ADS 509 Module 3: Group Comparison

The task of comparing two groups of text is fundamental to textual analysis. There are innumerable applications: survey respondents from different segments of customers, speeches by different political parties, words used in Tweets by different constituencies, etc. In this assignment you will build code to effect comparisons between groups of text data, using the ideas learned in reading and lecture.

This assignment asks you to analyze the lyrics and Twitter descriptions for the two artists you selected in Module 1. If the results from that pull were not to your liking, you are welcome to use the zipped data from the "Assignment Materials" section. Specifically, you are asked to do the following:

- Read in the data, normalize the text, and tokenize it. When you tokenize your Twitter descriptions, keep hashtags and emojis in your token set.
- Calculate descriptive statistics on the two sets of lyrics and compare the results.
- For each of the four corpora, find the words that are unique to that corpus.
- Build word clouds for all four corpora.

Each one of the analyses has a section dedicated to it below. Before beginning the analysis there is a section for you to read in the data and do your cleaning (tokenization and normalization).

General Assignment Instructions

These instructions are included in every assignment, to remind you of the coding standards for the class. Feel free to delete this cell after reading it.

One sign of mature code is conforming to a style guide. We recommend the Google Python Style Guide. If you use a different style guide, please include a cell with a link.

Your code should be relatively easy-to-read, sensibly commented, and clean. Writing code is a messy process, so please be sure to edit your final submission. Remove any cells that are not needed or parts of cells that contain unnecessary code. Remove inessential `import` statements and make sure that all such statements are moved into the designated cell.

Make use of non-code cells for written commentary. These cells should be grammatical and clearly written. In some of these cells you will have questions to answer. The questions will be marked by a "Q:" and will have a corresponding "A:" spot for you. *Make sure to answer every question marked with a* `Q:` *for full credit.*

In [1]:
```
import os
import re
import emoji
import pandas as pd

from collections import Counter, defaultdict
from nltk.corpus import stopwords
from string import punctuation
from wordcloud import WordCloud

from sklearn.feature_extraction.text import TfidfTransformer, CountVectorizer
```

In [333…
```
# Use this space for any additional import statements you need
#!pip install wordcloud
```

```
#!pip install sklearn

import numpy as np
```

In [310…

```
# Place any addtional functions or constants you need here.

# Some punctuation variations
punctuation = set(punctuation) # speeds up comparison
tw_punct = punctuation - {"#"}

# Stopwords
sw = stopwords.words("english")

# Two useful regex
whitespace_pattern = re.compile(r"\s+")
hashtag_pattern = re.compile(r"^#[0-9a-zA-Z]+")

# It's handy to have a full set of emojis
all_language_emojis = set()

for country in emoji.UNICODE_EMOJI :
    for em in emoji.UNICODE_EMOJI[country] :
        all_language_emojis.add(em)

# and now our functions
def descriptive_stats(tokens, num_tokens = 5, verbose=True) :
    """
        Given a list of tokens, print number of tokens, number of unique tokens,
        number of characters, lexical diversity, and num_tokens most common
        tokens. Return a list of
    """

    # Fill in the correct values here.
    tokes = tokens.split()
    num_tokens = sum(map(len, (s.split() for s in tokes)))
    num_unique_tokens = len(set(w.lower() for w in tokes))
    lexical_diversity = num_unique_tokens / num_tokens
    num_characters = sum(list(map(len, tokes)))

    if verbose :
        print(f"There are {num_tokens} tokens in the data.")
        print(f"There are {num_unique_tokens} unique tokens in the data.")
        print(f"There are {num_characters} characters in the data.")
        print(f"The lexical diversity is {lexical_diversity:.3f} in the data.")

        # print the five most common tokens

    return([num_tokens, num_unique_tokens,
            lexical_diversity,
            num_characters])

def is_emoji(s):
    return(s in all_language_emojis)

def contains_emoji(s):

    s = str(s)
    emojis = [ch for ch in s if is_emoji(ch)]

    return(len(emojis) > 0)


def remove_stop(tokens) :
    # modify this function to remove stopwords
    tokens_wo_sw = [word for word in tokens if not word in sw]
```

```
    return(tokens_wo_sw)

def remove_punctuation(text, punct_set=tw_punct) :
    return("".join([ch for ch in text if ch not in punct_set]))

def tokenize(text) :
    """ Splitting on whitespace rather than the book's tokenize function. That
        function will drop tokens like '#hashtag' or '2A', which we need for Twitter. ""

    # modify this function to return tokens
    text = text.split()
    return(text)

def prepare(text, pipeline) :
    tokens = str(text)

    for transform in pipeline :
        tokens = transform(tokens)

    return(tokens)
```

