['subjectivity'] = df['lyrics'].apply(sub)

df
```
Out[18]:

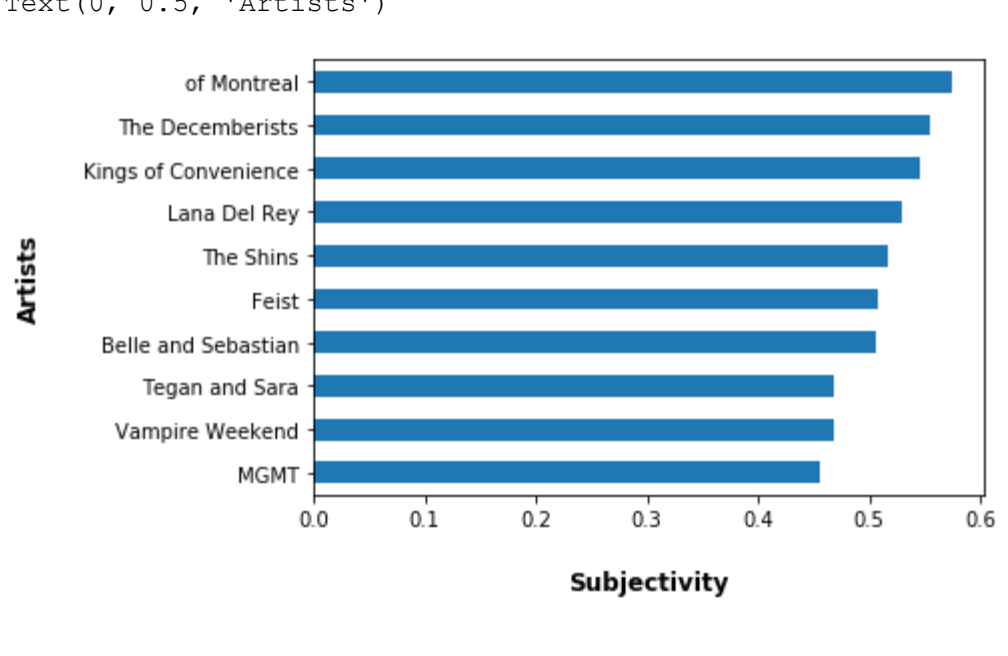
	lyrics	artist	number of words	polarity	subjectivity
Misread	if you want to be my friend\nyou want us to ge...	Kings of Convenience	132	0.044444	0.644444
Rocky Trail	one more time\nlets say you give me one more t...	Kings of Convenience	262	0.268833	0.570333
Mrs. Cold	hey baby mrs cold\nacting so tough\ndidnt know...	Kings of Convenience	177	0.050529	0.511111
Comb My Hair	ive got nothing to say to you\nive got to look...	Kings of Convenience	103	0.145833	0.490278
I'd Rather Dance With You	id rather dance with you than talk with you\ns...	Kings of Convenience	163	0.070357	0.428929
...	...	...	...	...	...
Video Games	swingin in the backyard pull up in your fast c...	Lana Del Rey	359	0.105882	0.541176
Chemtrail Over The Country Club	im on the run with you my sweet love\ntheres n...	Lana Del Rey	301	0.059592	0.545616
Young and Beautiful	ive seen the world done it all\nhad my cake no...	Lana Del Rey	306	0.447806	0.712226
Blue Jeans	blue jeans white shirt\nwalked into the room y...	Lana Del Rey	384	0.260426	0.532264
Ride	oohoohoohooh oohoohooh\noohoohoohooh oohoohooh...	Lana Del Rey	274	-0.098948	0.524544

100 rows × 5 columns

In [63]:
```
##Polarity graph
x = df.groupby('artist')['polarity'].mean().sort_values()
ax = x.plot(kind='barh')
ax.set_xlabel("Polarity", labelpad=20, weight='bold', size=12)
ax.set_ylabel("Artists", labelpad=20, weight='bold', size=12)
```
Out[63]: `Text(0, 0.5, 'Artists')`



In [59]:
```
## Subjectivity graph
ax = df.groupby('artist')['subjectivity'].mean().sort_values().plot.barh()
ax.set_xlabel("Subjectivity", labelpad=20, weight='bold', size=12)
ax.set_ylabel("Artists", labelpad=20, weight='bold', size=12)
```
Out[59]: `Text(0, 0.5, 'Artists')`



From the polarity and subectivity graph there isn't much insight other than most of the artist tend be either neutral or a little positive in their songs (polarity ratings are within difference of .1, so very similar for all artists). They also tend to be subjective (they all have a similar rating: around 0.5) which makes sense because most songs of their songs tend to be about subjective experiences.

Let's do a word cloud with the frequenly used words from each artist

To do this we will split the lyrics into different words but remove what are considered stopwords (to know more about stopwords: https://www.meaningcloud.com/developer/resources/doc/models/1.0/models/stopwords, just note that stopwords may differ from the ones used with the library I will be using) and also words that I added to that list because I felt that they didn't provide any insight.

First let's create a new column called words in our dataframe that splits the lyrics of each song into a list of words.

We will not be including stopwords.

In [21]:
```
from nltk.corpus import stopwords

addsw = ['dont', 'let', 'could', 'never', 'oh', 'oooh', 'got', 'like', 'youre', 'well', 'la', 'im', 'ooh
oohoohooh', 'urlcopyembedcopy', 'ive', 'whats', 'theyve', 'ohohh', 'youve', 'cant', 'wanna', 'another',
 'theres', 'know', 'one', 'want', 'good', 'get','ill']
stopwords = stopwords.words('english')

for word in addsw:
    stopwords.append(word)


def split_song(lyrics):
    x = [word for word in lyrics.split() if word not in stopwords]
    return x

