misogyny-in-lyrics

April 28, 2025

[1]:
```
#Setting the working directory to the project folder, easily access Git
%cd "C:/Users/siand/Downloads/Exeter/BEE2041 Data Science/Empirical Project/
 ↪misogyny-in-lyrics/data/Raw"
```

```
C:\Users\siand\Downloads\Exeter\BEE2041 Data Science\Empirical Project\misogyny-
in-lyrics\data\Raw
```

[2]:
```
#Skip executing cells shortcut as don't want to repeat webscrape
from IPython.core.magic import register_cell_magic

@register_cell_magic
def skip(line, cell):
    return
```

1 Data Gathering

[3]:
```
import pandas as pd

df_rawdata = pd.read_csv("Full_Billboard_year_end_hot_100_USA.csv",␣
 ↪encoding='latin1')
print(len(df_rawdata))
```

```
6789
```

[4]:
```
#Filtering relevant 50 years + top 50 of each year using raw data into a␣
 ↪filtered df
df_filtered = df_rawdata[(df_rawdata['Year'] >= 1972) & (df_rawdata['Year'] <=␣
 ↪2021)].copy()

#Extracting numeric ranking from No. column due to formatting, convert to int
df_filtered['No.'] = df_filtered['No.'].astype(str).str.extract(r'(\d+)')
df_filtered['No.'] = df_filtered['No.'].astype(int)
df_filtered = df_filtered[df_filtered['No.'] <= 50]

df_filtered
```

[4]:
```
      No.                                Title  \
1789    1  The First Time Ever I Saw Your Face
```

```
1790    2              Alone Again (Naturally)
1791    3                          American Pie
1792    4                           Without You
1793    5                         The Candy Man
…     …                                     …
6734   46                         Need to Know
6735   47                       Wants and Needs
6736   48                            Way 2 Sexy
6737   49                            Telepatía
6738   50                               Whoopty

                                    Artist(s)  Year
1789                           Roberta Flack  1972
1790                     Gilbert O'Sullivan  1972
1791                             Don McLean  1972
1792                           Harry Nilsson  1972
1793                       Sammy Davis Jr.  1972
…                                        …   …
6734                              Doja Cat  2021
6735              Drake featuring Lil Baby  2021
6736  Drake featuring Future and Young Thug  2021
6737                            Kali Uchis  2021
6738                                    CJ  2021

[2500 rows x 4 columns]
```

[5]:
```
import lyricsgenius #Import to access Genius Lyrics API
import time

#Using Genius Access Token to set up the API client
genius = lyricsgenius.
 ↪Genius("FUkh9geZ4oUUMJkC2tTdDRl8PMjI-BCocha0AnShIdv-owl6OxhkhrQj9xijAK_F",
                            skip_non_songs=True,
)
```

[6]:
```
#Defining the function using Genius API to get lyrics
#In hindsight, the 'none' didn't work and we didn't automatically save it to a␣
 ↪csv etc, took over 1000 minutes
def get_lyrics(title, artist):
    try:
        song = genius.search_song(title, artist)
        return song.lyrics if song else None
    except Exception as e:
        print(f"Error fetching '{title}' by {artist}: {e}")
        return None
```

[7]:
```
%%skip
#Scrape the lyrics, showing progress, pause to avoid the API rate limits
lyrics = []

for i, row in df_rawdata.iterrows():
    title = row['Title']
    artist = row['Artist(s)']

    print(f"Fetching lyrics for: {title} by {artist}...")
    lyric = get_lyrics(title, artist)
    lyrics.append(lyric)

    time.sleep(1)
```

[8]:
```
#Loading the lyrics from the json file to a list as we are not running the␣
 ↪webscrape again to get the lyrics, surely there's another method?
import json
%cd "C:/Users/siand/Downloads/Exeter/BEE2041 Data Science/Empirical Project/
 ↪misogyny-in-lyrics/data/Processed"
#Load the dataset with lyrics already saved
df_json = pd.read_json("lyrics_1972_2021.json")
#Rebuild the lyrics list
lyrics = df_json["Lyrics"].tolist()
```

```
C:\Users\siand\Downloads\Exeter\BEE2041 Data Science\Empirical Project\misogyny-
in-lyrics\data\Processed
```

[9]:
```
df_withlyrics = df_filtered.copy()
df_withlyrics["Lyrics"] = lyrics

df_withlyrics.to_csv("lyrics_1972_2021.csv", index=False)
df_withlyrics.to_json("lyrics_1972_2021.json", orient="records",␣
 ↪force_ascii=False, indent=2)
```

[10]:
```
#Songs containing no lyrics - something is wrong after inspection
df_json.tail(250) #read the json filelyrics_1972_2021.json and confirms missing␣
 ↪last 5 years
```

[10]:

	No.	Title	\
2250	1	Shape of You	
2251	2	Despacito (Remix)	
2252	3	That's What I Like	
2253	4	Humble	
2254	5	Something Just Like This	
…	…	…	
2495	46	Need to Know	
2496	47	Wants and Needs	
2497	48	Way 2 Sexy	

```
2498   49                Telepatía
2499   50                   Whoopty

                                                 Artist(s)  Year Lyrics
2250                                           Ed Sheeran  2017   None
2251  Luis Fonsi and Daddy Yankee featuring Justin B…  2017   None
2252                                           Bruno Mars  2017   None
2253                                        Kendrick Lamar  2017   None
2254                         The Chainsmokers and Coldplay  2017   None
…                                                     …   …    …
2495                                           Doja Cat  2021   None
2496                              Drake featuring Lil Baby  2021   None
2497                 Drake featuring Future and Young Thug  2021   None
2498                                           Kali Uchis  2021   None
2499                                                   CJ  2021   None

[250 rows x 5 columns]
```

After realising years 2017-2021 were missing, reran a webscrape for this specific time frame (later realise it's because there were 250 previous missing songs and instead of 'None' being placed where there were no lyrics, it was filled in with the next songs lyrics, 250 times)

[11]:
```
%cd "C:/Users/siand/Downloads/Exeter/BEE2041 Data Science/Empirical Project/
 ↪misogyny-in-lyrics/data/Raw"

#Set up the Genius API client
genius = lyricsgenius.
 ↪Genius("9ai6TZ9Im0CTUKQp0mP13oC0fbgVlYiUlqoHhtn00cmecDo98cJ8hzVpdsd2rAku",
                              skip_non_songs=True,
                              timeout=15)

#Filter from 2017-2021
df_missing = (
    df_filtered[(df_filtered["Year"] >= 2017) & (df_filtered["Year"] <= 2021)]
    .groupby("Year", group_keys=False)
    .apply(lambda x: x.nsmallest(50, columns="No."))
    .reset_index(drop=True)
)

#Define scraping function
def get_lyrics(title, artist):
    try:
        song = genius.search_song(title, artist)
        return song.lyrics if song else None
    except Exception as e:
        print(f"Error fetching '{title}' by {artist}: {e}")
        return None
```

```
C:\Users\siand\Downloads\Exeter\BEE2041 Data Science\Empirical Project\misogyny-in-lyrics\data\Raw
```

[12]:
```
%%skip
#Scrape for 2017-2021
lyrics2 = []

for i, row in df_missing.iterrows():
    title = row["Title"]
    artist = row["Artist(s)"]
    print(f"Fetching lyrics for: {title} by {artist}...")

    lyric = get_lyrics(title, artist)
    lyrics2.append(lyric)

    time.sleep(1)

#Add lyrics back to dataframe
df_missing["Lyrics"] = lyrics2

# Save for merging later
df_missing.to_json("lyrics_2017_2021.json", orient="records", lines=False)␣
 ↪#saved to the original file = caused a lot of problems + duplicates.
```

[13]:
```
#Loading the lyrics from the json file to a list as we are not running the␣
 ↪webscrape again
%cd "C:/Users/siand/Downloads/Exeter/BEE2041 Data Science/Empirical Project/
 ↪misogyny-in-lyrics/data/Processed"
#Load the dataset with lyrics already saved
df_json2 = pd.read_json("lyrics_2017_2021.json")
#Rebuild the lyrics list
lyrics2 = df_json2["Lyrics"].tolist()
```

```
C:\Users\siand\Downloads\Exeter\BEE2041 Data Science\Empirical Project\misogyny-in-lyrics\data\Processed
```

[14]:
```
df_missing["Lyrics"] = lyrics2
```

[15]:
```
#Save to JSON and CSV - this was stupid, why use the same name?
df_missing.to_json("lyrics_2017_2021.json", orient="records", lines=False)
df_missing.to_csv("lyrics_2017_2021.csv", index=False)
```

[16]:
```
#Load original full dataset from first webscrape (1972-2016)
df_withlyrics = pd.read_json("lyrics_1972_2021.json")
```

[17]:
```
#Merge first and second web scrape to try and get all years
#Remove 2017-2021 rows from df_withlyrics
```

```
df_withlyrics_clean = df_withlyrics[~df_withlyrics["Year"].isin([2017, 2018,␣
↪2019, 2020, 2021])]

#Combine with the updated entries
df_full = pd.concat([df_withlyrics_clean, df_missing], ignore_index=True)

#Sort by Year and Ranking
df_full.sort_values(by=["Year", "No."], inplace=True)
df_full.reset_index(drop=True, inplace=True)

df_full.to_json("lyrics_2017_2021_full.json", orient="records", lines=False)
df_full
```

[17]:

	No.	Title	Artist(s)	Year	Lyrics
0	1	The First Time Ever I Saw Your Face	Roberta Flack	1972	16 ContributorsThe First Time Ever I Saw Your …
1	2	Alone Again (Naturally)	Gilbert O'Sullivan	1972	50 ContributorsAlone Again (Naturally) Lyrics[…
2	3	American Pie	Don McLean	1972	262 ContributorsTranslationsPolskiAmerican Pie…
3	4	Without You	Harry Nilsson	1972	32 ContributorsWithout You Lyrics[Verse 1]\nNo…
4	5	The Candy Man	Sammy Davis Jr.	1972	15 ContributorsThe Candy Man Lyrics[Intro]\n(C…
…	…	…	…	…	…
2495	46	Need to Know	Doja Cat	2021	169 ContributorsTranslationsTürkçeEspañol …
2496	47	Wants and Needs	Drake featuring Lil Baby	2021	
2497	48	Way 2 Sexy	Drake featuring Future and Young Thug	2021	
2498	49	Telepatía	Kali Uchis	2021	
2499	50	Whoopty	CJ	2021	

```
2496  267 ContributorsTranslationsTürkçe    Portu…
2497  2 ContributorsTranslationsEnglishTürkçeEspañol…
2498                                                 None
2499  124 ContributorsTranslationsDeutschWhoopty Lyr…

[2500 rows x 5 columns]
```

[18]:
```
#after inspection of what I though was a full df, I noticed the lyrics were␣
↪wrong at 22 and consequently everything shifted afterwards

df_full.head(30)
```