Data Ingestion

Use this section to ingest your data into the data structures you plan to use. Typically this will be a dictionary or a pandas DataFrame.

In [452…
```
# Feel fre to use the below cells as an example or read in the data in a way you prefer

data_location = "/users/lenny/" # change to your location if it is not in the same direc
twitter_folder = "twitter/"
lyrics_folder = "lyrics/"

artist_files = {'mtrench':'mtrench_followers_data.txt',
                'NateWantsToBtl':'NateWantsToBtl_followers_data.txt'}
```

In [453…
```
twitter_data = pd.read_csv(data_location + twitter_folder + artist_files['mtrench'],
                           sep="\t",
                           quoting=3)

twitter_data['artist'] = "mtrench"
```

In [454…
```
twitter_data_2 = pd.read_csv(data_location + twitter_folder + artist_files['NateWantsToB
                             sep="\t",
                             quoting=3)
twitter_data_2['artist'] = "NateWantsToBtl"

twitter_data = pd.concat([
    twitter_data,twitter_data_2])

del(twitter_data_2)
twitter_data = twitter_data.drop('Unnamed: 0', axis=1)
```

In [455…
```
twitter_data.head()
```

Out[455]:

	Screen Name	Name	ID	Location	Follower Count	Friend Count	Description	artist
0	Flatprobably	Flat	1491491320778608642	NaN	0.0	6.0	aaa	mtrench
1	lennyj89	Leonard Littleton	250520140	Mount Sterling, KY	57.0	221.0	NaN	mtrench

2	KenishaShannon8	Kenisha Shannon	1511854002207608842	NaN	5.0	593.0	Kenisha . 17📞 . Join me on DM 💐	mtrench
3	BrookeFrein	Brooke Frein	1508824619637256196	NaN	16.0	722.0	Brooke . 19♡ . Just have fun🙏	mtrench
4	CourtenayPeder	Courtenay Peterson	228472111	British Columbia	92.0	353.0	ur mom is a dad joke. Her/She/x	mtrench

In [429…

```
# read in the lyrics here
artist_folders = os.listdir("lyrics/")
artist_folders = [f for f in artist_folders if os.path.isdir("lyrics/" + f)]
lyrics_data_list = []

for artist in artist_folders :
    artist_files = os.listdir("lyrics/" + artist)
    artist_files = [f for f in artist_files if 'txt' in f or 'csv' in f or 'tsv' in f]

    for f_name in artist_files :
        with open("lyrics/" + artist + "/" + f_name) as infile:
            lines = infile.read().replace('\n', ' ')
            lines.replace('\n', '')
            text = re.split(r'\s{3,}', lines)
            lyrics_data_list.append((artist, text[0], text[1]))

lyrics_data = pd.DataFrame(lyrics_data_list, columns = ['artist', 'title', 'lyrics'])
```

In [430…

```
lyrics_data.head()
```

Out[430]:

	artist	title	lyrics
0	marianas trench	Acadia	In the house I grew up in My room in the basem...
1	marianas trench	Alibis	From the scrapes and bruises To the familiar a...
2	marianas trench	Alive Again	I felt it turn to come and go Don't worry no o...
3	marianas trench	All To Myself	I don't patronize, I realize I'm losing and, t...
4	marianas trench	And So It Goes	In every heart There is a room A sanctuary is ...