df['words'] = df['lyrics'].apply(lambda x: split_song(x))
df
```
Out[21]:

	lyrics	artist	number of words	polarity	subjectivity	words
Misread	if you want to be my friend\nyou want us to ge...	Kings of Convenience	132	0.044444	0.644444	[friend, us, along, please, expect, wrap, keep...
Rocky Trail	one more time\nlets say you give me one more t...	Kings of Convenience	262	0.268833	0.570333	[time, lets, say, give, time, last, chance, sp...
Mrs. Cold	hey baby mrs cold\nacting so tough\ndidnt know...	Kings of Convenience	177	0.050529	0.511111	[hey, baby, mrs, cold, acting, tough, didnt, h...
Comb My Hair	ive got nothing to say to you\nive got to look...	Kings of Convenience	103	0.145833	0.490278	[nothing, say, look, ahead, love, felt, dead, ...
I'd Rather Dance With You	id rather dance with you than talk with you\ns...	Kings of Convenience	163	0.070357	0.428929	[id, rather, dance, talk, move, room, space, u...
...	...	...	...	...	...	...
Video Games	swingin in the backyard pull up in your fast c...	Lana Del Rey	359	0.105882	0.541176	[swingin, backyard, pull, fast, car, whistlin,...
Chemtrail Over The Country Club	im on the run with you my sweet love\ntheres n...	Lana Del Rey	301	0.059592	0.545616	[run, sweet, love, nothing, wrong, contemplati...
Young and Beautiful	ive seen the world done it all\nhad my cake no...	Lana Del Rey	306	0.447806	0.712226	[seen, world, done, cake, diamonds, brilliant,...
Blue Jeans	blue jeans white shirt\nwalked into the room y...	Lana Del Rey	384	0.260426	0.532264	[blue, jeans, white, shirt, walked, room, made...
Ride	oohoohoohooh oohoohooh\noohoohoohooh oohoohooh...	Lana Del Rey	274	-0.098948	0.524544	[oohoohooh, oohoohooh, open, road, fulltime, d...

100 rows × 6 columns

After creating the words column we can now groupby each artist and append the word list for all of their songs so that we have a data frame where we have the artist in one column and a list of all of the words (excluding stopwords) that their songs contain.

We will call the new data frame word_lyrics.

In [22]:
```
word_lyrics = df.groupby('artist').agg({'words': 'sum'})
word_lyrics
```
Out[22]:

artist	words
Belle and Sebastian	[mile, half, bus, takes, long, time, odour, ol...
Feist	[two, three, four, tell, love, sleepless, long,...
Kings of Convenience	[friend, us, along, please, expect, wrap, keep...
Lana Del Rey	[picture, wall, john, deere, jenny, handed, be...
MGMT	[feeling, rough, feeling, raw, prime, life, le...
Tegan and Sara	[little, bit, closer, come, little, closer, co...
The Decemberists	[late, night, counting, stitches, laying, side...
The Shins	[girl, inform, senses, warn, clever, eyes, eas...
Vampire Weekend	[baby, pain, natural, rain, thought, didnt, ra...
of Montreal	[mystery, stand, stare, nibble, ear, smell, oc...

Now let's create another column called word freq that tranforms the words column into a dictionary that has each word as key and its frequency as its value

To do this we will be using Counter from the collections library as it creates this dictionary (with words and their frequencies). Then we will be filtering this dictionary so that it only displays words that are used at least 5 times and have the function return the filtered dictionary in a sorted way, so that higher values are displayed first.

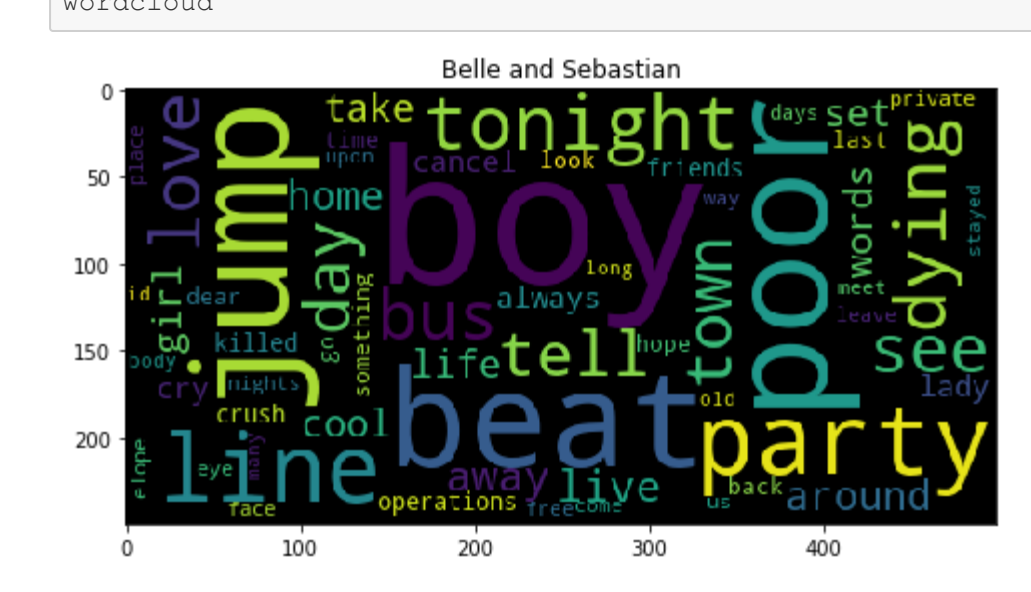
In [23]:
```
## Function takes as input a list of words, it creates a dictionary with words as key and frequency as
 values. Then it filters the dictionary so that only words that have a frequency of 5 or greater remain
 and then returns that dictionary sorted showing higher values first.

from collections import Counter

def word_counter(lst):
    c = Counter(lst)
    d = {}
    for (key, value) in c.items():
        if value > 4:
            d[key] = value
    return {k: v for k, v in sorted(d.items(), key=lambda item: item[1], reverse=True)}