[18]:

	No.	Title	Artist(s)	Year
0	1	The First Time Ever I Saw Your Face	Roberta Flack	1972
1	2	Alone Again (Naturally)	Gilbert O'Sullivan	1972
2	3	American Pie		
3	4	Without You		
4	5	The Candy Man		
5	6	I Gotcha		
6	7	Lean on Me		
7	8	Baby, Don't Get Hooked on Me		
8	9	Brand New Key		
9	10	Daddy Don't You Walk So Fast		
10	11	Let's Stay Together		
11	12	Brandy (You're a Fine Girl)		
12	13	Oh Girl		
13	14	Nice to Be with You		
14	15	My Ding-a-Ling		
15	16	(If Loving You Is Wrong) I Don't Want to Be Right		
16	17	Heart of Gold		
17	18	Betcha by Golly, Wow		
18	19	I'll Take You There		
19	20	Ben		
20	21	The Lion Sleeps Tonight		
21	22	Outa-Space		
22	23	Slippin' into Darkness		
23	24	Long Cool Woman in a Black Dress		
24	25	How Do You Do		
25	26	Song Sung Blue		
26	27	A Horse with No Name		
27	28	Popcorn		
28	29	Everybody Plays the Fool		
29	30	Precious and Few		

2	Don McLean	1972
3	Harry Nilsson	1972
4	Sammy Davis Jr.	1972
5	Joe Tex	1972
6	Bill Withers	1972
7	Mac Davis	1972
8	Melanie	1972
9	Wayne Newton	1972
10	Al Green	1972
11	Looking Glass	1972
12	The Chi-Lites	1972
13	Gallery	1972
14	Chuck Berry	1972
15	Luther Ingram	1972
16	Neil Young	1972
17	The Stylistics	1972
18	The Staple Singers	1972
19	Michael Jackson	1972
20	Robert John	1972
21	Billy Preston	1972
22	War	1972
23	The Hollies	1972
24	Mouth & MacNeal	1972
25	Neil Diamond	1972
26	America	1972
27	Hot Butter	1972
28	The Main Ingredient	1972
29	Climax	1972

	Lyrics
0	16 ContributorsThe First Time Ever I Saw Your …
1	50 ContributorsAlone Again (Naturally) Lyrics[…
2	262 ContributorsTranslationsPolskiAmerican Pie…
3	32 ContributorsWithout You Lyrics[Verse 1]\nNo…
4	15 ContributorsThe Candy Man Lyrics[Intro]\n(C…
5	7 ContributorsI Gotcha Lyrics[Chorus]\nI gotch…
6	46 ContributorsLean on Me Lyrics[Intro]\nHmm\n…
7	7 ContributorsBaby, Don’t Get Hooked on Me Lyr…
8	21 ContributorsBrand New Key Lyrics[Verse 1]\n…
9	5 ContributorsDaddy, Don’t You Walk So Fast Ly…
10	44 ContributorsTranslationsEnglishLet’s Stay T…
11	38 ContributorsBrandy (You’re a Fine Girl) Lyr…
12	13 ContributorsOh Girl Lyrics[Verse 1]\nOh, gi…
13	7 ContributorsNice to Be with You Lyrics[Choru…
14	25 ContributorsMy Ding-a-Ling Lyrics[Spoken in…
15	14 Contributors(If Loving You Is Wrong) I Don’…
16	36 ContributorsHeart of Gold Lyrics[Intro]\n\n…

```
17  11 ContributorsBetcha By Golly, Wow Lyrics[Ver…
18  28 ContributorsI'll Take You There Lyrics[Intr…
19  42 ContributorsTranslationsPolskiBen Lyrics[Ve…
20  6 ContributorsThe Lion Sleeps Tonight LyricsWe…
21  26 ContributorsSlippin' Into Darkness Lyrics[V…
22  31 ContributorsLong Cool Woman (In a Black Dre…
23  5 ContributorsTranslationsEspañolEspañolEspaño…
24  8 ContributorsSong Sung Blue LyricsSong sung b…
25  99 ContributorsTranslationsPortuguêsA Horse Wi…
26  13 ContributorsEverybody Plays the Fool Lyrics…
27  6 ContributorsPrecious and Few LyricsPrecious …
28  6 Contributors(Last Night) I Didn't Get to Sle…
29  52 ContributorsTranslationsDeutschNights in Wh…
```

[19]:
```
%%skip
%cd "C:/Users/siand/Downloads/Exeter/BEE2041 Data Science/Empirical Project/
 ↪misogyny-in-lyrics/data/Raw"
#Attempting to use fuzzymatching on the json file with the shifted lyrics, it␣
 ↪did not work
from fuzzywuzzy import fuzz

df_all = pd.read_csv("Full_Billboard_year_end_hot_100_USA.csv",␣
 ↪encoding="latin1")
df_all["No."] = pd.to_numeric(df_all["No."], errors="coerce")
df_all = df_all.dropna(subset=["No."])
df_all["No."] = df_all["No."].astype(int)

df_expected = df_all[(df_all["Year"] >= 1972) & (df_all["Year"] <= 2016) &␣
 ↪(df_all["No."] <= 50)].reset_index(drop=True)

#Load scraped lyrics
with open("lyrics_1972_2021_full.json", "r", encoding="utf-8") as f:
    lyrics_scraped = json.load(f)

#Convert to DataFrame
df_lyrics = pd.DataFrame(lyrics_scraped)

#Clean string fields for comparison
def clean_text(text):
    return str(text).lower().strip()

df_expected["match_key"] = df_expected["Title"].map(clean_text) + " - " +␣
 ↪df_expected["Artist(s)"].map(clean_text)
df_lyrics["match_key"] = df_lyrics["Title"].map(clean_text) + " - " +␣
 ↪df_lyrics["Artist(s)"].map(clean_text)

#Fuzzy match lyrics to expected songs
```

```
from tqdm import tqdm
tqdm.pandas()

def find_best_lyric(title_key):
    best_score = 0
    best_lyric = None
    for _, row in df_lyrics.iterrows():
        score = fuzz.partial_ratio(title_key, row["match_key"])
        if score > best_score and score > 90:
            best_score = score
            best_lyric = row.get("Lyrics", None)
    return best_lyric if best_lyric else "None"

#Apply matching
df_expected["Lyrics"] = df_expected["match_key"].progress_apply(find_best_lyric)

#df_expected.to_json("lyrics_1972_2016_matched.json", orient="records",␣
 ↪force_ascii=False, indent=2) #have now deleted this JSON file as all it did␣
 ↪was in the lyrics column add all prev info e.g. title, artist, year, rank

df_merged = df_expected.merge(df_log, on=["Title", "Artist(s)"], how="left")
df_merged.head(30)
```

Went back to the original web scrape to further diagnose the issue, realised we could save the output as a txt file and decided to inspect. When inspecting, noticed the scrape specified 'Error fetching...” and “Rejecting...” so I thought if I could find the songs attached to this I could identify which songs I could shift the lyrics downwards and replace with “None”.

[ ]:
```
%cd "C:/Users/siand/Downloads/Exeter/BEE2041 Data Science/Empirical Project/
 ↪misogyny-in-lyrics/data/Raw"
import regex as re
#Load the log file
with open("all_lyrics.txt", "r", encoding="utf-8") as f:
    log = f.read()

#Get all "Fetching lyrics for..." lines
fetch_lines = re.findall(r"Fetching lyrics for: (.+?) by (.+?)\.\.\.", log)

#Get all following lines to check status
statuses = re.findall(r"(?:Done\.|Specified song does not contain lyrics\.␣
 ↪Rejecting\.|Error.*)", log)

#Combine into a list of dictionaries
log_data = []
for (title, artist), status in zip(fetch_lines, statuses):
    log_data.append({
        "Title": title.strip(),
```

```
        "Artist(s)": artist.strip(),
        "Status": status.strip()
    })

#Convert to DataFrame
df_log = pd.DataFrame(log_data)

df_log
```

```
C:\Users\siand\Downloads\Exeter\BEE2041 Data Science\Empirical Project\misogyny-in-lyrics\data\Raw
```

[ ]:

	Title	Artist(s)	Status
0	The First Time Ever I Saw Your Face	Roberta Flack	Done.
1	Alone Again (Naturally)	Gilbert O'Sullivan	Done.
2	American Pie	Don McLean	Done.
3	Without You	Harry Nilsson	Done.
4	The Candy Man	Sammy Davis Jr	Done.
…	…	… …	
2450	Levitating	Dua Lipa	Done.
2451	Save Your Tears	The Weeknd and Ariana Grande	Done.
2452	Blinding Lights	The Weeknd	Done.
2453	Mood	24kGoldn featuring Iann Dior	Done.
2454	Good 4 U	Olivia Rodrigo	Done.

[2455 rows x 3 columns]

[ ]:
```
#Another attempt at using the all_lyrics.txt to identify songs where the scrape␣
 ↪failed and to move them, once again did not work.
#Load lyrics from JSON (misaligned due to previous scrape)
with open("lyrics_1972_2021_full.json", "r") as f:
    lyrics_data = json.load(f)

#Load lines from text file where scrape errors were logged
with open("all_lyrics.txt", "r") as f:
    log_lines = f.readlines()

#Detect error or rejected lines
error_lines = [line.strip() for line in log_lines if "Error fetching" in line.
 ↪lower() or "Rejecting." in line.lower()]

#Extract titles from the error lines
error_titles = []
for line in error_lines:
    parts = line.split("title=")
    if len(parts) > 1:
        title_part = parts[1].split("',")[0].strip()
```

```
        error_titles.append(title_part.lower())

#Create a fixed lyrics list
fixed_lyrics = []
i = 0
for _, row in df_filtered.iterrows():
    title = row["Title"].strip().lower()

    if title in error_titles:
        fixed_lyrics.append("None")
    else:
        if i < len(lyrics_data):
            fixed_lyrics.append(lyrics_data[i]["Lyrics"])
            i += 1
        else:
            fixed_lyrics.append("None")

#Add the corrected lyrics to the DataFrame
#df_filtered["Lyrics"] = fixed_lyrics

#Save the cleaned, realigned dataset
#df_filtered.to_json("lyrics_aligned_1972_2021.json", orient="records",␣
 ↪indent=2) #have now deleted this json
```

Uploaded the txt file to chatgpt, asked it to cross identify songs with the JSON file, incrementally identify songs where the txt file said error/rejecting, shift the lyrics downwards to the correct song, and replace the lyrics with the word NONE and then i noticed duplicates due to an error merging so also asked it to de-dupe. After a few minutes I inspected the file and was very impressed it had worked! "perfect_lyrics_1st2nd", orient="records", indent=2

After briefly looking at our sparkling newly aligned JSON file, I was suspicious about the amount of NONEs I was seeing. Counted, 211 missing with it skewed towards the more recent years

