ADS 509 Assignment 2.1: Tokenization, Normalization, Descriptive Statistics

This notebook holds Assignment 2.1 for Module 2 in ADS 509, Applied Text Mining. Work through this notebook, writing code and answering questions where required.

In the previous assignment you pulled lyrics data on two artists. In this assignment we explore this data set and a pull from the now-defunct Twitter API for the artists Cher and Robyn. If, for some reason, you did not complete that previous assignment, data to use for this assignment can be found in the assignment materials section of Canvas.

This assignment asks you to write a short function to calculate some descriptive statistics on a piece of text. Then you are asked to find some interesting and unique statistics on your corpora.

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
!pip install emoji
import emoji
import pandas as pd
import numpy as np

from collections import Counter, defaultdict
import nltk
nltk.download('stopwords')
```

```
from nltk.corpus import stopwords
from string import punctuation

sw = stopwords.words("english")
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: emoji in c:\users\ebbi_\appdata\roaming\python\python311\site-packages (2.10.0)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ebbi_\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [2]:
```
# Add any additional import statements you need here

import glob
```

In [3]:
```
# change `data_location` to the location of the folder on your machine.
data_location = "C:/USD/MSADS/Spring 24/ADS 509/Module 1/Assignment/Assignment 1.1/ADS-509-TM/"

# These subfolders should still work if you correctly stored the
# data from the Module 1 assignment
twitter_folder = "twitter/"
lyrics_folder = "lyrics/"
```

In [4]:
```
def descriptive_stats(tokens, num_tokens = 5, verbose=True) :
    counter = Counter()
    tokens.map(counter.update)
    frequency_df = pd.DataFrame.from_dict(counter, orient='index',
                                          columns=['freq'])

    counter_df = pd.DataFrame.from_dict(counter, orient='index').reset_index()

    # Fill in the correct values here.
    num_tokens = sum(frequency_df['freq'])
    num_unique_tokens = frequency_df.shape[0]
    lexical_diversity = num_unique_tokens / num_tokens
    num_characters = sum((counter_df['index'].str.len()) * counter_df[0])


    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
        print(f"The top 5 most common words are")
        print(counter.most_common(5))

    return([num_tokens, num_unique_tokens,
            lexical_diversity,
```

```
            num_characters])
```

In [5]:
```
sampleText = ["This is an Acura TLX",
              "TLX comes standard with Adaptive Cruise",
              "There is an optional HeadsUp Display",
              "There are two engine variations",
              "A Two Litre InLine 4 and a Three Litre V6"]

sampleText_df = pd.DataFrame(sampleText, columns=['Text'])
sampleText_df
```

Out[5]:

	Text
0	This is an Acura TLX
1	TLX comes standard with Adaptive Cruise
2	There is an optional HeadsUp Display
3	There are two engine variations
4	A Two Litre InLine 4 and a Three Litre V6

In [6]:
```
def tokenizeText(text):
    return re.findall(r'\w+(?:\.?\w+)*', text)
```

In [7]:
```
transform = [str.lower, tokenizeText]
def prepareData(text, transform):
    tokens = text
    for info in transform:
        tokens = info(tokens)
    return tokens
```

In [8]:
```
sampleText_df['tokens'] = sampleText_df['Text'].apply(
    prepareData, transform=transform)
sampleText_df['tokens']
```

Out[8]:
```
0                          [this, is, an, acura, tlx]
1       [tlx, comes, standard, with, adaptive, cruise]
2          [there, is, an, optional, headsup, display]
3                [there, are, two, engine, variations]
4    [a, two, litre, inline, 4, and, a, three, litr...
Name: tokens, dtype: object
```

In [9]:
```
#text = """here is some example text with other example text here in this text""".split()
assert(descriptive_stats(sampleText_df["tokens"],
                         verbose=True)[0] == 32)
assert(descriptive_stats(sampleText_df["tokens"],
                         verbose=False)[1] == 25)
```

```
assert(descriptive_stats(sampleText_df["tokens"],
                         verbose=False)[2] == 0.78125)
assert(descriptive_stats(sampleText_df["tokens"],
                         verbose=False)[3] == 140)
```

```
There are 32 tokens in the data.
There are 25 unique tokens in the data.
There are 140 characters in the data.
The lexical diversity is 0.781 in the data.
The top 5 most common words are
[('is', 2), ('an', 2), ('tlx', 2), ('there', 2), ('two', 2)]
```

Q: Why is it beneficial to use assertion statements in your code?