Tokenization and Normalization

In this next section, tokenize and normalize your data. We recommend the following cleaning.

Lyrics

- Remove song titles
- Casefold to lowercase
- Remove punctuation
- Split on whitespace
- Remove stopwords (optional)

Removal of stopwords is up to you. Your descriptive statistic comparison will be different if you include stopwords, though TF-IDF should still find interesting features for you.

Twitter Descriptions

- Casefold to lowercase
- Remove punctuation other than emojis or hashtags
- Split on whitespace
- Remove stopwords

Removing stopwords seems sensible for the Twitter description data. Remember to leave in emojis and hashtags, since you analyze those.

In [456…
```
# apply the `pipeline` techniques from BTAP Ch 1 or 5

my_pipeline = [str.lower, remove_punctuation, tokenize, remove_stop]

lyrics_data["tokens"] = lyrics_data["lyrics"].apply(prepare,pipeline=my_pipeline)
lyrics_data["num_tokens"] = lyrics_data["tokens"].map(len)

twitter_data["tokens"] = twitter_data["Description"].apply(prepare,pipeline=my_pipeline)
twitter_data["num_tokens"] = twitter_data["tokens"].map(len)
```

In [457…
```
twitter_data = twitter_data.dropna()
twitter_data = twitter_data.reset_index()
twitter_data = twitter_data.drop('index', axis=1)
```

In [458…
```
twitter_data.head()
```

Out[458]:

	Screen Name	Name	ID	Location	Follower Count	Friend Count	Description	artist	token
0	CourtenayPeder	Courtenay Peterson	228472111	British Columbia	92.0	353.0	ur mom is a dad joke. Her/She/x	mtrench	[ur, mom dad, joke hershex
1	d73m	Donna Y	40288440	Scotland, United Kingdom	78.0	424.0	A life without music would be a lonely place. ...	mtrench	[life without music would lonely place, l.
2	grimoiretheband	Grimoire is a band!	1238646768126038017	Ontario, Canada	151.0	550.0	We're a group of 5 angsty teenagers from Missi...	mtrench	[we're group, 5 angsty teenagers mississau.
3	Alicat613	Ali Bowie	66555002	Ottawa, Ontario	697.0	1054.0	Festival Producer 🎶💞🐾⚙ Booking for #OGF2023 - ...	mtrench	[festiva producer 🎶💞🐾⚙ booking #ogf2023 .
4	WittleSquish	Squish	1099159332850024450	British Columbia, Canada	24.0	73.0	Semi-nerdy Canadian who likes to stream and is...	mtrench	[seminerdy canadian likes stream funny, ma.

In [459…
```
twitter_data['has_emoji'] = twitter_data["Description"].apply(contains_emoji)
```

Let's take a quick look at some descriptions with emojis.

In [460…
```
twitter_data[twitter_data.has_emoji].sample(10)[["artist","Description","tokens"]]
```

Out[460]:

	artist	Description	tokens
19434	mtrench	for legal reasons everything i post is a joke 💔	[legal, reasons, everything, post, joke, 💔]
32324	mtrench	probably at foodland // grayden 🤤💔	[probably, foodland, grayden, 🤤💔]
121525	NateWantsToBtl	I really like food,ouat,maximum ride, ,underta...	[really, like, foodouatmaximum, ride, undertal...
53012	mtrench	I love cats😻😻😻😻😻😻	[love, cats😻😻😻😻😻😻]
4546	mtrench	23 years old ✌\\ Cree Native ✊// ᴄᴀ // 😺Min ...	[23, years, old, ✌, cree, native, ✊, ᴄᴀ, 😺mi...
253	mtrench	Alhumdulillah for everything ♡	[Alhumdulillah, for, everything, ♡]
24619	mtrench	I have 1000 hobbies and no time Art: @arcaneba...	[1000, hobbies, time, art, arcanebat, next, co...
79859	NateWantsToBtl	personal account• mostly just a big sad 😭• i s...	[personal, account•, mostly, big, sad, 😭•, som...
7558	mtrench	F.B.G.M😖	[fbgm😖]
3842	mtrench	A good book and a great cup of coffee are a ne...	[good, book, great, cup, coffee, necessity, ❤]

With the data processed, we can now start work on the assignment questions.