[22]:
```
#Load the deduplicated JSON
with open("perfect_lyrics_1st2nd.json", "r", encoding="utf-8") as f:
    data = json.load(f)

#Convert to DataFrame
df = pd.DataFrame(data)

#Count total missing values
null_count = df["Lyrics"].isnull().sum()
none_count = (df["Lyrics"] == "None").sum()
total_missing = null_count + none_count
print(f"Total missing lyrics: {total_missing}")

#Count missing per year
df["Year"] = df["Year"].astype(int)
```

```
missing_per_year = df[df["Lyrics"].isnull() | (df["Lyrics"] == "None")].
  ↪groupby("Year").size()
print("\nMissing lyrics per year:\n", missing_per_year)
```

```
Total missing lyrics: 211

Missing lyrics per year:
 Year
1972     3
1973     1
1974     2
1976     4
1977     2
1978     3
1979     1
1980     2
1981     2
1982     1
1983     4
1984     2
1985     3
1986     3
1988     4
1989     2
1990     2
1992     7
1993     1
1994     1
1995     4
1996     1
1997     4
1998     2
1999     1
2000     3
2001     6
2002     7
2003     2
2004     7
2005     4
2006     1
2007     4
2008     2
2009     3
2010     7
2011     4
2012     2
2013    10
2014    11
```

```
2015     6
2016    14
2017    10
2018    19
2019     3
2020     7
2021    17
dtype: int64
```

Had a look at the song misisng, and turned them into a CSV because I was not satisfied with the missing songs that I knew were available, especially as I checked the all_lyrics.txt and a fair few errors were not due to the song not being found but due to other issues e.g. timeout

[23]:
```
#Load deduplicated file
with open("perfect_lyrics_1st2nd.json", "r", encoding="utf-8") as f:
    data = json.load(f)

df = pd.DataFrame(data)

#Filter only missing lyrics
missing_lyrics = df[df["Lyrics"].isnull() | (df["Lyrics"] == "None")]

#Extract title and artist
missing_pairs = missing_lyrics[["Title", "Artist(s)", "Year"]].
 ↪drop_duplicates().values.tolist()

#Save to CSV or just loop over in scrape
pd.DataFrame(missing_pairs, columns=["Title", "Artist(s)", "Year"]).
 ↪to_csv("missing_lyrics_to_scrape.csv", index=False)

missing_pairs
```

[23]:
```
[['Outa-Space', 'Billy Preston', 1972],
 ['Popcorn', 'Hot Butter', 1972],
 ['Scorpio', 'Dennis Coffey', 1972],
 ['Frankenstein', 'The Edgar Winter Group', 1973],
 ["Love's Theme", 'Love Unlimited Orchestra', 1974],
 ['The Entertainer', 'Marvin Hamlisch', 1974],
 ['Love Is Alive', 'Gary Wright', 1976],
 ['A Fifth of Beethoven', 'Walter Murphy & The Big Apple Band', 1976],
 ['Bohemian Rhapsody', 'Queen', 1976],
 ['Theme from S.W.A.T.', 'Rhythm Heritage', 1976],
 ['Evergreen (Love Theme from A Star Is Born)', 'Barbra Streisand', 1977],
 ['Angel in Your Arms', 'Hot', 1977],
 ["Stayin' Alive", 'Bee Gees', 1978],
 ['Feels So Good', 'Chuck Mangione', 1978],
 ['We Will Rock You/We Are the Champions', 'Queen', 1978],
 ['Music Box Dancer', 'Frank Mills', 1979],
```

```
['Sexy Eyes', 'Dr. Hook', 1980],
['Steal Away', 'Robbie Dupree', 1980],
['Just the Two of Us', 'Grover Washington, Jr. & Bill Withers', 1981],
['Love on the Rocks', 'Neil Diamond', 1981],
['Chariots of Fire', 'Vangelis', 1982],
['Every Breath You Take', 'The Police', 1983],
['Billie Jean', 'Michael Jackson', 1983],
['Beat It', 'Michael Jackson', 1983],
['Sweet Dreams (Are Made of This)', 'Eurythmics', 1983],
['Girls Just Want to Have Fun', 'Cyndi Lauper', 1984],
['Time After Time', 'Cyndi Lauper', 1984],
['Everybody Wants to Rule the World', 'Tears for Fears', 1985],
['Miami Vice Theme', 'Jan Hammer', 1985],
['Separate Lives', 'Phil Collins and Marilyn Martin', 1985],
['West End Girls', 'Pet Shop Boys', 1986],
['Alive and Kicking', 'Simple Minds', 1986],
['Live to Tell', 'Madonna', 1986],
["Sweet Child o' Mine", "Guns N' Roses", 1988],
['Man in the Mirror', 'Michael Jackson', 1988],
['Kokomo', 'The Beach Boys', 1988],
["I Don't Wanna Live Without Your Love", 'Chicago', 1988],
['Rock On', 'Michael Damian', 1989],
['Real Love', 'Jody Watley', 1989],
['How Am I Supposed to Live Without You', 'Michael Bolton', 1990],
["(Can't Live Without Your) Love and Affection", 'Nelson', 1990],
['Baby Got Back', 'Sir Mix-a-Lot', 1992],
["I'm Too Sexy", 'Right Said Fred', 1992],
['Black or White', 'Michael Jackson', 1992],
['Jump Around', 'House of Pain', 1992],
['Live and Learn', 'Joe Public', 1992],
['Smells Like Teen Spirit', 'Nirvana', 1992],
['Bohemian Rhapsody', 'Queen', 1992],
["Nuthin' but a 'G' Thang", 'Dr. Dre', 1993],
['Any Time, Any Place" / "And On and On', 'Janet Jackson', 1994],
['Kiss from a Rose', 'Seal', 1995],
['I Know', 'Dionne Farris', 1995],
['Water Runs Dry', 'Boyz II Men', 1995],
['I Got 5 on It', 'Luniz', 1995],
['Tha Crossroads', 'Bone Thugs-n-Harmony', 1996],
['How Do I Live', 'LeAnn Rimes', 1997],
['G.H.E.T.T.O.U.T.', 'Changing Faces', 1997],
['Honey', 'Mariah Carey', 1997],
['Not Tonight',
 "Lil' Kim featuring Da Brat, Left Eye, Missy Elliott and Angie Martinez",
 1997],
['The Boy Is Mine', 'Brandy and Monica', 1998],
['How Do I Live', 'LeAnn Rimes', 1998],
```

```
['All Star', 'Smash Mouth', 1999],
['Maria Maria', 'Santana featuring The Product G&B', 2000],
['I Wanna Know', 'Joe', 2000],
['Hot Boyz', 'Missy Elliott featuring Nas, Eve and Q-Tip', 2000],
["I'm Real (Murder Remix)", 'Jennifer Lopez featuring Ja Rule', 2001],
['Stutter', 'Joe featuring Mystikal', 2001],
['Ms. Jackson', 'Outkast', 2001],
['Put It on Me', "Ja Rule featuring Lil' Mo and Vita", 2001],
['Someone to Call My Lover', 'Janet Jackson', 2001],
['Fiesta', 'R. Kelly featuring Jay-Z', 2001],
['In the End', 'Linkin Park', 2002],
["Ain't It Funny (Murder Remix)", 'Jennifer Lopez featuring Ja Rule', 2002],
["I'm Gonna Be Alright", 'Jennifer Lopez featuring Nas', 2002],
['Heaven', 'DJ Sammy featuring Yanou and Do', 2002],
['Hey Baby', 'No Doubt featuring Bounty Killer', 2002],
['Girlfriend', "'N Sync featuring Nelly", 2002],
['Addictive', 'Truth Hurts featuring Rakim', 2002],
['In da Club', '50 Cent', 2003],
['Into You', 'Fabolous featuring Tamia', 2003],
['Turn Me On', 'Kevin Lyttle featuring Spragga Benz', 2004],
['Numb', 'Linkin Park', 2004],
['Jesus Walks', 'Kanye West', 2004],
['Locked Up', 'Akon featuring Styles P', 2004],
['Stand Up', 'Ludacris featuring Shawnna', 2004],
['All Falls Down', 'Kanye West featuring Syleena Johnson', 2004],
['Toxic', 'Britney Spears', 2004],
['Boulevard of Broken Dreams', 'Green Day', 2005],
["Drop It Like It's Hot", 'Snoop Dogg featuring Pharrell', 2005],
['Hate It or Love It', 'The Game featuring 50 Cent', 2005],
['Feel Good Inc.', 'Gorillaz', 2005],
['Crazy', 'Gnarls Barkley', 2006],
['Umbrella', 'Rihanna featuring Jay-Z', 2007],
['Stronger', 'Kanye West', 2007],
['Rockstar', 'Nickelback', 2007],
["I'm a Flirt", 'R. Kelly featuring T.I. and T-Pain', 2007],
['Live Your Life', 'T.I. featuring Rihanna', 2008],
['A Milli', 'Lil Wayne', 2008],
['Poker Face', 'Lady Gaga', 2009],
['I Gotta Feeling', 'The Black Eyed Peas', 2009],
['Live Your Life', 'T.I. featuring Rihanna', 2009],
['Bad Romance', 'Lady Gaga', 2010],
['Dynamite', 'Taio Cruz', 2010],
['I Gotta Feeling', 'The Black Eyed Peas', 2010],
['Fireflies', 'Owl City', 2010],
['Like a G6', 'Far East Movement featuring The Cataracs and Dev', 2010],
['Bulletproof', 'La Roux', 2010],
["Live Like We're Dying", 'Kris Allen', 2010],
```

```
['E.T.', 'Katy Perry featuring Kanye West', 2011],
['Last Friday Night (T.G.I.F.)', 'Katy Perry', 2011],
['Born This Way', 'Lady Gaga', 2011],
['Dynamite', 'Taio Cruz', 2011],
['The One That Got Away', 'Katy Perry', 2012],
['Gangnam Style', 'Psy', 2012],
["Can't Hold Us", 'Macklemore & Ryan Lewis featuring Ray Dalton', 2013],
['Cruise', 'Florida Georgia Line featuring Nelly', 2013],
['Royals', 'Lorde', 2013],
['Wrecking Ball', 'Miley Cyrus', 2013],
['Feel This Moment', 'Pitbull featuring Christina Aguilera', 2013],
['Summertime Sadness', 'Lana Del Rey and Cédric Gervais', 2013],
["It's Time", 'Imagine Dragons', 2013],
['Power Trip', 'J. Cole featuring Miguel', 2013],
['Girl on Fire', 'Alicia Keys featuring Nicki Minaj', 2013],
['Heart Attack', 'Demi Lovato', 2013],
['Happy', 'Pharrell Williams', 2014],
['Dark Horse', 'Katy Perry featuring Juicy J', 2014],
['All of Me', 'John Legend', 2014],
['Fancy', 'Iggy Azalea featuring Charli XCX', 2014],
['Counting Stars', 'OneRepublic', 2014],
['Talk Dirty', 'Jason Derulo featuring 2 Chainz', 2014],
['Rude', 'Magic!', 2014],
['All About That Bass', 'Meghan Trainor', 2014],
['Problem', 'Ariana Grande featuring Iggy Azalea', 2014],
['Royals', 'Lorde', 2014],
['Wrecking Ball', 'Miley Cyrus', 2014],
['Cheerleader (Felix Jaehn Remix)', 'OMI', 2015],
['All About That Bass', 'Meghan Trainor', 2015],
['Hotline Bling', 'Drake', 2015],
['Hello', 'Adele', 2015],
['Stitches', 'Shawn Mendes', 2015],
['Time of Our Lives', 'Pitbull and Ne-Yo', 2015],
['Stressed Out', 'Twenty One Pilots', 2016],
['Hello', 'Adele', 2016],
['Cheap Thrills', 'Sia featuring Sean Paul', 2016],
['Needed Me', 'Rihanna', 2016],
['My House', 'Flo Rida', 2016],
['I Took a Pill in Ibiza', 'Mike Posner', 2016],
['Work from Home', 'Fifth Harmony featuring Ty Dolla Sign', 2016],
['This Is What You Came For', 'Calvin Harris featuring Rihanna', 2016],
['Me, Myself & I', 'G-Eazy and Bebe Rexha', 2016],
['Ride', 'Twenty One Pilots', 2016],
['Heathens', 'Twenty One Pilots', 2016],
['Pillowtalk', 'Zayn', 2016],
['Stitches', 'Shawn Mendes', 2016],
['Hotline Bling', 'Drake', 2016],
```

```
['Shape of You', 'Ed Sheeran', 2017],
['Despacito (Remix)',
 'Luis Fonsi and Daddy Yankee featuring Justin Bieber',
 2017],
['Humble', 'Kendrick Lamar', 2017],
['Bad and Boujee', 'Migos featuring Lil Uzi Vert', 2017],
['XO Tour Llif3', 'Lil Uzi Vert', 2017],
['Mask Off', 'Future', 2017],
["I Don't Wanna Live Forever", 'Zayn and Taylor Swift', 2017],
['Love on the Brain', 'Rihanna', 2017],
['Feel It Still', 'Portugal. The Man', 2017],
['Mi Gente', 'J Balvin and Willy William featuring Beyoncé', 2017],
["God's Plan", 'Drake', 2018],
['In My Feelings', 'Drake', 2018],
['Girls Like You', 'Maroon 5 featuring Cardi B', 2018],
['Lucid Dreams', 'Juice Wrld', 2018],
['Finesse', 'Bruno Mars featuring Cardi B', 2018],
['Sad!', 'XXXTentacion', 2018],
['Look Alive', 'BlocBoy JB featuring Drake', 2018],
['Yes Indeed', 'Lil Baby and Drake', 2018],
['Friends', 'Marshmello and Anne-Marie', 2018],
['Bad at Love', 'Halsey', 2018],
['Taste', 'Tyga featuring Offset', 2018],
['Let You Down', 'NF', 2018],
['No Limit', 'G-Eazy featuring ASAP Rocky and Cardi B', 2018],
['Fefe', '6ix9ine featuring Nicki Minaj and Murda Beatz', 2018],
['Tequila', 'Dan + Shay', 2018],
['Feel It Still', 'Portugal. The Man', 2018],
['MotorSport', 'Migos, Nicki Minaj and Cardi B', 2018],
['I Like Me Better', 'Lauv', 2018],
['Gucci Gang', 'Lil Pump', 2018],
['7 Rings', 'Ariana Grande', 2019],
['Girls Like You', 'Maroon 5 featuring Cardi B', 2019],
['Lucid Dreams', 'Juice Wrld', 2019],
['Blinding Lights', 'The Weeknd', 2020],
['I Hope', 'Gabby Barrett featuring Charlie Puth', 2020],
['Whats Poppin',
 'Jack Harlow featuring DaBaby, Tory Lanez and Lil Wayne',
 2020],
['Savage', 'Megan Thee Stallion featuring Beyoncé', 2020],
['Roses (Imanbek Remix)', 'Saint Jhn', 2020],
['Lose You to Love Me', 'Selena Gomez', 2020],
['Break My Heart', 'Dua Lipa', 2020],
['Save Your Tears', 'The Weeknd and Ariana Grande', 2021],
['Blinding Lights', 'The Weeknd', 2021],
['Kiss Me More', 'Doja Cat featuring SZA', 2021],
['Leave the Door Open', 'Silk Sonic (Bruno Mars and Anderson .Paak)', 2021],
```

```
 ['Drivers License', 'Olivia Rodrigo', 2021],
 ['Peaches', 'Justin Bieber featuring Daniel Caesar and Giveon', 2021],
 ['Butter', 'BTS', 2021],
 ['Stay', 'The Kid Laroi and Justin Bieber', 2021],
 ['Deja Vu', 'Olivia Rodrigo', 2021],
 ['Positions', 'Ariana Grande', 2021],
 ['Bad Habits', 'Ed Sheeran', 2021],
 ['Heat Waves', 'Glass Animals', 2021],
 ['Without You', 'The Kid Laroi', 2021],
 ['Forever After All', 'Luke Combs', 2021],
 ['34+35', 'Ariana Grande featuring Doja Cat and Megan Thee Stallion', 2021],
 ['I Hope', 'Gabby Barrett featuring Charlie Puth', 2021],
 ['Way 2 Sexy', 'Drake featuring Future and Young Thug', 2021]]
```