A: Assertions in code act like a safety net, helping catch and fix mistakes early on. They also serve as notes to explain how things should work, making it easier for developers to understand and collaborate. By using them, we ensure our code behaves as expected, catching issues before they reach the final product.

Data Input

Now read in each of the corpora. For the lyrics data, it may be convenient to store the entire contents of the file to make it easier to inspect the titles individually, as you'll do in the last part of the assignment. In the solution, I stored the lyrics data in a dictionary with two dimensions of keys: artist and song. The value was the file contents. A data frame would work equally well.

For the Twitter data, we only need the description field for this assignment. Feel free all the descriptions read it into a data structure. In the solution, I stored the descriptions as a dictionary of lists, with the key being the artist.

In [10]:
```
# Read in the lyrics data

os.chdir(data_location+lyrics_folder)

taylorswift_file_list = glob.glob(
    os.path.join(
        os.getcwd(), "taylorswift", "*.txt"))

taylorswiftlyrics = []

for file_path in taylorswift_file_list:
    with open(file_path) as f_input:
        lyrics = (f_input.read())
        file_name = file_path.split("/")[-1]
        taylorswiftlyrics.append(
            {
                'Artist': "Taylor Swift",
                'Song': file_name,
                'Lyrics': lyrics
```

```
            }
        )

taylorswiftlyrics = pd.DataFrame(taylorswiftlyrics)
```

In [11]:
```
# There are two artist in our module 1

eminem_file_list = glob.glob(
    os.path.join(
        os.getcwd(), "eminem", "*.txt"))

eminemlyrics = []

for file_path in eminem_file_list:
    with open(file_path) as f_input:
        lyrics = (f_input.read())
        file_name = file_path.split("/")[-1]
        eminemlyrics.append(
            {
                'Artist': "Eminem",
                'Song': file_name,
                'Lyrics': lyrics
            }
        )

eminemlyrics = pd.DataFrame(eminemlyrics)
```

In [12]:
```
combined_lyrics = [taylorswiftlyrics, eminemlyrics]

combined_lyrics_df = pd.concat(combined_lyrics)

combined_lyrics_df
```

Out[12]:

	Artist	Song	Lyrics
0	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	alittlemorelikeyou\n\nThink of something you c...
1	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	babyblue\n\nSmall town, big blue sky\nLittle b...
2	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	beautifuldays\n\nA moment just passed\nAnd I k...
3	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	fire\n\nAsking you questions is like asking fo...
4	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	halfwaytotexas\n\nWhen I got mad at you, I alw...
5	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	houstonrodeo\n\nShooting star flew across the ...
6	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	iusedtofly\n\nI... I used to fly\nDidn't matte...
7	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	kidinthecrowd\n\nLittle girl, she looks around...
8	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	luckyyou\n\nThere's a little girl in this litt...
9	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	maryjo\n\nThere's a little green house off a g...
10	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	myturntobeme\n\nSomething about me didn't fit ...
11	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	neverfade\n\nI think you fell off of your clou...
12	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	onesidedgoodbye\n\nYou said it, then you walke...
13	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	picturetoburn\n\nState the obvious, I didn't g...
14	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	pointofview\n\nTime is passing slowly for the ...
15	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	rideon\n\nThink of an angel tonight\nThink of ...
16	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	samegirl\n\nI'm still wearing my blue jeans\nN...
17	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	smokeyblacknights\n\nLet's watch\nThe tide cha...
18	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	teardropsonmyguitar\n\nDrew looks at me\nI fak...
19	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	timmcgraw\n\nHe said the way my blue eyes shin...
20	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	whydoyoutellme\n\nI tried to impress you\nIt d...
21	Taylor Swift	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	yourheartssomewhereelse\n\nI used to know ever...
0	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	313\n\nNow what you know about a sweet MC, fro...
1	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	backstabber\n\nAttention all units, attention ...
2	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	
3	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	ifihad\n\n"Life" by Marshall Mathers\nWhat is ...
4	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	infinite\n\nAw, yeah (It's like this, like thi...
5	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	introslimshady\n\n[Slim Shady:]\n"Eminem"\n[Em...