word_lyrics['word counter'] = word_lyrics['words'].apply(lambda x: word_counter(x))
word_lyrics
```
Out[23]:

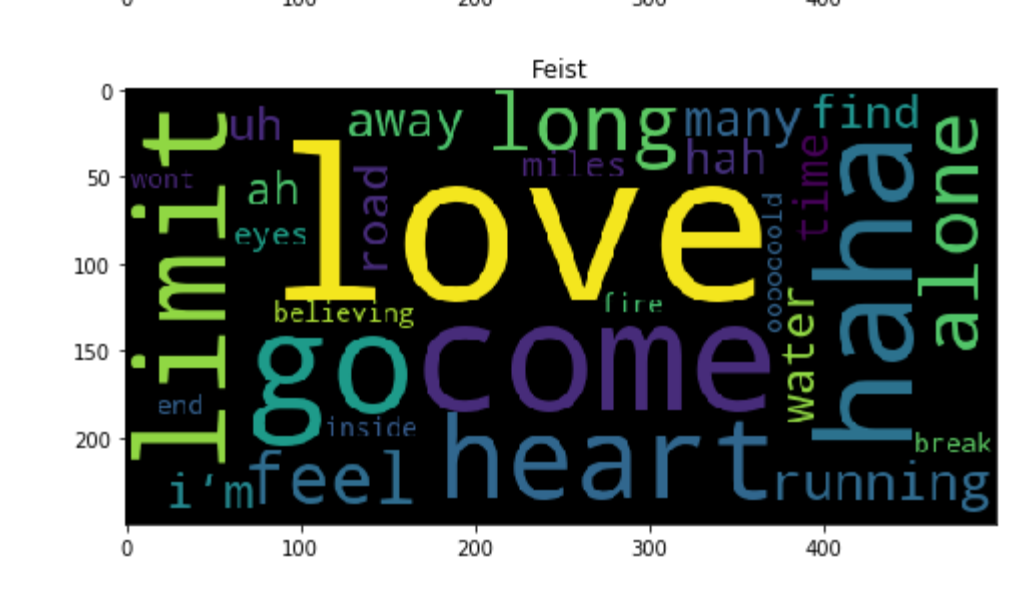
artist	words	word counter
Belle and Sebastian	[mile, half, bus, takes, long, time, odour, ol...	{'boy': 35, 'poor': 17, 'beat': 16, 'jump': 15,...
Feist	[two, three, four, tell, love, sleepless, long,...	{'love': 42, 'come': 19, 'go': 16, 'haha': 14,...
Kings of Convenience	[friend, us, along, please, expect, wrap, keep...	{'tell': 19, 'around': 8, 'okay': 8, 'belong':...
Lana Del Rey	[picture, wall, john, deere, jenny, handed, be...	{'love': 31, 'baby': 30, 'summertime': 24, 'sa...
MGMT	[feeling, rough, feeling, raw, prime, life, le...	{'together': 66, 'change': 19, 'feel': 18, 'st...
Tegan and Sara	[little, bit, closer, come, little, closer, co...	{'go': 36, 'goodbye': 28, 'mind': 26, 'love': ...
The Decemberists	[late, night, counting, stitches, laying, side...	{'die': 40, 'life': 36, 'young': 34, 'head': 1...
The Shins	[girl, inform, senses, warn, clever, eyes, eas...	{'name': 15, 'painting': 12, 'hole': 12, 'noth...
Vampire Weekend	[baby, pain, natural, rain, thought, didnt, ra...	{'cheating': 52, 'baby': 18, 'thought': 18, 'n...
of Montreal	[mystery, stand, stare, nibble, ear, smell, oc...	{'hair': 14, 'girl': 14, 'different': 13, 'sou...

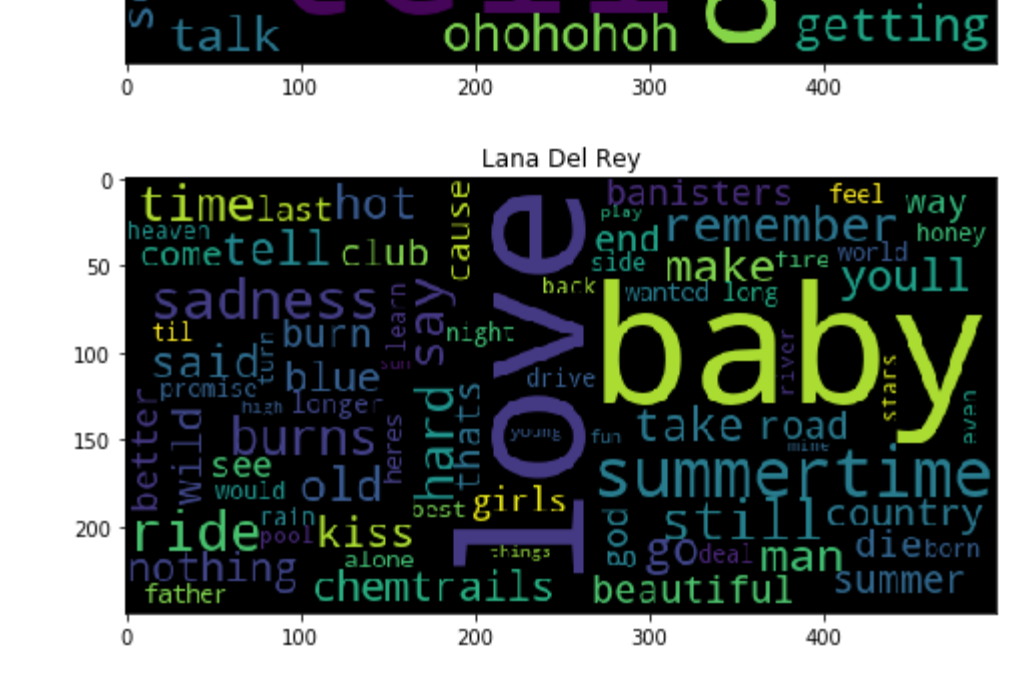
With this newly created dataframe we can now create a word cloud for each of the artists

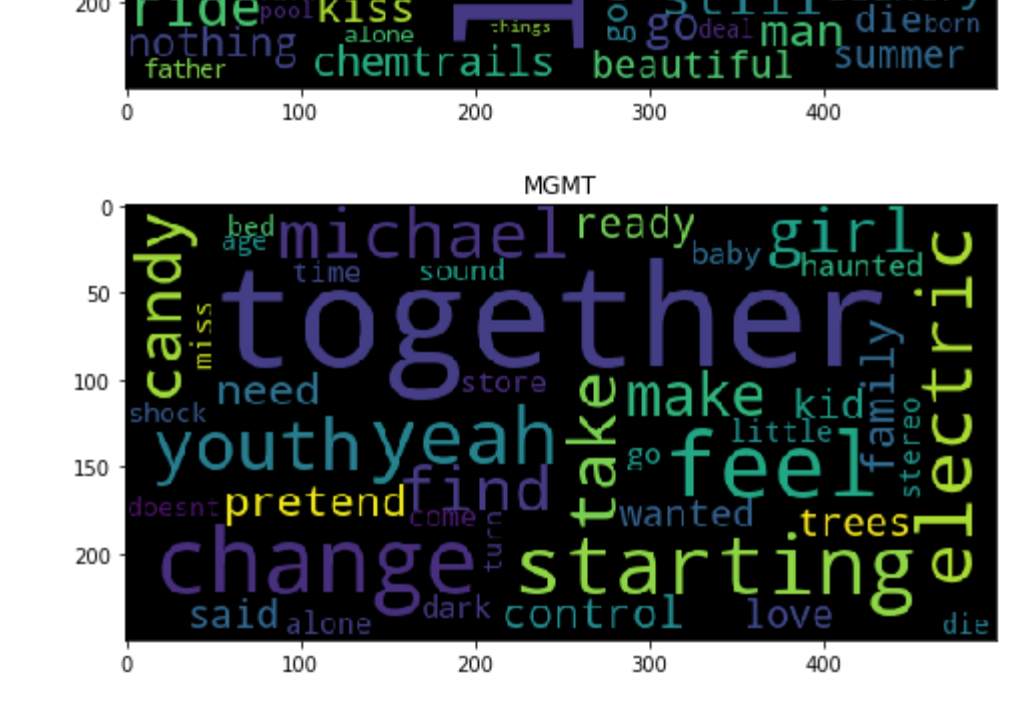
I will be doing this with a function that takes an input a dataframe and prints a word cloud for each artist and their word counter.

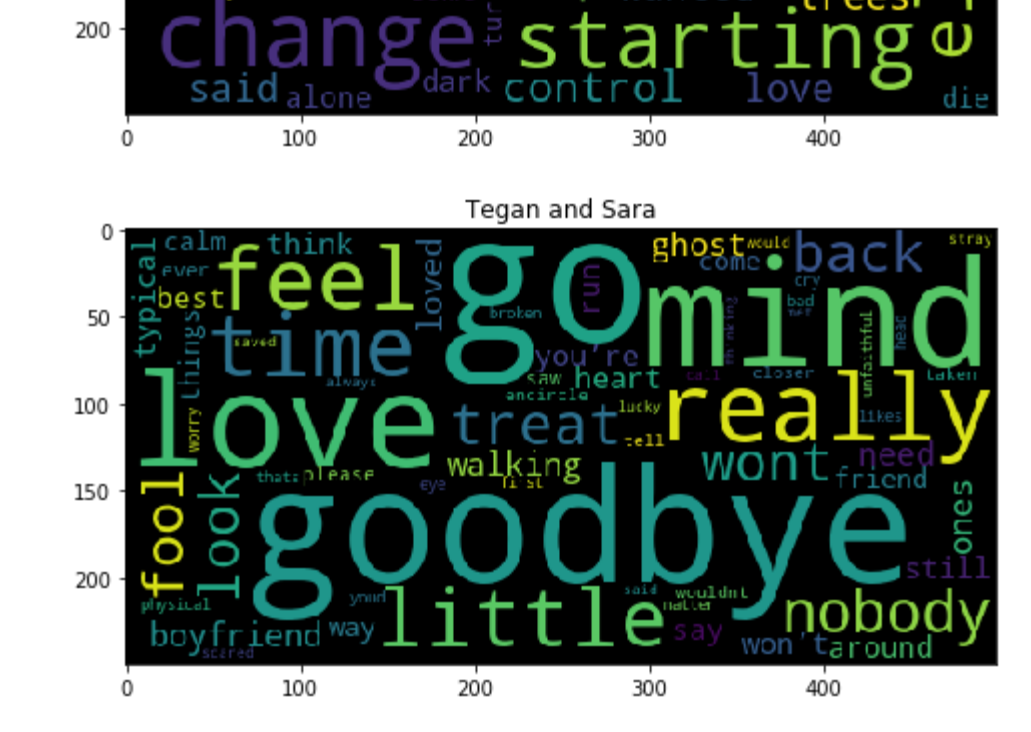
In [24]:
```
import matplotlib.pyplot as plt
from wordcloud import WordCloud