[24]:
```
%%skip
from rapidfuzz import fuzz

#Set up Genius API
genius = lyricsgenius.
 ↪Genius("20Y2IB47Ha-FxWMhpV6f26xiH7r-wnLDyWnXaVIGUnSAgCybJTw5TpuQubs9eegv",␣
 ↪skip_non_songs=True, remove_section_headers=True, timeout=10)
genius.verbose = False  # Hide output

#Threshold for fuzzy title matching
FUZZY_THRESHOLD = 85

#Load the missing lyrics CSV
df = pd.read_csv("missing_lyrics_to_scrape.csv")

#Create a new column for the lyrics
lyrics_list = []

#Loop through each missing entry
for idx, row in df.iterrows():
    title = str(row["Title"])
    artist = str(row["Artist(s)"])
    year = int(row["Year"])
    print(f" {year} - Searching for: {title} by {artist}")

    try:
        song = genius.search_song(title, artist)
    except Exception as e:
        print("Genius API failed:", e)
        lyrics_list.append("None")
        continue

    if song is None:
```

```
        print("Exact match not found, trying fuzzy fallback...")
        lyrics_list.append("None")
        continue

    #Fuzzy match verification (score + lyrics content)
    score = fuzz.ratio(song.title.lower(), title.lower())
    if score > FUZZY_THRESHOLD or title.lower() in song.lyrics.lower():
        print(f"Accepted (score: {score})")
        lyrics_list.append(song.lyrics)
    else:
        print(f"Rejected (score: {score})")
        lyrics_list.append("None")

    time.sleep(1)

#Save back to CSV
df["Lyrics"] = lyrics_list
df.to_csv("missing_lyrics_to_scrape_filled.csv", index=False)
df.to_json("missing_lyrics_to_scrape_filled.json", index=False)
print("Updated CSV saved.")
```

Merged the missing lyrics with previous lyrics. Realised I missed a certain phrase 'no results found' in the original webscrape which accounts for now about 40 songs, at the worst, the lyrics are approx 20 places off. Unhappy but have easily spent over 15 hours trying to fix it and think this is the closest we will get.

[25]:
```
df_nearlyfinal = pd.read_json("MOST ACCURATE.json")
df_nearlyfinal
```

[25]:

	No.	Title	Artist(s)	Year
0	1	The First Time Ever I Saw Your Face	Roberta Flack	1972
1	2	Alone Again (Naturally)	Gilbert O'Sullivan	1972
2	3	American Pie	Don McLean	1972
3	4	Without You	Harry Nilsson	1972
4	5	The Candy Man	Sammy Davis Jr.	1972
…	…	…		
2495	46	Need to Know		
2496	47	Wants and Needs		
2497	48	Way 2 Sexy		
2498	49	Telepatía		
2499	50	Whoopty		

```
…                                      …   …
2495                                Doja Cat  2021
2496               Drake featuring Lil Baby  2021
2497  Drake featuring Future and Young Thug  2021
2498                              Kali Uchis  2021
2499                                      CJ  2021

                                                 Lyrics
0     16 ContributorsThe First Time Ever I Saw Your …
1     50 ContributorsAlone Again (Naturally) Lyrics[…
2     262 ContributorsTranslationsPolskiAmerican Pie…
3     32 ContributorsWithout You Lyrics[Verse 1]\nNo…
4     15 ContributorsThe Candy Man Lyrics[Intro]\n(C…
…                                                   …
2495  169 ContributorsTranslationsTürkçeEspañol   …
2496  267 ContributorsTranslationsTürkçe    Portu…
2497  2 ContributorsTranslationsEnglishTürkçeEspañol…
2498                                             None
2499  124 ContributorsTranslationsDeutschWhoopty Lyr…

[2500 rows x 5 columns]
```

[ ]:
```
null_count = df_nearlyfinal["Lyrics"].isnull().sum()
none_count = (df_nearlyfinal["Lyrics"] == "None").sum()
total_missing = null_count + none_count
print(f"Total missing lyrics: {total_missing}")

#Count missing per year
df_nearlyfinal["Year"] = df_nearlyfinal["Year"].astype(int)
missing_per_year = df_nearlyfinal[df_nearlyfinal["Lyrics"].isnull() |␣
 ↪(df_nearlyfinal["Lyrics"] == "None")].groupby("Year").size()
print("\nMissing lyrics per year:\n", missing_per_year)
```

```
Total missing lyrics: 188

Missing lyrics per year:
 Year
1972    3
1973    1
1974    2
1975    2
1976    6
1977    2
1978    4
1979    1
1980    6
1981    2
1982    2
```

```
1983     4
1984     2
1985     3
1986     4
1988     4
1989     1
1990     4
1992     7
1993     2
1994     4
1995     4
1996     6
1997     6
1998     6
1999     2
2000     3
2001     4
2002     6
2003     7
2004     1
2005     8
2006     6
2007     2
2008     3
2009     7
2010     3
2011     9
2012     4
2013     5
2014    14
2015     4
2016     8
2019     1
2020     2
2021     1
dtype: int64
```