	Artist	Song	Lyrics
6	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	itsok\n\nCheck it out\nHey Kyu!\nIt's a broke ...
7	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	jealousywoesii\n\nJealous!\nJealous!\nJealous!...
8	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	justdontgiveafuck\n\nWhoa, a-get your hands in...
9	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	justthetwoofus\n\n(Just the two of us, just th...
10	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	lowdowndirty\n\n"I'm low down and dirty, but n...
11	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	maxine\n\nMaxine!\nHello\nCan I speak to Maxin...
12	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	mommyskit\n\n[*The sound of a body being dragg...
13	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	murdermurder\n\n"All I see is murder murder, m...
14	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	mynameis\n\n[Eminem:]\nHi, my name is, what? M...
15	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	never2far\n\nDenaun, what up, man?\nWhat up, d...
16	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	noonesillerthanme\n\nYeah... ha ha ha (Bang!)\nY...
17	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	openmic\n\nHey yo, what's up, man?\nHey yo, yo...
18	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	publicserviceannouncementskit\n\n[Jeff Bass:]\...
19	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	searchin\n\n[Angela Workman:]\nAin't no one sp...
20	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	tonight\n\nTonight, tonight, tonight, tonight,...
21	Eminem	C:\USD\MSADS\Spring 24\ADS 509\Module 1\Assign...	wegointerlude\n\nYeah, soul intent, checking i...

In [13]:
```
# Read in the twitter data
# change `data_location` to the location of the folder on your machine.
twitter_location = "C:/USD/MSADS/Spring 24/ADS 509/Module 2/Assignment/M1 Assignment Data/M1 Results/twitter/"

os.chdir(twitter_location)

twitter_files = os.listdir(twitter_location)
desc_files = [f for f in twitter_files if "followers_data" in f]
twitter_data = defaultdict(list)
for f in desc_files :
    artist = f.split("_")[0]

    with open(twitter_location + f,'r', encoding='utf8') as infile :
        next(infile)
        for idx, line in enumerate(infile.readlines()) :
            line = line.strip().split("\t")
            if len(line) == 7 :
                twitter_data[artist].append(line[6])
```

Data Cleaning

Now clean and tokenize your data. Remove punctuation chacters (available in the `punctuation` object in the `string` library), split on whitespace, fold to lowercase, and remove stopwords. Store your cleaned data, which must be accessible as an interable for `descriptive_stats`, in new objects or in new columns in your data frame.

In [14]:
```
punctuation = set(punctuation) # speeds up comparison
```

In [15]:
```
# create your clean twitter data here
def remove_stop(tokens):
    return [t for t in tokens if t.lower() not in stopwords]

# Reuse regex from tokenizeText Function
# Reuse the Transform function and the prepareData function

twitter_df = pd.DataFrame([(k, x) for k, v in twitter_data.items() for x in v],
                          columns=['artist','tweet'])
twitter_df['tokens'] = twitter_df['tweet'].apply(
    prepareData, transform=transform)
chertweets = twitter_df[twitter_df["artist"] == "cher"]
robyntweets = twitter_df[twitter_df["artist"] != "cher"]
```

In [16]:
```
# create your clean lyrics data here

combined_lyrics_df['tokens'] = combined_lyrics_df['Lyrics'].apply(
    prepareData, transform=transform)
taylorswiftlyrics['tokens'] = taylorswiftlyrics['Lyrics'].apply(
    prepareData, transform=transform)
eminemlyrics['tokens'] = eminemlyrics['Lyrics'].apply(
    prepareData, transform=transform)
```

Basic Descriptive Statistics

Call your `descriptive_stats` function on both your lyrics data and your twitter data and for both artists (four total calls).