def word_cloud(data):
    c, r = data.shape
    for i in range(c):
        e = data["word counter"].iloc[i]
        wordcloud = WordCloud(width = 500, height = 250).generate_from_frequencies(e)

        plt.figure(figsize=(8,4))
        plt.imshow(wordcloud)
        plt.title(data.index[i])

wordcloud = word_cloud(word_lyrics)
wordcloud
```

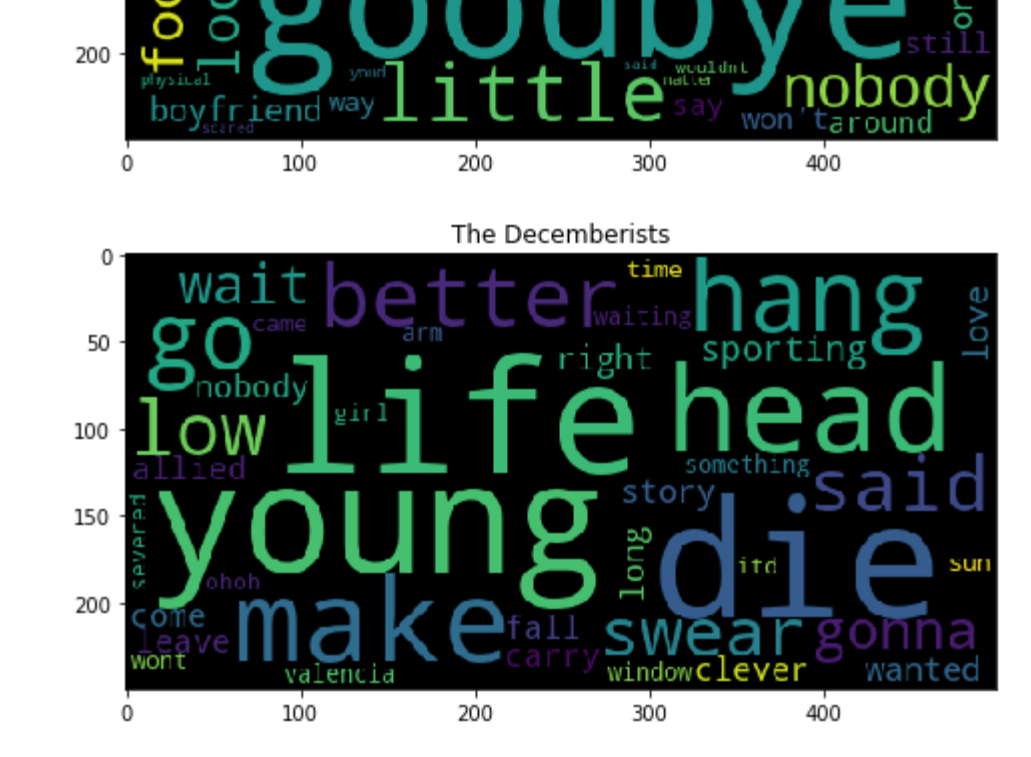


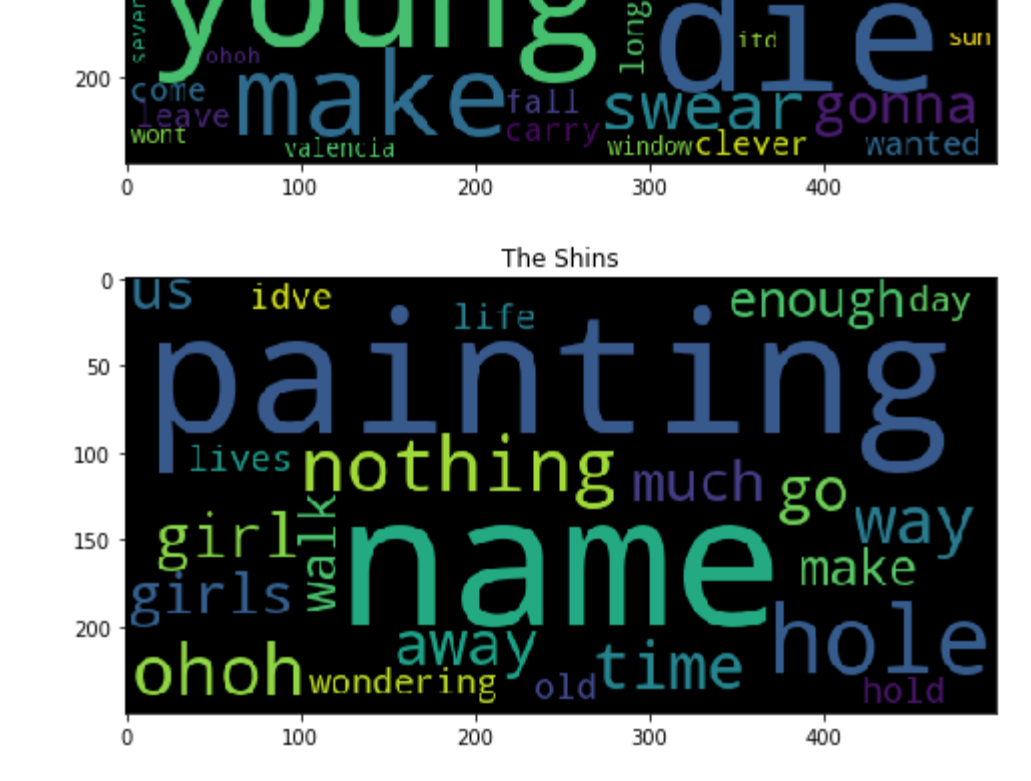


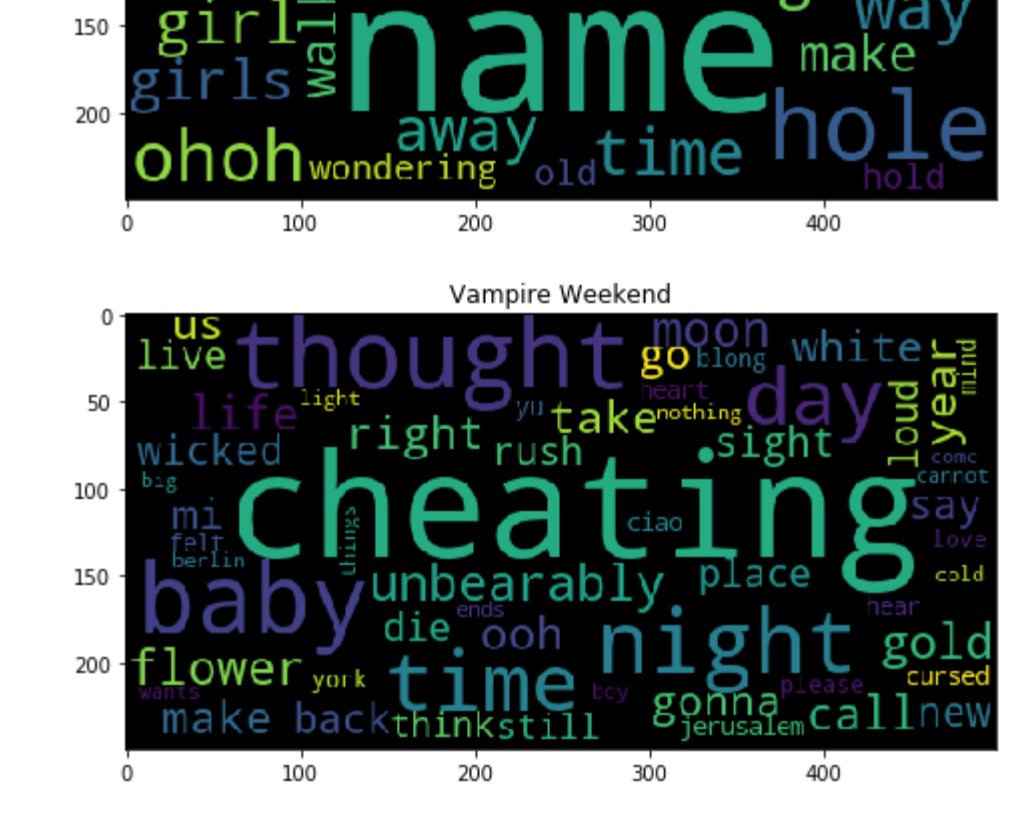


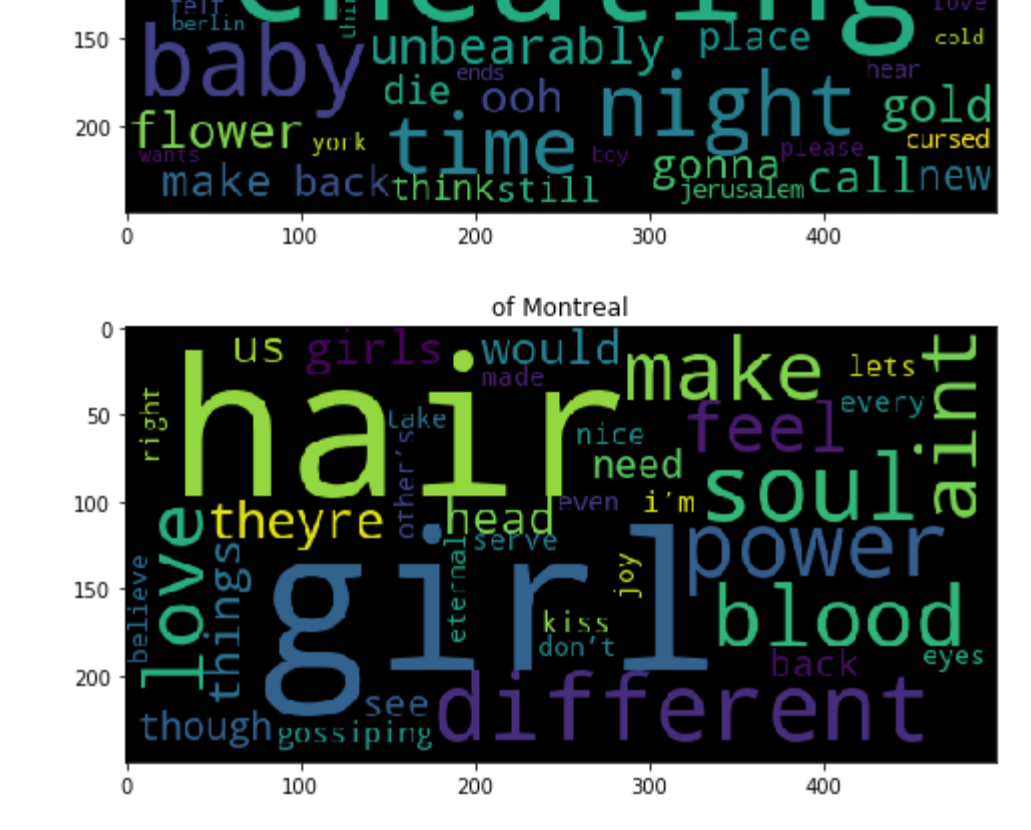












I mainly use the word clouds to make sure my data is making sense, but I feel it is hard to get any good comparison with just this. However, it is a good way to find words to include in your stopwords list. Many of the words I have included in that list came from here.

To find common topics between the artists and their songs we will be using LDA.

LDA is an unsupervised method that puts words in different topics boxes, for a more detailed explanation pease visit this site https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf.

We will be creating a data frame matrix which contains every word as one of its columns and each row will have the frequency for each artist.

First we will create a new data frame called combined_lyrics where we combine all of the songs' lyrics into a single long string for each artist.