[27]:
```
#Not content with the large number of missing songs in recent years, copied the␣
 ↪smaller webscrape from above to re-do it and gather more songs
df_2000s = df_nearlyfinal[(df_nearlyfinal["Year"] >= 2000) &␣
 ↪(df_nearlyfinal["Year"] <= 2016)]

#Filter only missing lyrics
missing_lyrics = df_2000s[df_2000s["Lyrics"].isnull() | (df_2000s["Lyrics"] ==␣
 ↪"None")]

#Extract title and artist
```

```
missing_pairs = missing_lyrics[["Title", "Artist(s)", "Year"]].
 ↪drop_duplicates().values.tolist()

#Save to CSV or just loop over
pd.DataFrame(missing_pairs, columns=["Title", "Artist(s)", "Year"]).
 ↪to_csv("missing_2000slyrics_to_scrape.csv", index=False)

missing_pairs
```

[27]:
```
[['Music', 'Madonna', 2000],
 ['I Try', 'Macy Gray', 2000],
 ["It's Gonna Be Me", "'N Sync", 2000],
 ['He Loves U Not', 'Dream', 2001],
 ['Crazy', 'K-Ci & JoJo', 2001],
 ['With Arms Wide Open', 'Creed', 2001],
 ['Missing You', 'Case', 2001],
 ['Dilemma', 'Nelly featuring Kelly Rowland', 2002],
 ['Always on Time', 'Ja Rule featuring Ashanti', 2002],
 ["Gangsta Lovin'", 'Eve featuring Alicia Keys', 2002],
 ["Nothin'", 'N.O.R.E.', 2002],
 ['A Moment Like This', 'Kelly Clarkson', 2002],
 ['Just Like a Pill', 'Pink', 2002],
 ["When I'm Gone", '3 Doors Down', 2003],
 ['Right Thurr', 'Chingy', 2003],
 ['Miss You', 'Aaliyah', 2003],
 ['Picture', 'Kid Rock featuring Sheryl Crow', 2003],
 ['Beautiful', 'Christina Aguilera', 2003],
 ['The Game of Love', 'Santana featuring Michelle Branch', 2003],
 ['Headstrong', 'Trapt', 2003],
 ['Freek-a-Leek', 'Petey Pablo', 2004],
 ['Since U Been Gone', 'Kelly Clarkson', 2005],
 ['Behind These Hazel Eyes', 'Kelly Clarkson', 2005],
 ['Beverly Hills', 'Weezer', 2005],
 ['Oh', 'Ciara featuring Ludacris', 2005],
 ['Lonely No More', 'Rob Thomas', 2005],
 ['Lovers & Friends', 'Lil Jon featuring Usher and Ludacris', 2005],
 ['Caught Up', 'Usher', 2005],
 ["Sugar, We're Goin Down", 'Fall Out Boy', 2005],
 ['Bad Day', 'Daniel Powter', 2006],
 ['Temperature', 'Sean Paul', 2006],
 ['Check on It', 'Beyoncé featuring Slim Thug', 2006],
 ['Run It!', 'Chris Brown featuring Juelz Santana', 2006],
 ['Ms. New Booty', 'Bubba Sparxxx featuring Ying Yang Twins', 2006],
 ['What You Know', 'T.I.', 2006],
 ['What Goes Around… Comes Around', 'Justin Timberlake', 2007],
 ['Bartender', 'T-Pain featuring Akon', 2007],
 ['Love in This Club', 'Usher featuring Young Jeezy', 2008],
```

```
["Don't Stop the Music", 'Rihanna', 2008],
['Tattoo', 'Jordin Sparks', 2008],
['Poker Face', 'Lady Gaga', 2009],
['I Gotta Feeling', 'The Black Eyed Peas', 2009],
['Halo', 'Beyoncé', 2009],
['Fire Burning', 'Sean Kingston', 2009],
['LoveGame', 'Lady Gaga', 2009],
['Turnin Me On', 'Keri Hilson featuring Lil Wayne', 2009],
['I Hate This Part', 'Pussycat Dolls', 2009],
['I Gotta Feeling', 'The Black Eyed Peas', 2010],
["Club Can't Handle Me", 'Flo Rida featuring David Guetta', 2010],
['Down', 'Jay Sean featuring Lil Wayne', 2010],
['Party Rock Anthem', 'LMFAO featuring Lauren Bennett and GoonRock', 2011],
['S&M', 'Rihanna', 2011],
['Pumped Up Kicks', 'Foster the People', 2011],
["What's My Name?", 'Rihanna featuring Drake', 2011],
['Good Life', 'OneRepublic', 2011],
['Black and Yellow', 'Wiz Khalifa', 2011],
['The Time (Dirty Bit)', 'The Black Eyed Peas', 2011],
['Only Girl (In the World)', 'Rihanna', 2011],
['Yeah 3x', 'Chris Brown', 2011],
['We Are Young', 'Fun featuring Janelle Moní¡e', 2012],
['Take Care', 'Drake featuring Rihanna', 2012],
['Too Close', 'Alex Clare', 2012],
['Give Your Heart a Break', 'Demi Lovato', 2012],
["Can't Hold Us", 'Macklemore & Ryan Lewis featuring Ray Dalton', 2013],
['I Love It', 'Icona Pop featuring Charli XCX', 2013],
["Hold On, We're Going Home", 'Drake featuring Majid Jordan', 2013],
['Beauty and a Beat', 'Justin Bieber featuring Nicki Minaj', 2013],
['Home', 'Phillip Phillips', 2013],
['Demons', 'Imagine Dragons', 2014],
['Habits (Stay High)', 'Tove Lo', 2014],
['Boom Clap', 'Charli XCX', 2014],
['Drunk in Love', 'Beyoncé featuring Jay-Z', 2014],
['Anaconda', 'Nicki Minaj', 2014],
['Break Free', 'Ariana Grande featuring Zedd', 2014],
['Bailando',
 'Enrique Iglesias featuring Descemer Bueno and Gente de Zona',
 2014],
['Burn', 'Ellie Goulding', 2014],
['Wiggle', 'Jason Derulo featuring Snoop Dogg', 2014],
['Rather Be', 'Clean Bandit featuring Jess Glynne', 2014],
["Don't Tell 'Em", 'Jeremih featuring YG', 2014],
['Show Me', 'Kid Ink featuring Chris Brown', 2014],
['Not a Bad Thing', 'Justin Timberlake', 2014],
['Roar', 'Katy Perry', 2014],
['Blank Space', 'Taylor Swift', 2015],
```

```
 ['Want to Want Me', 'Jason Derulo', 2015],
 ['G.D.F.R.', 'Flo Rida featuring Sage the Gemini and Lookas', 2015],
 ['Flex (Ooh, Ooh, Ooh)', 'Rich Homie Quan', 2015],
 ['Work', 'Rihanna featuring Drake', 2016],
 ["Don't Let Me Down", 'The Chainsmokers featuring Daya', 2016],
 ['Cake by the Ocean', 'DNCE', 2016],
 ['Cold Water', 'Major Lazer featuring Justin Bieber and Mí\x98', 2016],
 ['Send My Love (To Your New Lover)', 'Adele', 2016],
 ['Too Good', 'Drake featuring Rihanna', 2016],
 ['No', 'Meghan Trainor', 2016],
 ['Let Me Love You', 'DJ Snake featuring Justin Bieber', 2016]]
```

[28]:
```
%%skip

#Set up Genius API
genius = lyricsgenius.
 ↪Genius("20Y2IB47Ha-FxWMhpV6f26xiH7r-wnLDyWnXaVIGUnSAgCybJTw5TpuQubs9eegv",␣
 ↪skip_non_songs=True, remove_section_headers=True, timeout=10)
genius.verbose = False  # Hide output

#Threshold for fuzzy title matching
FUZZY_THRESHOLD = 85

#Load the missing lyrics CSV
df = pd.read_csv("missing_2000slyrics_to_scrape.csv")

# Create a new column for the lyrics
lyrics_list = []

#Loop through each missing entry
for idx, row in df.iterrows():
    title = str(row["Title"])
    artist = str(row["Artist(s)"])
    year = int(row["Year"])
    print(f"  {year} - Searching for: {title} by {artist}")

    try:
        song = genius.search_song(title, artist)
    except Exception as e:
        print("Genius API failed:", e)
        lyrics_list.append("None")
        continue

    if song is None:
        print("Exact match not found, trying fuzzy fallback...")
        lyrics_list.append("None")
        continue
```

```
    #Fuzzy match verification (score + lyrics content)
    score = fuzz.ratio(song.title.lower(), title.lower())
    if score > FUZZY_THRESHOLD or title.lower() in song.lyrics.lower():
        print(f"Accepted (score: {score})")
        lyrics_list.append(song.lyrics)
    else:
        print(f"Rejected (score: {score})")
        lyrics_list.append("None")

    time.sleep(1)

#Save back to CSV
df["Lyrics"] = lyrics_list
df.to_csv("missing_2000slyrics_to_scrape_filled.csv", index=False)
df.to_json("missing_2000slyrics_to_scrape_filled.json", index=False)
print("Updated CSV saved.")
```

[29]:
```
df_final = pd.read_json("MOST_ACCURATE2.json")
df_final
```

[29]:
```
      No.                                Title  \
0       1  The First Time Ever I Saw Your Face
1       2              Alone Again (Naturally)
2       3                           American Pie
3       4                            Without You
4       5                          The Candy Man
…   …                                      …
2495   46                          Need to Know
2496   47                       Wants and Needs
2497   48                            Way 2 Sexy
2498   49                            Telepatía
2499   50                               Whoopty

                                  Artist(s)  Year  \
0                             Roberta Flack  1972
1                     Gilbert O'Sullivan  1972
2                               Don McLean  1972
3                            Harry Nilsson  1972
4                        Sammy Davis Jr.  1972
…                                      …   …
2495                              Doja Cat  2021
2496                Drake featuring Lil Baby  2021
2497  Drake featuring Future and Young Thug  2021
2498                              Kali Uchis  2021
2499                                      CJ  2021
```

```
                                                  Lyrics
0      16 ContributorsThe First Time Ever I Saw Your …
1      50 ContributorsAlone Again (Naturally) Lyrics[…
2      262 ContributorsTranslationsPolskiAmerican Pie…
3      32 ContributorsWithout You Lyrics[Verse 1]\nNo…
4      15 ContributorsThe Candy Man Lyrics[Intro]\n(C…
…                                                    …
2495   169 ContributorsTranslationsTürkçeEspañol   …
2496   267 ContributorsTranslationsTürkçe    Portu…
2497   2 ContributorsTranslationsEnglishTürkçeEspañol…
2498                                              None
2499   124 ContributorsTranslationsDeutschWhoopty Lyr…

[2500 rows x 5 columns]
```