Q: What is one area of improvement to your tokenization that you could theoretically carry out? (No need to actually do it; let's not make perfect the enemy of good enough.)

A: I think if we were able to tokenize the data as we read it in, it would be more efficient. Currently, we are iterating through the data a lot.

Calculate descriptive statistics on the two sets of lyrics and compare the results.

In [461…
```
# your code here
for artist in artist_folders:
    #create a temporary list of tokens through all sets of songs by artist
    temp_tokens = lyrics_data['tokens'].loc[lyrics_data['artist'] == artist]

    #create a list to store each token from each song
    token_list = []
    if artist == 'marianas trench':
        for i in range(len(temp_tokens)):
            for token in temp_tokens[i]:
                token_list.append(token)
    if artist == 'nate wants to battle':
        for i in range(71 ,len(temp_tokens)):
            for token in temp_tokens[i]:
                token_list.append(token)

    #create a string object of the list
    string_list = str(token_list)

    #generate the descriptive stats
    print("The statists for " + artist + " are:")
    descriptive_stats(string_list, verbose=True)
    print('\n')
```

```
The statists for marianas trench are:
There are 10849 tokens in the data.
There are 1580 unique tokens in the data.
There are 85874 characters in the data.
The lexical diversity is 0.146 in the data.


The statists for nate wants to battle are:
There are 35789 tokens in the data.
There are 4211 unique tokens in the data.
There are 281605 characters in the data.
The lexical diversity is 0.118 in the data.
```

Q: what observations do you make about these data?

A: Nate Wants to Battle has many more songs than Marianas Trench. It seems that as the number of unique tokens goes up, the lower the lexical diversty.

Find tokens uniquely related to a corpus

Typically we would use TF-IDF to find unique tokens in documents. Unfortunately, we either have too few documents, if we view each data source as a single document, or too many, if we view each description as a separate document. In the latter case, our problem will be that descriptions tend to be short, so our matrix would be too sparse to support analysis.

To get around this, we find tokens for each corpus that match the following criteria:

1. The token appears at least `n` times in all corpora
2. The tokens are in the top 10 for the highest ratio of appearances in a given corpora vs appearances in other corpora.

You will choose a cutoff for yourself based on the side of the corpus you're working with. If you're working with the Robyn-Cher corpora provided, `n=5` seems to perform reasonably well.

In [462…
```
def tokensOverThreshold(word_list, threshold):
    new_word_list = []
    unique_word_list = []
    corpora_length = len(word_list)
    corpora_wordcount = Counter(word_list)

    for word in word_list:
        new_word_list.append((word, corpora_wordcount[word], corpora_length, threshold))

    for tokes in new_word_list:
        if tokes not in unique_word_list:
            unique_word_list.append(tokes)

    df = pd.DataFrame(unique_word_list, columns = ['token', 'corpus_count', 'corpus_toke
    return df.sort_values(by=['corpus_count'], ascending=False)
```

In [476…
```
# your code here

#define a list for each corpus

#Marianas Trench Lyrics
corp1 = lyrics_data['tokens'].loc[lyrics_data['artist'] == 'marianas trench']
```

```
corpus1 = []
for i in range(len(corp1)):
    for token in corp1[i]:
        corpus1.append(token)

#Nate Wants to Battle Lyrics
corp2 = lyrics_data['tokens'].loc[lyrics_data['artist'] == 'nate wants to battle']
corpus2 = []
for i in range(71, len(corp2)):
    for token in corp2[i]:
        corpus2.append(token)

#Marianas Trench Twitter Descriptions
corp3 = twitter_data['Description'].loc[twitter_data['artist'] == 'mtrench']
corpus3 = []
for i in range(len(corp3)):
    for token in corp3[i].split():
        corpus3.append(token)

#Nate Wants to Battle Twitter Descriptions
corp4 = twitter_data['Description'].loc[twitter_data['artist'] == 'NateWantsToBtl']
corpus4 = []
for i in range(73130, len(corp4)):
    for token in corp4[i].split():
        corpus4.append(token)
```