In [25]:
```
combined_lyrics = df.groupby('artist')['lyrics'].apply(','.join).reset_index()
combined_lyrics
```

Out[25]:

	artist	lyrics
0	Belle and Sebastian	a mile and a half on a bus takes a long time\n...
1	Feist	one two three four\ntell me that you love me m...
2	Kings of Convenience	if you want to be my friend\nyou want us to ge...
3	Lana Del Rey	theres a picture on the wall\nof me on a john ...
4	MGMT	im feeling rough im feeling raw\nim in the pri...
5	Tegan and Sara	all i want to get is a little bit closer\nall ...
6	The Decemberists	it was late one night i was counting out my st...
7	The Shins	girl inform me all my senses warn me\nyour cle...
8	Vampire Weekend	baby i know pain is as natural as the rain\ni ...
9	of Montreal	youre such a mystery\ni just want to stand and...

Now we will use the library sklearn to create that matrix data frame.

In [67]:
```
from sklearn.feature_extraction import text
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(stop_words=stopwords)
data_cv = cv.fit_transform(combined_lyrics.lyrics) ##We are transforming the column lyrics from the com
bined_lyrics df
data_stop = pd.DataFrame(data_cv.toarray(), columns=cv.get_feature_names())
data_stop.index = word_lyrics.index

data_stop
```

Out[67]:

artist	10	12	1917	1960	1embedshare	24	25	318	323embedshare	344embedshare	...	york	you2embedshare	youd	youll
Belle and Sebastian	1	0	0	2	2	0	0	2	0	0	...	2	1	2	2
Feist	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
Kings of Convenience	0	0	0	0	1	1	1	0	0	0	...	0	0	2	1
Lana Del Rey	0	0	0	0	0	0	0	0	1	2	...	0	0	3	10
MGMT	0	0	0	0	0	0	0	0	0	0	...	0	0	0	1
Tegan and Sara	0	0	0	0	0	0	0	0	0	0	...	0	0	6	0
The Decemberists	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0
The Shins	0	0	0	0	0	0	0	0	0	0	...	0	0	1	3
Vampire Weekend	0	0	1	0	0	0	0	0	0	0	...	6	0	0	0
of Montreal	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0

10 rows × 2913 columns

Let's transpose the data frame as it is easier to read that way and will help when we run LDA on it.

In [68]:
```
tds = data_stop.transpose()
tds
```

Out[68]:

artist	Belle and Sebastian	Feist	Kings of Convenience	Lana Del Rey	MGMT	Tegan and Sara	The Decemberists	The Shins	Vampire Weekend	of Montreal
10	1	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	1	0	0	0
1917	0	0	0	0	0	0	0	0	1	0
1960	2	0	0	0	0	0	0	0	0	0
1embedshare	2	0	1	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
youngembedshare	0	0	0	0	0	0	1	0	0	0
youth	0	1	0	0	11	0	0	0	0	0
youth14embedshare	0	0	0	0	1	0	0	0	0	0
yu	0	0	0	0	0	0	0	0	6	0
zombies	0	0	0	0	0	0	0	2	0	0

2913 rows × 10 columns

We will be using the gensim library to run LDA as well as sparse from the Scipy library

In [69]:
```
import smart_open
smart_open.open = smart_open.smart_open
from gensim import matutils, models
import scipy.sparse

sparse_counts = scipy.sparse.csr_matrix(tds) ## We are running this on tds - the transposed data frame
 matrix.
corpus = matutils.Sparse2Corpus(sparse_counts)
```

We also need to give each word an id, which will be stored in a dictionary called id2word

In [70]:
```
id2word = dict((v, k) for k, v in cv.vocabulary_.items())
```

We are now finally ready to run LDA.

Let's go ahead and run it for 2, 3, 4, and 5 topics with a few passes (10 passes). If we find one that seems significant we can run it for more passes.

In [75]:
```
lda = models.LdaModel(corpus=corpus, id2word=id2word, num_topics=2, passes=10)
lda.print_topics()
```

Out[75]:
```
[(0,
  '0.010*"together" + 0.010*"love" + 0.008*"go" + 0.008*"life" + 0.008*"die" + 0.007*"cheating" + 0.0
07*"make" + 0.006*"time" + 0.006*"come" + 0.006*"young"'),
 (1,
  '0.011*"love" + 0.008*"go" + 0.006*"time" + 0.005*"feel" + 0.005*"boy" + 0.005*"back" + 0.005*"min
d" + 0.005*"baby" + 0.005*"say" + 0.005*"still"')]
```

In [72]:
```
lda = models.LdaModel(corpus=corpus, id2word=id2word, num_topics=3, passes=10)
lda.print_topics()
```

Out[72]:
```
[(0,
  '0.011*"boy" + 0.010*"tell" + 0.006*"see" + 0.006*"around" + 0.006*"poor" + 0.005*"love" + 0.005*"b
eat" + 0.005*"tonight" + 0.005*"jump" + 0.005*"line"'),
 (1,
  '0.015*"love" + 0.013*"go" + 0.010*"cheating" + 0.009*"time" + 0.007*"mind" + 0.007*"come" + 0.006
*"heart" + 0.006*"back" + 0.006*"goodbye" + 0.006*"feel"'),
 (2,
  '0.012*"together" + 0.010*"love" + 0.010*"die" + 0.008*"make" + 0.008*"life" + 0.007*"young" + 0.00
7*"baby" + 0.006*"go" + 0.006*"feel" + 0.005*"girl"')]
```

In [77]:
```
lda = models.LdaModel(corpus=corpus, id2word=id2word, num_topics=4, passes=10)
lda.print_topics()
```

Out[77]:
```
[(0,
  '0.017*"love" + 0.011*"go" + 0.007*"time" + 0.007*"feel" + 0.007*"come" + 0.005*"say" + 0.005*"bac
k" + 0.005*"baby" + 0.005*"make" + 0.005*"mind"'),
 (1,
  '0.011*"cheating" + 0.010*"boy" + 0.008*"tell" + 0.006*"thought" + 0.006*"time" + 0.006*"day" + 0.0
05*"feels" + 0.005*"love" + 0.005*"take" + 0.005*"see"'),
 (2,
  '0.024*"die" + 0.021*"life" + 0.020*"young" + 0.011*"head" + 0.011*"make" + 0.010*"hang" + 0.008*"g
o" + 0.007*"better" + 0.005*"said" + 0.005*"low"'),
 (3,
  '0.037*"together" + 0.011*"change" + 0.010*"feel" + 0.009*"starting" + 0.008*"yeah" + 0.007*"electr
ic" + 0.006*"michael" + 0.006*"youth" + 0.006*"take" + 0.005*"girl"')]
```

In [78]:
```
lda = models.LdaModel(corpus=corpus, id2word=id2word, num_topics=5, passes=10)
lda.print_topics()
```

Out[78]:
```
[(0,
  '0.011*"life" + 0.010*"die" + 0.008*"boy" + 0.008*"young" + 0.007*"love" + 0.007*"he
ad" + 0.006*"girl" + 0.006*"go" + 0.005*"see"'),
 (1,
  '0.009*"name" + 0.007*"hole" + 0.007*"painting" + 0.007*"nothing" + 0.007*"ohoh" + 0.007*"way" + 0.
005*"time" + 0.004*"go" + 0.004*"girl" + 0.004*"away"'),
 (2,
  '0.022*"go" + 0.011*"goodbye" + 0.010*"mind" + 0.015*"love" + 0.012*"really" + 0.011*"time" + 0.011
*"feel" + 0.009*"little" + 0.009*"nobody" + 0.008*"time"'),
 (3,
  '0.013*"cheating" + 0.010*"baby" + 0.010*"love" + 0.008*"tell" + 0.007*"say" + 0.006*"ohoh" + 0.006
*"alone" + 0.006*"day" + 0.006*"thought"'),
 (4,
  '0.028*"together" + 0.011*"feel" + 0.010*"go" + 0.010*"change" + 0.00
*"alone" + 0.007*"starting" + 0.006*"heart" + 0.006*"find"')]
```

I don't feel I have gathered anything that gives a commonality to each topic. Let's try something different.

Let's create a filter that we only consider the nouns from the lyrics.

First we will need to create a function that uses the nltk library to classify all the words in our lyrics and then just get those that are labeled as nouns. We will create a new data frame called nouns which is a copy of our combined lyrics data frame with an added column which contains the nouns contained in the lyrics column.

In [79]:
```
from nltk import word_tokenize, pos_tag

def nouns(text):
    '''Given a string of text, tokenize the text and pull out only the nouns.'''
    is_noun = lambda pos: pos[:2] == 'NN'
    tokenized = word_tokenize(text)
    all_nouns = [word for (word, pos) in pos_tag(tokenized) if is_noun(pos)]
    return ' '.join(all_nouns)