2 Cleaning the Lyrics

[30]:
```
df_final = df_final.copy()
```

[31]:
```
null_count = df_final["Lyrics"].isnull().sum()
none_count = (df_final["Lyrics"] == "None").sum()
total_missing = null_count + none_count
print(f"Total missing lyrics: {total_missing}")

#Count missing per year
df_final["Year"] = df_final["Year"].astype(int)
missing_per_year = df_final[df_final["Lyrics"].isnull() | (df_final["Lyrics"]
  ↪== "None")].groupby("Year").size()
print("\nMissing lyrics per year:\n", missing_per_year)
```

```
Total missing lyrics: 100

Missing lyrics per year:
 Year
1972    3
1973    1
1974    2
1975    2
1976    6
1977    2
1978    4
1979    1
1980    6
1981    2
1982    2
1983    4
1984    2
```

```
1985    3
1986    4
1988    4
1989    1
1990    4
1992    7
1993    2
1994    4
1995    4
1996    6
1997    6
1998    6
1999    2
2009    1
2010    1
2012    1
2013    1
2015    1
2016    1
2019    1
2020    2
2021    1
dtype: int64
```

[32]:
```python
#Load combined JSON file
with open("MOST_ACCURATE2.json", "r", encoding="utf-8") as file:
    data = json.load(file)

#Define cleaning function
def clean_lyrics(text):
    if text is None or text == "None":
        return None

    text = re.sub(r"\d+\s+Contributors", "", text)
    text = re.sub(r"Translations[\s\S]*?Lyrics", "Lyrics", text)
    text = re.sub(r"\[.*?\]", "", text)  #
    text = re.sub(r"[^A-Za-z,\n ]", "", text)
    text = re.sub(r"\s+", " ", text)
    text = re.sub(r" *\n *", "\n", text).strip()

    return text

#Apply the cleaning function to each entry
for entry in data:
    entry["Lyrics"] = clean_lyrics(entry.get("Lyrics"))

#Save the cleaned version
```

```
with open("all_lyrics_cleaned.json", "w", encoding="utf-8") as file:
    json.dump(data, file, indent=2, ensure_ascii=False)

print("Lyrics cleaned and saved to all_lyrics_cleaned.json")
```

```
Lyrics cleaned and saved to all_lyrics_cleaned.json
```

[33]:
```
df_cleaned = pd.read_json("all_lyrics_cleaned.json")

def clean_text(text):
    if not text or text == 'None':
        return ""
    return re.sub(r'[^\w\s]', '', text.lower())

df_cleaned['Lyrics'] = df_cleaned['Lyrics'].apply(clean_text).fillna('').
 ↪astype(str)

#Create custom decade bins
def create_custom_decade(df, year_column='Year'):
    bins = [1971, 1981, 1991, 2001, 2011, 2021]
    labels = ['1972-1981', '1982-1991', '1992-2001', '2002-2011', '2012-2021']
    df['Custom_Decade'] = pd.cut(df[year_column], bins=bins, labels=labels)
    return df

df_cleaned = create_custom_decade(df_cleaned)
```

3 Visualisations/Analysis

Preface that the lyrics were not as cleaned as I would've liked, e.g. german uses our alphabet so cannot really get rid of the german, so its not just english words, some songs were scraped in russian etc so whilst lyrics are available, they couldn't be saved there are 45 shifts by the end that are out of place

[34]:
```
from textblob import TextBlob

def get_sentiment(text):
    if isinstance(text, str):
        return TextBlob(text).sentiment.polarity
    return None

df_cleaned["Sentiment(textblob)"] = df_cleaned["Lyrics"].apply(get_sentiment)

df_cleaned[["Title", "Year", "Sentiment(textblob)"]].head(5)
```

[34]:

	Title	Year	Sentiment(textblob)
0	The First Time Ever I Saw Your Face	1972	0.275000
1	Alone Again (Naturally)	1972	0.076019

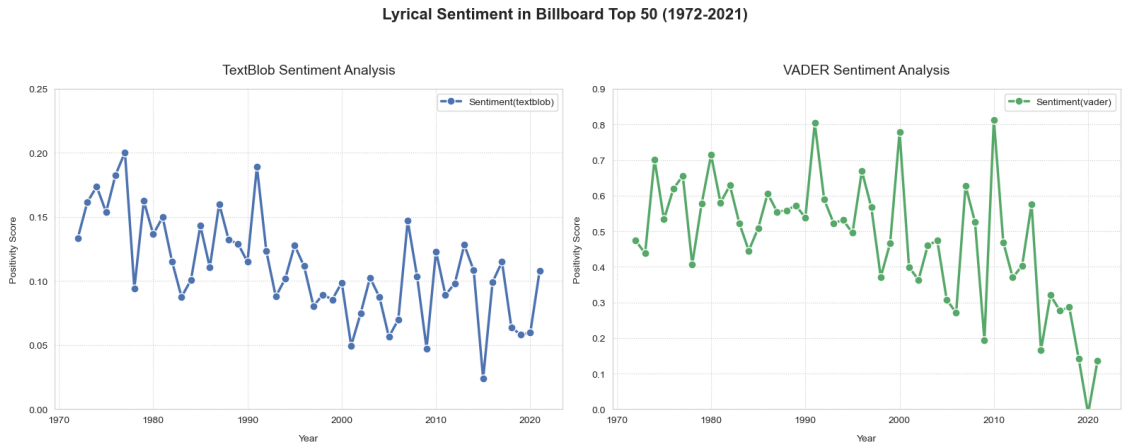
```
2                        American Pie  1972            0.152513
3                         Without You  1972            0.204959
4                       The Candy Man  1972            0.633429
```

[35]:
```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

analyzer = SentimentIntensityAnalyzer()

def get_vader_sentiment(text):
    if isinstance(text, str):
        return analyzer.polarity_scores(text)["compound"]
    return None

df_cleaned["Sentiment(vader)"] = df_cleaned["Lyrics"].apply(get_vader_sentiment)
df_cleaned[["Title", "Year", "Sentiment(vader)"]].head(5)
```

[35]:
```
                                Title  Year  Sentiment(vader)
0  The First Time Ever I Saw Your Face  1972            0.9806
1              Alone Again (Naturally)  1972           -0.9345
2                        American Pie  1972           -0.9979
3                         Without You  1972            0.1290
4                       The Candy Man  1972            0.9981
```

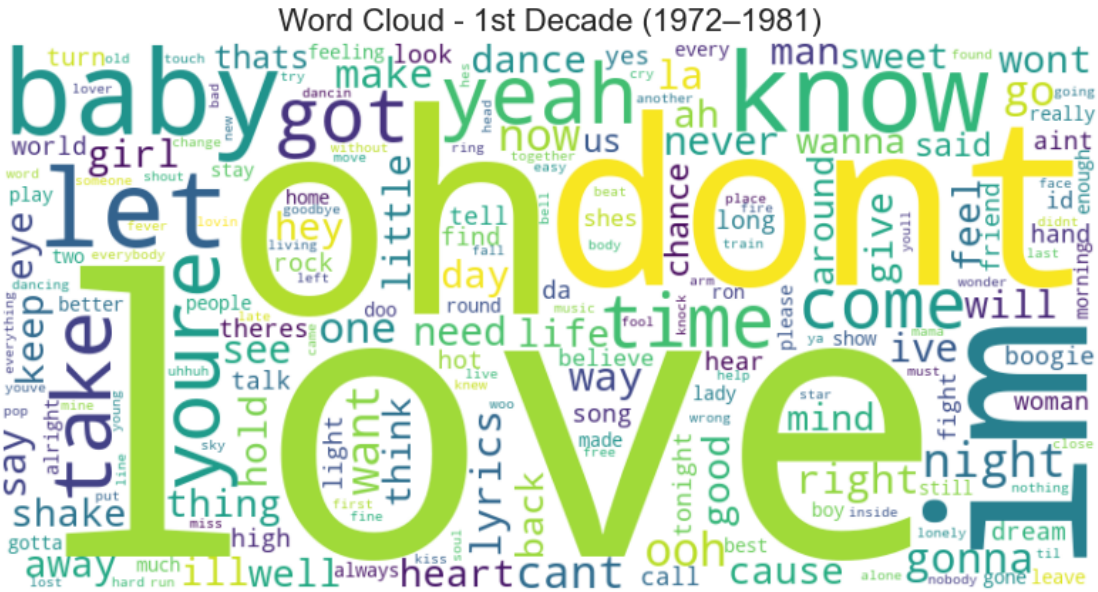
[36]:
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style("whitegrid", {'grid.linestyle': ':'})
plt.rcParams['font.family'] = 'sans-serif'

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6), sharey=False)
fig.suptitle('Lyrical Sentiment in Billboard Top 50 (1972-2021)',
             y=1.05, fontsize=16, fontweight='bold')

#TextBlob Plot
ax1 = df_cleaned.groupby("Year")["Sentiment(textblob)"].mean().plot(
    ax=ax1,
    color='#4C72B0',
    marker='o',
    markersize=8,
    linewidth=2.5,
    markeredgecolor='white',
    markeredgewidth=1
)
ax1.set_title("TextBlob Sentiment Analysis", pad=15, fontsize=14)
ax1.set_xlabel("Year", labelpad=10)
ax1.set_ylabel("Positivity Score", labelpad=10)
ax1.set_ylim(0, 0.25)
```

```
#VADER Plot
ax2 = df_cleaned.groupby("Year")["Sentiment(vader)"].mean().plot(
    ax=ax2,
    color='#55A868',
    marker='o',
    markersize=8,
    linewidth=2.5,
    markeredgecolor='white',
    markeredgewidth=1
)
ax2.set_title("VADER Sentiment Analysis", pad=15, fontsize=14)
ax2.set_xlabel("Year", labelpad=10)
ax2.set_ylabel("Positivity Score", labelpad=10)
ax2.set_ylim(0, 0.9)