In [17]:
```
# calls to descriptive_stats here
print("Taylor Swift Stats")
descriptive_stats(taylorswiftlyrics["tokens"])
print("\nEminem Stats")
descriptive_stats(eminemlyrics["tokens"])
print("\nCher Stats")
descriptive_stats(chertweets["tokens"])
print("\nRobyn Stats")
descriptive_stats(robyntweets["tokens"])
```

```
Taylor Swift Stats
There are 6323 tokens in the data.
There are 897 unique tokens in the data.
There are 23061 characters in the data.
The lexical diversity is 0.142 in the data.
The top 5 most common words are
[('i', 326), ('you', 285), ('the', 187), ('to', 164), ('and', 138)]

Eminem Stats
There are 14200 tokens in the data.
There are 2623 unique tokens in the data.
There are 52899 characters in the data.
The lexical diversity is 0.185 in the data.
The top 5 most common words are
[('i', 669), ('the', 463), ('you', 450), ('and', 359), ('a', 341)]

Cher Stats
There are 22645956 tokens in the data.
There are 988979 unique tokens in the data.
There are 106792530 characters in the data.
The lexical diversity is 0.044 in the data.
The top 5 most common words are
[('i', 600604), ('and', 575886), ('a', 424923), ('the', 408343), ('to', 351244)]

Robyn Stats
There are 2058267 tokens in the data.
There are 208184 unique tokens in the data.
There are 10230438 characters in the data.
The lexical diversity is 0.101 in the data.
The top 5 most common words are
[('i', 45376), ('and', 45049), ('the', 34843), ('a', 31096), ('of', 25970)]
```

Out[17]:
```
[2058267, 208184, 0.10114528387230617, 10230438]
```

Q: How do you think the "top 5 words" would be different if we left stopwords in the data?

A: If we kept common words like "the" and "and" in the data, they would likely dominate the "top 5 words" list, overshadowing more artist-specific or meaningful words.

Q: What were your prior beliefs about the lexical diversity between the artists? Does the difference (or lack thereof) in lexical diversity between the artists conform to your prior beliefs?

A: My expectations about how diverse an artist's lyrics are depended on their style. The observed differences in lexical diversity match what I anticipated. Artists with varied vocabulary and lyrical styles, like Taylor Swift, show higher diversity, while those with repetitive themes, like Cher, exhibit lower diversity.

Specialty Statistics

The descriptive statistics we have calculated are quite generic. You will now calculate a handful of statistics tailored to these data.

1. Ten most common emojis by artist in the twitter descriptions.
2. Ten most common hashtags by artist in the twitter descriptions.
3. Five most common words in song titles by artist.
4. For each artist, a histogram of song lengths (in terms of number of tokens)

We can use the `emoji` library to help us identify emojis and you have been given a function to help you.

In [18]:
```
assert(emoji.is_emoji("❤️"))
assert(not emoji.is_emoji(":-)"))
```

Emojis 😁

What are the ten most common emojis by artist in the twitter descriptions?

In [19]:
```
# Your code here
contains_emoji = []
emojis = []

for row in chertweets['tweet']:
    emoji_found = False
    for char in row:
        if emoji.is_emoji(char):
            emoji_found = True
            emojis.append(char)
    contains_emoji.append(emoji_found)

chertweets['has_emoji'] = contains_emoji
```

```
C:\Users\ebbi_\AppData\Local\Temp\ipykernel_1800\1524961650.py:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  chertweets['has_emoji'] = contains_emoji
```

In [20]:
```
cheremojis = emojis
cheremojis = pd.DataFrame(cheremojis, columns=['emojis'])
```

In [21]:
```
contains_emoji = []
emojis = []

for row in robyntweets['tweet']:
    emoji_found = False
    for char in row:
        if emoji.is_emoji(char):
            emoji_found = True
            emojis.append(char)
    contains_emoji.append(emoji_found)

robyntweets['has_emoji'] = contains_emoji
```

```
C:\Users\ebbi_\AppData\Local\Temp\ipykernel_1800\2943622266.py:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  robyntweets['has_emoji'] = contains_emoji
```

In [22]:
```
robynemojis = emojis
robynemojis = pd.DataFrame(robynemojis, columns=['emojis'])
```

In [23]:
```
n = 10
print("Top Cher Emojis")
print(cheremojis['emojis'].value_counts()[:n].index.tolist())
print("\n Top Robyn Emojis")
print(robynemojis['emojis'].value_counts()[:n].index.tolist())
```

```
Top Cher Emojis
['🖤', '🌈', '♥', '🤞', '✨', '💙', '🅱', '🌊', '✌', '💜']

 Top Robyn Emojis
['🖤', '🌈', '🤞', '♥', '✨', '🅱', '✌', '🅱', '♀', '💙']
```

Hashtags

What are the ten most common hashtags by artist in the twitter descriptions?

In [24]:
```
# Your code here
#Cher
cherhashtags = chertweets.tweet.str.findall(r'#.*?(?=\s|$)')
cherhashtags = cherhashtags.to_frame()
cherhashtags = cherhashtags[
    cherhashtags['tweet'].astype(str).str.contains("#")]
cherhashtags = cherhashtags.explode("tweet")
#Robyn
```

```
robynhashtags = robyntweets.tweet.str.findall(r'#.*?(?=\s|$)')
robynhashtags = robynhashtags.to_frame()
robynhashtags = robynhashtags[
    robynhashtags['tweet'].astype(str).str.contains("#")]

robynhashtags = robynhashtags.explode("tweet")
```

In [25]:
```
n = 10
print("Top Cher Hashtags")
print(cherhashtags['tweet'].value_counts()[:n].index.tolist())
print("\n Top Robyn Hashtags")
print(robynhashtags['tweet'].value_counts()[:n].index.tolist())
```

```
Top Cher Hashtags
['#BLM', '#Resist', '#BlackLivesMatter', '#resist', '#', '#FBR', '#TheResistance', '#blacklivesmatter', '#1', '#Resistance']

 Top Robyn Hashtags
['#BlackLivesMatter', '#BLM', '#', '#blacklivesmatter', '#1', '#music', '#Music', '#EDM', '#TeamFollowBack', '#blm']
```

Song Titles

What are the five most common words in song titles by artist? The song titles should be on the first line of the lyrics pages, so if you have kept the raw file contents around, you will not need to re-read the data.

In [26]:
```
# Your code here
os.chdir(data_location+lyrics_folder)

taylorswift_title_list = glob.glob(
    os.path.join(
        os.getcwd(), "taylorswift", "*.txt"))

taylorswiftsongtitles = []

for file_path in taylorswift_title_list:
    with open(file_path) as f:
        taylorswiftsongtitles.append(f.readline().strip('\n'))

taylorswiftsongtitles = pd.DataFrame(taylorswiftsongtitles, columns=['titles'])
```

In [27]:
```
# Your code here
os.chdir(data_location+lyrics_folder)

eminem_title_list = glob.glob(
    os.path.join(
        os.getcwd(), "eminem", "*.txt"))

eminemsongtitles = []
```

```
for file_path in eminem_title_list:
    with open(file_path) as f:
        eminemsongtitles.append(f.readline().strip('\n'))

eminemsongtitles = pd.DataFrame(eminemsongtitles, columns=['titles'])
```

In [28]:
```
taylorswiftsongtitles['tokens'] = taylorswiftsongtitles['titles'].apply(
    prepareData, transform=transform)
eminemsongtitles['tokens'] = eminemsongtitles['titles'].apply(
    prepareData, transform=transform)

n = 5
print("Top Talor Swift Title Words")
descriptive_stats(taylorswiftsongtitles["tokens"])
print("\n Top Eminem Title Words")
descriptive_stats(eminemsongtitles["tokens"])
```

```
Top Talor Swift Title Words
There are 22 tokens in the data.
There are 22 unique tokens in the data.
There are 262 characters in the data.
The lexical diversity is 1.000 in the data.
The top 5 most common words are
[('alittlemorelikeyou', 1), ('babyblue', 1), ('beautifuldays', 1), ('fire', 1), ('halfwaytotexas', 1)]

 Top Eminem Title Words
There are 21 tokens in the data.
There are 21 unique tokens in the data.
There are 229 characters in the data.
The lexical diversity is 1.000 in the data.
The top 5 most common words are
[('313', 1), ('backstabber', 1), ('ifihad', 1), ('infinite', 1), ('introslimshady', 1)]
```

Out[28]:
```
[21, 21, 1.0, 229]
```

Song Lengths

For each artist, a histogram of song lengths (in terms of number of tokens). If you put the song lengths in a data frame with an artist column, matplotlib will make the plotting quite easy. An example is given to help you out.

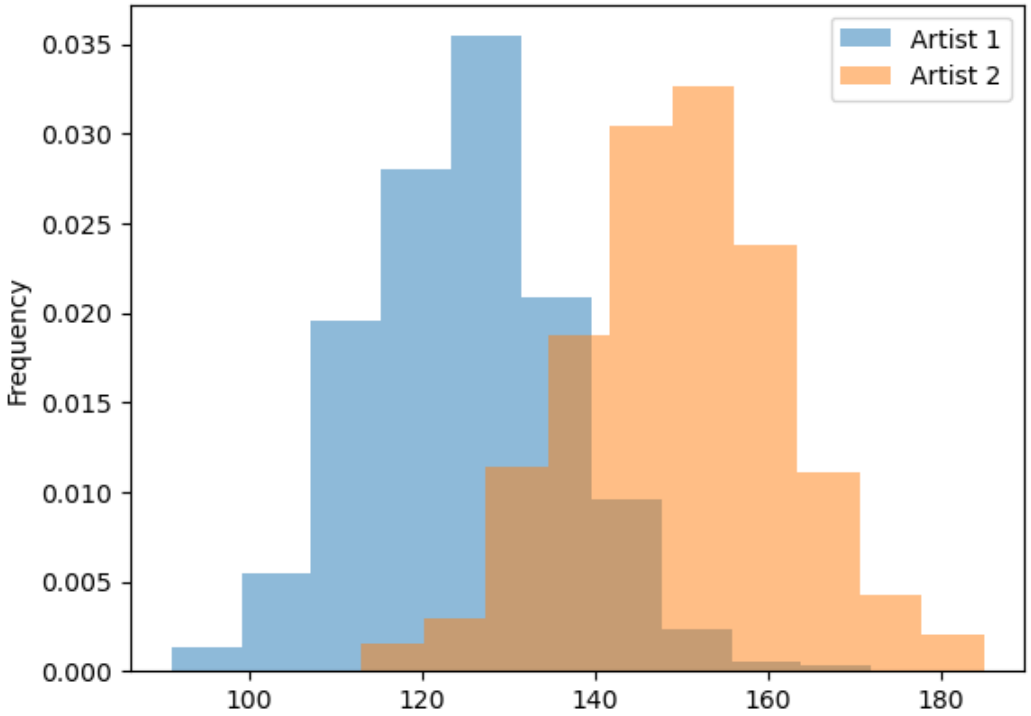
In [29]:
```
num_replicates = 1000

df = pd.DataFrame({
    "artist" : ['Artist 1'] * num_replicates + ['Artist 2']*num_replicates,
    "length" : np.concatenate((np.random.poisson(125,num_replicates),
                               np.random.poisson(150,num_replicates)))
})
```

```
df.groupby('artist')['length'].plot(
    kind="hist",density=True,alpha=0.5,legend=True)
```

Out[29]:

```
artist
Artist 1    Axes(0.125,0.11;0.775x0.77)
Artist 2    Axes(0.125,0.11;0.775x0.77)
Name: length, dtype: object
```



Since the lyrics may be stored with carriage returns or tabs, it may be useful to have a function that can collapse whitespace, using regular expressions, and be used for splitting.

Q: What does the regular expression `'\s+'` match on?

A: One or more whitespaces.

In [30]:

```
collapse_whitespace = re.compile(r'\s+')

def tokenize_lyrics(lyric) :
    """strip and split on whitespace"""
    return([item.lower() for item in collapse_whitespace.split(lyric)])
```

In [31]:
```
# Your lyric length comparison chart here.
combined_lyrics_df['length'] = combined_lyrics_df['Lyrics'].str.split().str.len()
```

In [32]:
```
combined_lyrics_df.groupby('Artist')['length'].plot(
    kind="hist",density=True,alpha=0.5,legend=True)
```

Out[32]:
```
Artist
Eminem          Axes(0.125,0.11;0.775x0.77)
Taylor Swift    Axes(0.125,0.11;0.775x0.77)
Name: length, dtype: object
```