data_nouns = combined_lyrics.copy()
data_nouns['nouns'] = data_nouns['lyrics'].apply(lambda x: nouns(x))
data_nouns
```

Out[79]:

	artist	lyrics	nouns
0	Belle and Sebastian	a mile and a half on a bus takes a long time\n...	mile half bus time odour prison food time day ...
1	Feist	one two three four\ntell me that you love me m...	tell nights youth teenage hopes door nothing y...
2	Kings of Convenience	if you want to be my friend\nyou want us to ge...	friend please observation im demeanor friend m...
3	Lana Del Rey	theres a picture on the wall\nof me on a john ...	theres picture wall john beer hell oklahoma mm...
4	MGMT	im feeling rough im feeling raw\nim in the pri...	im im im prime life lets music money models wi...
5	Tegan and Sara	all i want to get is a little bit closer\nall ...	i bit i closer breath bit rush doors wind nigh...
6	The Decemberists	it was late one night i was counting out my st...	night i stitches side road i time britches tim...
7	The Shins	girl inform me all my senses warn me\nyour cle...	girl senses eyes backwards sighs spit eye lips...
8	Vampire Weekend	baby i know pain is as natural as the rain\ni ...	baby i pain rain i rain california baby i isnt...
9	of Montreal	youre such a mystery\ni just want to stand and...	youre mystery i ear ocean hair i knot blues bl...

In [80]:
```
from sklearn.feature_extraction import text
from sklearn.feature_extraction.text import CountVectorizer

cvn = CountVectorizer(stop_words=stopwords)
data_cvn = cvn.fit_transform(data_nouns.nouns)
data_stopn = pd.DataFrame(data_cvn.toarray(), columns=cvn.get_feature_names())
data_stopn.index = word_lyrics.index

data_stopn
```

Out[80]:

artist	abberation	abrades	accusations	act	acts	admit	advantage	affair	affection	afraid	...	yesterday	york	you2embedsha
Belle and Sebastian	1	0	0	0	0	0	0	1	0	2	...	0	2	
Feist	0	0	0	0	0	1	0	0	0	0	...	0	0	
Kings of Convenience	0	0	1	0	0	0	0	0	0	0	...	0	0	
Lana Del Rey	0	0	0	0	0	0	0	0	0	0	...	0	0	
MGMT	0	0	0	0	0	0	0	0	0	0	...	0	0	
Tegan and Sara	0	0	0	0	0	0	1	0	0	0	...	0	0	
The Decemberists	0	1	0	0	0	0	0	0	0	0	...	0	0	
The Shins	0	0	0	1	0	0	0	0	0	0	...	0	0	
Vampire Weekend	0	0	0	0	0	0	0	0	4	0	...	0	6	
of Montreal	0	0	0	0	1	0	0	0	0	0	...	1	0	

10 rows × 1608 columns

In [81]:
```
corpusn = matutils.Sparse2Corpus(scipy.sparse.csr_matrix(data_stopn.transpose()))
id2wordn = dict((v, k) for k, v in cvn.vocabulary_.items())
```

Again let's run lda for the nouns of the lyrics with 2, 3, and 4 topics.

In [82]:
```
ldan = models.LdaModel(corpus=corpusn, num_topics=2, id2word=id2wordn, passes=10)
ldan.print_topics()
```

Out[82]:
```
[(0,
  '0.017*"time" + 0.016*"life" + 0.013*"love" + 0.011*"baby" + 0.011*"mind" + 0.010*"heart" + 0.008
*"night" + 0.008*"head" + 0.007*"nobody" + 0.007*"go"'),
 (1,
  '0.012*"boy" + 0.010*"girl" + 0.008*"time" + 0.007*"day" + 0.007*"way" + 0.006*"nothing" + 0.006*"n
ame" + 0.006*"jump" + 0.005*"life" + 0.005*"beat"')]
```

In [83]:
```
ldan = models.LdaModel(corpus=corpusn, num_topics=3, id2word=id2wordn, passes=10)
ldan.print_topics()
```

Out[83]:
```
[(0,
  '0.012*"yeah" + 0.011*"youth" + 0.011*"feel" + 0.009*"girl" + 0.008*"family" + 0.008*"kid" + 0.008
*"candy" + 0.007*"control" + 0.007*"trees" + 0.007*"store"'),
 (1,
  '0.021*"life" + 0.011*"time" + 0.011*"head" + 0.010*"girl" + 0.010*"day" + 0.008*"nothing" + 0.008
*"night" + 0.007*"wan" + 0.007*"things" + 0.007*"power"'),
 (2,
  '0.015*"time" + 0.015*"love" + 0.012*"boy" + 0.011*"mind" + 0.010*"heart" + 0.009*"way" + 0.008*"ba
by" + 0.008*"go" + 0.008*"goodbye" + 0.007*"summertime"')]
```

In [84]:
```
ldan = models.LdaModel(corpus=corpusn, num_topics=4, id2word=id2wordn, passes=10)
ldan.print_topics()
```

Out[84]:
```
[(0,
  '0.015*"baby" + 0.015*"love" + 0.014*"summertime" + 0.013*"time" + 0.011*"sadness" + 0.010*"road" +
 0.010*"haha" + 0.009*"heart" + 0.009*"nothing" + 0.008*"limit"'),
 (1,
  '0.019*"life" + 0.013*"boy" + 0.011*"time" + 0.009*"girl" + 0.008*"head" + 0.007*"way" + 0.007*"da
y" + 0.007*"name" + 0.006*"line" + 0.006*"nothing"'),
 (2,
  '0.024*"mind" + 0.022*"time" + 0.017*"go" + 0.016*"goodbye" + 0.016*"wan" + 0.014*"love" + 0.013*"n
ight" + 0.012*"heart" + 0.011*"nobody" + 0.011*"baby"'),
 (3,
  '0.016*"hair" + 0.014*"power" + 0.012*"girl" + 0.012*"blood" + 0.010*"soul" + 0.009*"head" + 0.009
*"things" + 0.009*"girls" + 0.008*"love" + 0.006*"joy"')]
```

I still haven't found what each topic means, let's see if by using adjectives as well we can derive more insight from lda.

Now let's consider nouns and adjectives only.

In [85]:
```
def nouns_adj(text):
    '''Given a string of text, tokenize the text and pull out only the nouns.'''
    is_noun = lambda pos: pos[:2] == 'NN'
    tokenized = word_tokenize(text)
    all_nouns = [word for (word, pos) in pos_tag(tokenized) if is_noun(pos)]
    return ' '.join(all_nouns)

data_nouns_adj = data_nouns.copy()
data_nouns_adj['nouns_adj'] = data_nouns_adj['lyrics'].apply(lambda x: nouns(x))
data_nouns_adj
```

Out[85]:

	artist	lyrics	nouns	nouns_adj
0	Belle and Sebastian	a mile and a half on a bus takes a long time\n...	mile half bus time odour prison food time day ...	mile half bus time odour prison food time day ...
1	Feist	one two three four\ntell me that you love me m...	tell nights youth teenage hopes door nothing y...	tell nights youth teenage hopes door nothing y...
2	Kings of Convenience	if you want to be my friend\nyou want us to ge...	friend please observation im demeanor friend m...	friend please observation im demeanor friend m...
3	Lana Del Rey	theres a picture on the wall\nof me on a john ...	theres picture wall john beer hell oklahoma mm...	theres picture wall john beer hell oklahoma mm...
4	MGMT	im feeling rough im feeling raw\nim in the pri...	im im im prime life lets music money models wi...	im im im prime life lets music money models wi...
5	Tegan and Sara	all i want to get is a little bit closer\nall ...	i bit i closer breath bit rush doors wind nigh...	i bit i closer breath bit rush doors wind nigh...
6	The Decemberists	it was late one night i was counting out my st...	night i stitches side road i time britches tim...	night i stitches side road i time britches tim...
7	The Shins	girl inform me all my senses warn me\nyour cle...	girl senses eyes backwards sighs spit eye lips...	girl senses eyes backwards sighs spit eye lips...
8	Vampire Weekend	baby i know pain is as natural as the rain\ni ...	baby i pain rain i rain california baby i isnt...	baby i pain rain i rain california baby i isnt...
9	of Montreal	youre such a mystery\ni just want to stand and...	youre mystery i ear ocean hair i knot blues bl...	youre mystery i ear ocean hair i knot blues bl...

In [86]:
```
cvna = CountVectorizer(stop_words=stopwords, max_df=.8)
data_cvna = cvna.fit_transform(data_nouns_adj.nouns_adj)
data_dtmna = pd.DataFrame(data_cvna.toarray(), columns=cvna.get_feature_names())
data_dtmna.index = data_nouns_adj['artist']
data_dtmna
```

Out[86]:

artist	abberation	abrades	accusations	act	acts	admit	advantage	affair	affection	afraid	...	yesterday	york	you2embedsha
Belle and Sebastian	1	0	0	0	0	0	0	1	0	2	...	0	2	
Feist	0	0	0	0	0	1	0	0	0	0	...	0	0	
Kings of Convenience	0	0	1	0	0	0	0	0	0	0	...	0	0	
Lana Del Rey	0	0	0	0	0	0	0	0	0	0	...	0	0	
MGMT	0	0	0	0	0	0	0	0	0	0	...	0	0	
Tegan and Sara	0	0	0	0	0	0	1	0	0	0	...	0	0	
The Decemberists	0	1	0	0	0	0	0	0	0	0	...	0	0	
The Shins	0	0	0	1	0	0	0	0	0	0	...	0	0	
Vampire Weekend	0	0	0	0	0	0	0	0	4	0	...	0	6	
of Montreal	0	0	0	0	1	0	0	0	0	0	...	1	0	

10 rows × 1597 columns

In [87]:
```
corpusna = matutils.Sparse2Corpus(scipy.sparse.csr_matrix(data_dtmna.transpose()))
id2wordna = dict((v, k) for k, v in cvna.vocabulary_.items())
```

In [88]:
```
ldana = models.LdaModel(corpus=corpusna, num_topics=2, id2word=id2wordna, passes=10)
ldana.print_topics()
```

Out[88]:
```
[(0,
  '0.011*"boy" + 0.010*"day" + 0.009*"baby" + 0.008*"girl" + 0.008*"head" + 0.007*"nothing" + 0.005
*"cause" + 0.005*"summertime" + 0.005*"tonight" + 0.005*"girls"'),
 (1,
  '0.015*"mind" + 0.012*"go" + 0.011*"goodbye" + 0.009*"nobody" + 0.007*"ones" + 0.007*"haha" + 0.007
*"yeah" + 0.006*"nothing" + 0.006*"fool" + 0.006*"boyfriend"')]
```

In [92]:
```
ldana = models.LdaModel(corpus=corpusna, num_topics=3, id2word=id2wordna, passes=10)
ldana.print_topics()
```

Out[92]:
```
[(0,
  '0.019*"baby" + 0.009*"summertime" + 0.008*"nothing" + 0.007*"wan" + 0.007*"sadness" + 0.007*"girl"
 + 0.006*"day" + 0.006*"yeah" + 0.006*"end" + 0.006*"mind"'),
 (1,
  '0.011*"hair" + 0.011*"haha" + 0.010*"power" + 0.010*"limit" + 0.009*"girl" + 0.009*"blood" + 0.008
*"head" + 0.008*"soul" + 0.006*"day" + 0.006*"girls"'),
 (2,
  '0.015*"boy" + 0.012*"mind" + 0.012*"head" + 0.010*"nobody" + 0.010*"go" + 0.010*"goodbye" + 0.008
*"day" + 0.007*"name" + 0.007*"beat" + 0.006*"nothing"')]
```

In [90]:
```
ldana = models.LdaModel(corpus=corpusna, num_topics=4, id2word=id2wordna, passes=10)
ldana.print_topics()
```

Out[90]:
```
[(0,
  '0.020*"head" + 0.009*"hang" + 0.008*"gon" + 0.008*"story" + 0.008*"nobody" + 0.008*"dance" + 0.007
*"something" + 0.007*"window" + 0.007*"valencia" + 0.006*"girl"'),
 (1,
  '0.013*"baby" + 0.010*"day" + 0.010*"wan" + 0.009*"haha" + 0.008*"girl" + 0.008*"yeah" + 0.008*"moo
n" + 0.008*"limit" + 0.008*"youth" + 0.007*"nothing"'),
 (2,
  '0.014*"baby" + 0.013*"summertime" + 0.013*"nothing" + 0.011*"sadness" + 0.011*"name" + 0.009*"hol
e" + 0.007*"cause" + 0.007*"man" + 0.007*"burns" + 0.007*"end"'),
 (3,
  '0.018*"boy" + 0.015*"mind" + 0.012*"goodbye" + 0.012*"go" + 0.009*"nobody" + 0.009*"jump" + 0.008
*"girl" + 0.008*"head" + 0.008*"beat" + 0.008*"party"')]
```

Eventhough I have failed to determine what each topic means, let's see under what topic each artist would belong to if we were to use the nouns and adjective method with four topics.

In [91]:
```
corpus_transformed = ldana[corpusna]
list(zip([a for [(a,b)] in corpus_transformed], data_stop.index))
```

Out[91]:
```
[(3, 'Belle and Sebastian'),
 (1, 'Feist'),
 (0, 'Kings of Convenience'),
 (2, 'Lana Del Rey'),
 (1, 'MGMT'),
 (3, 'Tegan and Sara'),
 (0, 'The Decemberists'),
 (2, 'The Shins'),
 (1, 'Vampire Weekend'),
 (3, 'of Montreal')]
```

I have failed to find what the topics mean for all the different methods I have used:

1. Using words that excluded stop words (and others I included)
2. Including only nouns
3. Including only nouns and adjectives.

In other words I have not found a method where the topics made sense on what they were representing.

Conclusion: Artists that belong to the indie/pop genre are very similar to each other.

I initially set out to figure out which indie/pop artists were similar to Kings of Convenience. However, my conclusion is that most of the artists from that genre are too similar to try to find big differences among them. I reach this conclusion because they had very similar ratings for polarity and subjectivity. Also, because using LDA (with three different methods) I could not find any clear topics that were shared by some artists and excluded by others. My next experiment will be trying this notebook with artists from different genres.

Limitations

As mentioned in the conclusion some limitations include that artists were too similar to each other as they were all from the same genre. Also, I might get better results if I include more artists and more songs for each artist.

References

https://www.youtube.com/watch?v=xvqsFTUsOmc&ab_channel=PyOhio