ax1.legend(frameon=True, facecolor='white')
ax2.legend(frameon=True, facecolor='white')
plt.tight_layout()
plt.show()
```

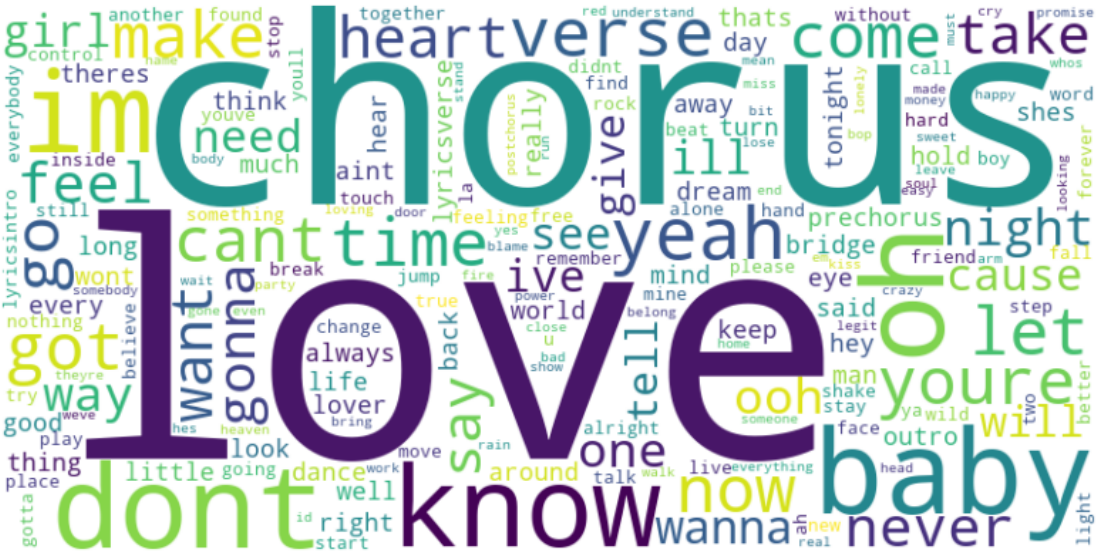


[37]:
```
#Looking at most common words overall & significance of the word 'love'
from wordcloud import WordCloud
#Clean and combine lyrics by decade
first_decade = (" ".join(df_cleaned[df_cleaned['Year'].between(1972,␣
 ↪1982)]["Lyrics"].dropna()))
second_decade = clean_text(" ".join(df[df['Year'].between(1982,␣
 ↪1992)]["Lyrics"].dropna()))
third_decade = clean_text(" ".join(df[df['Year'].between(1992, 2002)]["Lyrics"].
 ↪dropna()))
```

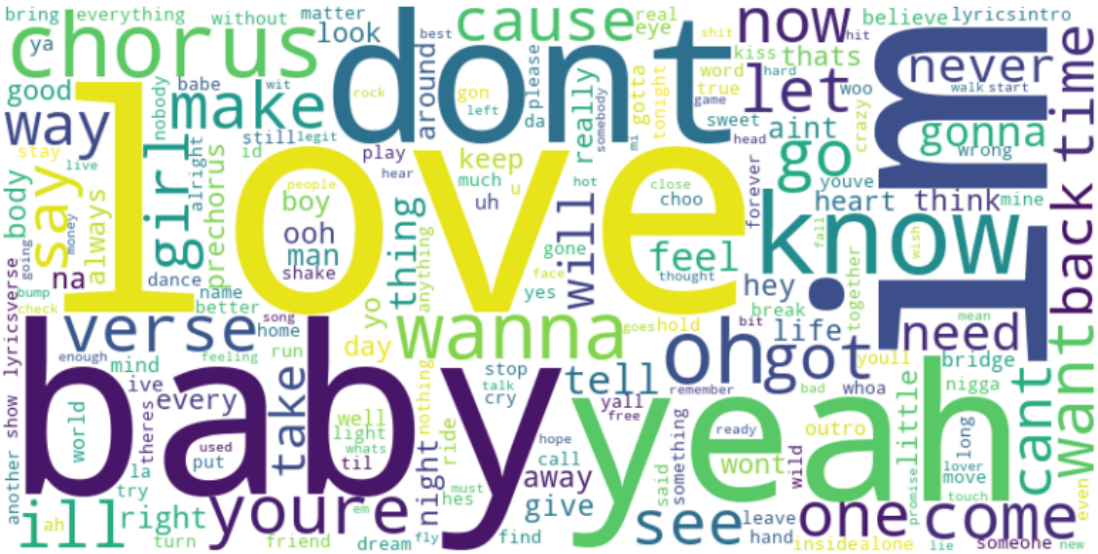
```
fourth_decade = clean_text(" ".join(df[df['Year'].between(2002,␣
 ↪2012)]["Lyrics"].dropna()))
fifth_decade = clean_text(" ".join(df[df['Year'].between(2012, 2021)]["Lyrics"].
 ↪dropna()))

def plot_wordcloud(text, title):
    wc = WordCloud(width=800, height=400, background_color='white',␣
 ↪collocations=False).generate(text)
    plt.figure(figsize=(10, 5))
    plt.imshow(wc, interpolation='bilinear')
    plt.axis('off')
    plt.title(title, fontsize=16)
    plt.show()

plot_wordcloud(first_decade, "Word Cloud - 1st Decade (1972-1981)")
plot_wordcloud(second_decade, "Word Cloud - 2nd Decade (1982-1991)")
plot_wordcloud(third_decade, "Word Cloud - 3rd Decade (1992-2001)")
plot_wordcloud(fourth_decade, "Word Cloud - 4th Decade (2002-2011)")
plot_wordcloud(fifth_decade, "Word Cloud - 5th Decade (2012-2021)")
```



Word Cloud - 2nd Decade (1982–1991)

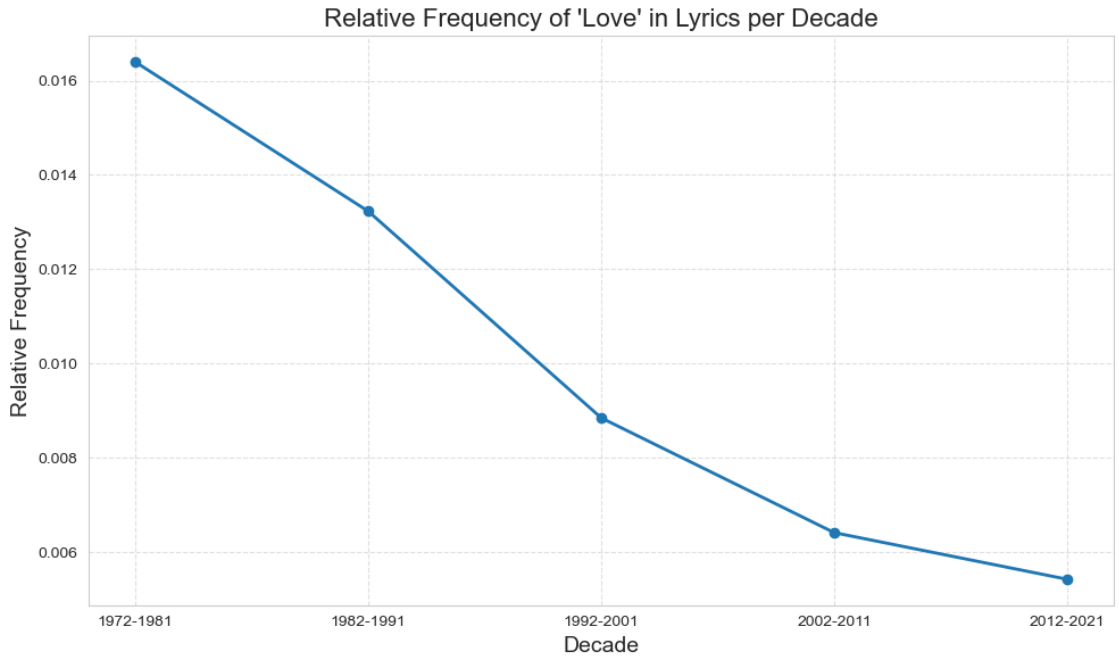


Word Cloud - 3rd Decade (1992–2001)



[ ]:
```
df = df_cleaned
df = df.dropna(subset=['Lyrics']).copy()

def count_words_and_love(text):
    text = re.sub(r'[^\w\s]', '', text.lower())
    words = text.split()
    total_words = len(words)
```

```
    love_count = sum(1 for word in words if word == 'love')
    return total_words, love_count

df[['Total_Words', 'Love_Count']] = df['Lyrics'].apply(lambda x: pd.
 ↪Series(count_words_and_love(x)))

decade_summary = df.groupby('Custom_Decade').sum()

#Calculate relative frequency
decade_summary['Love_Relative_Frequency'] = decade_summary['Love_Count'] /␣
 ↪decade_summary['Total_Words']

plt.figure(figsize=(10,6))
plt.plot(decade_summary.index, decade_summary['Love_Relative_Frequency'],␣
 ↪marker='o', linewidth=2)
plt.title("Relative Frequency of 'Love' in Lyrics per Decade", fontsize=16)
plt.xlabel("Decade", fontsize=14)
plt.ylabel("Relative Frequency", fontsize=14)
plt.grid(True, linestyle='--', alpha=0.6)
plt.xticks(decade_summary.index, labels=decade_summary.index)
plt.tight_layout()
plt.show()
```



[39]:
```
#Define misogynistic words
misogynistic_words = [
```

```
    "pussy", "bitch", "bitches", "hoe", "whore", "female", "booty", "gold␣
 ↪diggers",
    "side chick", "thot", "bunny", "brick house", "good girl", "pipe her",
    "tits", "breasts", "vagina", "her body", "that body", "fuck her",
    "shake that", "good girls", "ass", "chick", "ho", "baby girl",
    "pum pum", "slap her", "smack that", "hoes", "twerk", "run a train"
]

pattern = r'\b(' + '|'.join(map(re.escape, misogynistic_words)) + r')\b'

df_cleaned['Misogyny_Count'] = df_cleaned['Lyrics'].str.count(pattern, flags=re.
 ↪IGNORECASE)

word_counts = {word: df_cleaned['Lyrics'].str.count(rf'\b{re.escape(word)}\b',␣
 ↪flags=re.IGNORECASE).sum()
               for word in misogynistic_words}

word_counts_df = (
    pd.DataFrame.from_dict(word_counts, orient='index', columns=['Count'])
    .sort_values(by='Count', ascending=False)
)

print(word_counts_df)
```

```
             Count
bitch          648
ass            321
booty          278
ho             235
bitches        161
hoes           151
baby girl      132
chick           93
shake that      74
that body       72
pussy           66
good girl       52
twerk           37
good girls      34
smack that      34
pum pum         25
thot            15
bunny           10
whore           10
her body         8
breasts          6
female           5
```

```
pipe her          5
fuck her          4
tits              3
gold diggers      3
hoe               3
vagina            2
slap her          1
brick house       1
side chick        1
run a train       1
```

[40]:
```
def count_misogyny(text):
    if not isinstance(text, str):
        return 0
    pattern = re.compile(r'\b(?:' + '|'.join(map(re.escape,␣
 ↪misogynistic_words)) + r')\b', flags=re.IGNORECASE)
    return len(pattern.findall(text.lower()))

df_cleaned['Misogyny_Count'] = df_cleaned['Lyrics'].apply(count_misogyny)
df_cleaned['Is_Misogynistic'] = df_cleaned['Misogyny_Count'] > 0

yearly_summary = df_cleaned.groupby("Year").agg(
    TotalMentions=("Misogyny_Count", "sum"),
    ContainsMisogyny=("Is_Misogynistic", "sum")
).reset_index()

df_cleaned
```

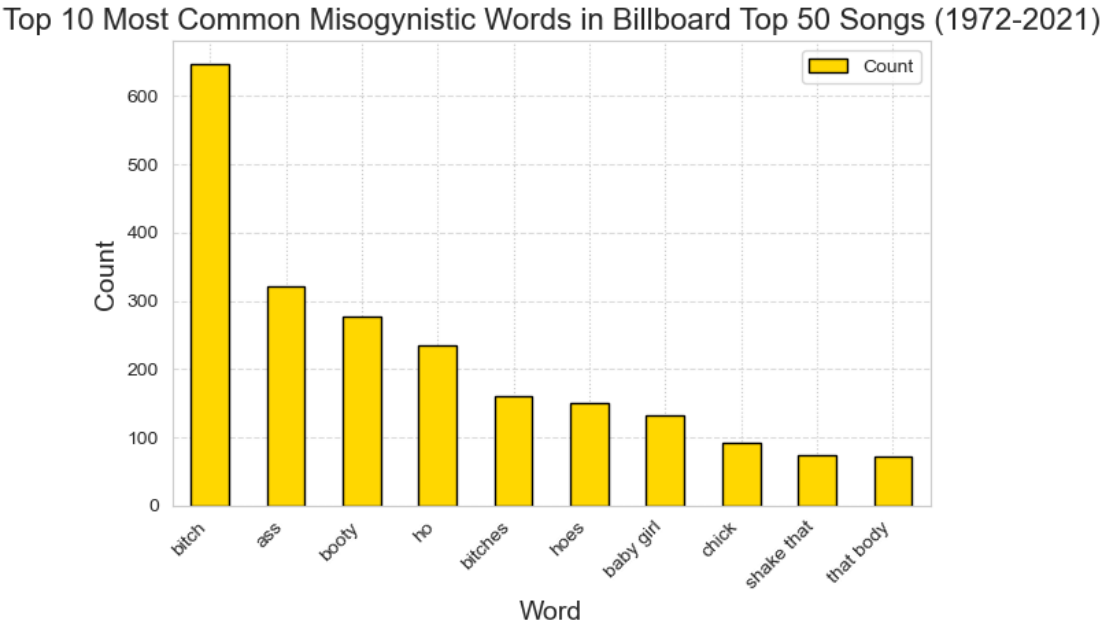
[40]:
```
      No.                                Title  \
0       1  The First Time Ever I Saw Your Face
1       2              Alone Again (Naturally)
2       3                           American Pie
3       4                            Without You
4       5                         The Candy Man
…     …                                      …
2495   46                         Need to Know
2496   47                       Wants and Needs
2497   48                            Way 2 Sexy
2498   49                            Telepatía
2499   50                               Whoopty

                                  Artist(s)  Year  \
0                             Roberta Flack  1972
1                     Gilbert O'Sullivan  1972
2                               Don McLean  1972
3                             Harry Nilsson  1972
4                          Sammy Davis Jr.  1972
```

```
…                                    …   …
2495                               Doja Cat  2021
2496                 Drake featuring Lil Baby  2021
2497  Drake featuring Future and Young Thug  2021
2498                              Kali Uchis  2021
2499                                      CJ  2021

                                                 Lyrics Custom_Decade  \
0     the first time ever i saw your face lyrics the…     1972-1981
1     alone again naturally lyrics oh in a little wh…     1972-1981
2     lyrics a long long time ago i can still rememb…     1972-1981
3     without you lyrics no i cant forget this eveni…     1972-1981
4     the candy man lyrics candy man hey candy man a…     1972-1981
…                                                   …           …
2495  lyrics yeah wanna know what its like like baby…     2012-2021
2496  lyrics six yeah yeah yeah leave me out the com…     2012-2021
2497  lyrics way sexy    dl read more                …     2012-2021
2498                                                       2012-2021
2499  lyrics heavy on the sso shit man loyalty over …     2012-2021

      Sentiment(textblob)  Sentiment(vader)  Misogyny_Count  Is_Misogynistic
0                0.275000           0.9806               0            False
1                0.076019          -0.9345               0            False
2                0.152513          -0.9979               0            False
3                0.204959           0.1290               0            False
4                0.633429           0.9981               0            False
…                     …                …               …              …
2495            -0.052719           0.9806               3             True
2496             0.010370           0.9969               0            False
2497             0.500000           0.5267               0            False
2498             0.000000           0.0000               0            False
2499            -0.108766          -0.9947               8             True

[2500 rows x 10 columns]
```

[41]:
```
top_10_misogynistic_words = word_counts_df.head(10)

plt.figure(figsize=(10,6))
top_10_misogynistic_words.plot(kind='bar', color='gold', edgecolor='black')

plt.title('Top 10 Most Common Misogynistic Words in Billboard Top 50 Songs␣
 ↪(1972-2021)', fontsize=16)
plt.xlabel('Word', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
```

```
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



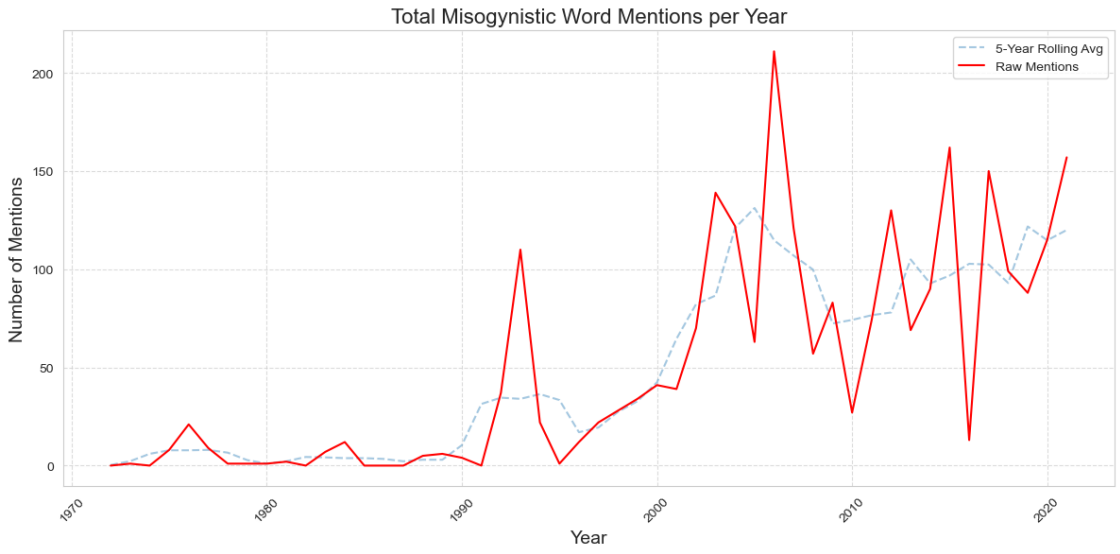
[42]:
```
#Compile regex pattern to match words
pattern = re.compile(r'\b(?:' + '|'.join(map(re.escape, misogynistic_words)) +␣
 ↪r')\b', re.IGNORECASE)

#Count how many misogynistic terms  in each lyric
df_cleaned["Misogynistic_Count"] = df_cleaned["Lyrics"].apply(lambda x:␣
 ↪len(pattern.findall(x)) if isinstance(x, str) else 0)

#Flag whether the song is misogynistic
df_cleaned["Is_Misogynistic"] = df_cleaned["Misogynistic_Count"] > 0

#Group by year and summarise
summary = df_cleaned.groupby("Year").agg(
    Songs_Checked=("Lyrics", "count"),
    Misogynistic_Songs=("Is_Misogynistic", "sum"),
    Total_Misogynistic_Words=("Misogynistic_Count", "sum")
).reset_index()

#Add % of misogynistic songs
summary["Percentage_Misogynistic_Songs"] = (
    summary["Misogynistic_Songs"] / summary["Songs_Checked"] * 100
)
```

```
summary.tail(20)
```

[42]:

	Year	Songs_Checked	Misogynistic_Songs	Total_Misogynistic_Words	Percentage_Misogynistic_Songs
30	2002	50	19	70	38.0
31	2003	50	22	139	44.0
32	2004	50	19	122	38.0
33	2005	50	18	63	36.0
34	2006	50	18	211	36.0
35	2007	50	20	121	40.0
36	2008	50	16	57	32.0
37	2009	50	19	83	38.0
38	2010	50	12	27	24.0
39	2011	50	12	74	24.0
40	2012	50	10	130	20.0
41	2013	50	9	69	18.0
42	2014	50	16	90	32.0
43	2015	50	19	162	38.0
44	2016	50	7	13	14.0
45	2017	50	19	150	38.0
46	2018	50	18	99	36.0
47	2019	50	14	88	28.0
48	2020	50	14	115	28.0
49	2021	50	17	157	34.0

[43]:
```
def count_misogyny(text):
    if not isinstance(text, str):
        return 0
    pattern = re.compile(r'\b(?:' + '|'.join(map(re.escape,␣
 ↪misogynistic_words)) + r')\b', flags=re.IGNORECASE)
    return len(pattern.findall(text.lower()))

df_cleaned['Misogyny_Count'] = df_cleaned['Lyrics'].apply(count_misogyny)
df_cleaned['Is_misogynistic'] = df_cleaned['Misogyny_Count'] > 0

yearly_summary = df_cleaned.groupby("Year").agg(
    TotalMentions=("Misogyny_Count", "sum"),
    ContainsMisogyny=("Is_misogynistic", "sum")
).reset_index()

yearly_summary["MentionsRolling"] = yearly_summary["TotalMentions"].
 ↪rolling(window=5, center=True, min_periods=1).mean()
yearly_summary["SongsRolling"] = yearly_summary["ContainsMisogyny"].
 ↪rolling(window=5, center=True, min_periods=1).mean()

plt.figure(figsize=(12, 6))
plt.plot(yearly_summary["Year"], yearly_summary["MentionsRolling"], alpha=0.4,␣
 ↪linestyle='--', label="5-Year Rolling Avg")
plt.plot(yearly_summary["Year"], yearly_summary["TotalMentions"], color='red',␣
 ↪linewidth=1.5, label="Raw Mentions")
plt.title("Total Misogynistic Word Mentions per Year", fontsize=16)
plt.xlabel("Year", fontsize=14)
plt.ylabel("Number of Mentions", fontsize=14)
plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[44]:
```
decade_counts = {}

for decade, group in df_cleaned.groupby('Custom_Decade'):
    lyrics = " ".join(group['Lyrics'].dropna().astype(str))
    word_freq = {
        word: len(re.findall(r'\b' + re.escape(word) + r'\b', lyrics.lower()))
        for word in misogynistic_words
    }
    decade_counts[decade] = word_freq

#Convert to df
decade_word_df = pd.DataFrame(decade_counts).fillna(0).astype(int)
top_words = word_counts_df.head(10).index.tolist()
decade_word_df = decade_word_df.loc[top_words]

#Plot the stacked bar chart
decade_word_df.T.plot(
    kind='bar',
    stacked=True,
    figsize=(14, 8),
    colormap='coolwarm'
)

plt.title('Stacked Misogynistic Word Usage by Decade', fontsize=18)
plt.xlabel('Decade', fontsize=14)
plt.xticks (rotation = 45)
plt.ylabel('Word Count', fontsize=14)
plt.legend(title='Words', bbox_to_anchor=(1.05, 1), loc='upper left')
```

```
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```