In [478…

```
corpus1_df = tokensOverThreshold(corpus1, 50)
corpus1_df.head(10)
```

Out[478]:

	token	corpus_count	corpus_tokens	cutoff
92	know	254	10849	50
91	dont	236	10849	50
87	im	169	10849	50
138	like	152	10849	50
335	love	150	10849	50
65	ill	128	10849	50
190	cant	124	10849	50
16	one	120	10849	50
68	never	107	10849	50
120	get	104	10849	50

In [479…

```
corpus2_df = tokensOverThreshold(corpus2, 50)
corpus2_df.head(10)
```

Out[479]:

	token	corpus_count	corpus_tokens	cutoff
347	im	818	35789	50
77	dont	459	35789	50
58	know	442	35789	50
358	ill	384	35789	50
233	youre	362	35789	50
82	like	330	35789	50

232	see	327	35789	50
207	cant	321	35789	50
0	go	320	35789	50
302	never	282	35789	50

In [484…
```
corpus3_df = tokensOverThreshold(corpus3, 50)
corpus3_df.head(10)
```

In [482…
```
corpus4_df = tokensOverThreshold(corpus4, 50)
corpus4_df.head(10)
```

Out[482]:

	token	corpus_count	corpus_tokens	cutoff
60	|	2747	83716	50
17	and	2017	83716	50
33	I	1392	83716	50
7	a	1256	83716	50
128	to	962	83716	50
12	of	933	83716	50
10	the	825	83716	50
205	my	690	83716	50
53	•	620	83716	50
824	||	532	83716	50

Q: What are some observations about the top tokens? Do you notice any interesting items on the list?

A: It seems that the top tokens are repeated across different corpora.

Build word clouds for all four corpora.

For building wordclouds, we'll follow exactly the code of the text. The code in this section can be found here. If you haven't already, you should absolutely clone the repository that accompanies the book.

In [485…
```
from matplotlib import pyplot as plt

def wordcloud(word_freq, title=None, max_words=200, stopwords=None):

    wc = WordCloud(width=800, height=400,
                   background_color= "black", colormap="Paired",
                   max_font_size=150, max_words=max_words)

    # convert data frame into dict
    if type(word_freq) == pd.Series:
        counter = Counter(word_freq.fillna(0).to_dict())
    else:
        counter = word_freq

    # filter stop words in frequency counter
    if stopwords is not None:
        counter = {token:freq for (token, freq) in counter.items()
                              if token not in stopwords}
```

```
    wc.generate_from_frequencies(counter)

    plt.title(title)

    plt.imshow(wc, interpolation='bilinear')
    plt.axis("off")


def count_words(df, column='tokens', preprocess=None, min_freq=2):

    # process tokens and update counter
    def update(doc):
        tokens = doc if preprocess is None else preprocess(doc)
        counter.update(tokens)

    # create counter and run through all data
    counter = Counter()
    df[column].map(update)

    # transform counter into data frame
    freq_df = pd.DataFrame.from_dict(counter, orient='index', columns=['freq'])
    freq_df = freq_df.query('freq >= @min_freq')
    freq_df.index.name = 'token'

    return freq_df.sort_values('freq', ascending=False)
```

In [496…
```
count_words(twitter_data)
```

Out[496]:

token	freq
im	17467
love	14282
music	9662
•	8318
life	7945
…	…
gfl	2
arianna	2
😜😎	2
sham	2
ukeuro	2

41403 rows × 1 columns

In [497…
```
count_words(lyrics_data)
```

Out[497]:

token	freq
im	1249
dont	819
know	807

ill	594
youre	569
...	...
shop	2
purse	2
spilled	2
terrible	2
amends	2

3116 rows × 1 columns

Q: What observations do you have about these (relatively straightforward) wordclouds?

A: Couldn't get the wordcloud code to execute.

In [ ]: