title: Data Procurement And Processing

nav_include: 3

Data procurement

To get data that describes songs ("audio features" in Spotify), we queried Spotify API for song identifier (Spotify id). We tried to fetch song similarity data from Spotify but that was taking too much time and hence wasn't added to our dataset. Overall, we used compressed csv files to store intermediate and final data sets.

Data processing

- All names (playlists names, song names) where cleaned by removing punctuation, extra spaces, etc
- Songs in playlists processed to find co-occurences of songs in playlists. Our take is that song co-occurence in playlist shows signal for song similarity (of course, Spotify promoted songs and "hits" add noise). If songs are found in the same list - that adds a count and this pair is saved into resulting set.

Sources below. These are Python scripts, not Jupyter notebooks to decrease memory pressure and let scripts run unattended.

In [1]:
```
# Preprocessing:
#!/usr/bin/env python
# coding: utf-8

import sys
import json
import codecs
import datetime
import numpy as np
import pandas as pd
import re
import time
import gzip
import csv

DATA_DIR = "./data/data/"
pretty = True
compact = False
cache = {}

def cleanString(s):
    s = re.sub(r'[^\w\s]','', s) # remove punctuation
    s = re.sub(' +', ' ', s) # remove double spaces
    s = "".join(i for i in s if ord(i)<128) # remove non-ascii letters
    return s.lower().strip()

def getId(dict, str) :
    s = cleanString(str)
    if len(s) == 0 :
        return -1
    if s in dict :
        id = dict[s]
    else :
        id = len(dict)
        dict[s] = id
    return id

def getSongId(dict, track, s_id) :
    s = cleanString(track['track_name'])
    if len(s) == 0 :
        return (-1, s_id)
    if s in dict :
        return (dict[s][0], s_id)
    id = s_id
    s_id += 1
    dict[s] = [ id, track['track_uri'].split(':')[2], float(track['duration_ms'
]) / 1000 ]
    return (id, s_id)

def getPlaylId(dict, playlist) :
    s = cleanString(playlist['name'])
    if len(s) == 0 :
        return -1
    id = len(dict)
    dict[id] = [ s, int(playlist['num_followers']), 1 if bool(playlist['collabo
rative']) else 0,
                int(playlist['num_tracks']), int(playlist['num_albums']) ]
    return id

def full_playlist(start, end, showOnly=True, namesOnly=False, data=None, playlD
ata=None, lastIndex=0):
    playlists = None
    prevFile = None
```

In [ ]:
```
# Song info fetch. Several files were stored and merged to account for transmis
sion failures, etc.
import urllib
import urllib.request as request
import json
import time
import gzip
import csv
import sys

regSleep = 1.0 / 165;
authH = {'Accept': 'application/json',
    'User-agent': 'Mozilla/5.0',
    "Content-Type": "application/json",
    'Authorization' : 'Bearer {}'.format(
    sys.argv[3]
    ) }

startId = sys.argv[2]
searchFirst = True
DATA_DIR = "./data/data/"
with gzip.open(DATA_DIR + sys.argv[1] + "_aug_songs_" + (startId if startId is
not None else "") + ".csv.gz", 'wt') as fz:
    writer = csv.writer(fz, delimiter=',')
    if startId is None :
        writer.writerow(['name', 'id', 'uri', 'duration', 'danceability', 'ener
gy', 'key', 'loudness', 'mode',
               'speechiness', 'acousticness', 'instrumentalness', 'liveness', '
valence', 'tempo', 'time_signature'])
    with gzip.open(DATA_DIR + "songs.csv.gz", mode="rt") as f:
        print("file", f)
        csvobj = csv.reader(f, delimiter=',', quotechar="'")
        first = True
        fullStop = False
        for line in csvobj:
            if first :
                first = False
                continue
            if len(line) == 0 :
                continue
            id = line[2]
            print("->", id)
            if startId is not None and searchFirst:
               if startId == id :
                   print("found", startId)
                   searchFirst = False
               else:
                   continue
            for retry in range(0, 10):
                code = -1
                try :
                    req = request.Request(url = "https://api.spotify.com/v1/aud
io-features/{}".format(id), headers=authH)
                    resp = request.urlopen(req)
                    content = resp.read()
                    resp = json.loads(content)
                    out = [line[0], line[1], line[2], line[3], resp['danceabili
ty'], resp['energy'], resp['key'], resp['loudness'], resp['mode'],
                           resp['speechiness'], resp['acousticness'], resp['ins
trumentalness'], resp['liveness'], resp['valence'], resp['tempo'],
                           resp['time_signature'] ]
                    writer.writerow(out)
                    time.sleep(regSleep)
```

In [ ]:
```
# Song similarity - done in chunks to fit into memory
import gzip
import csv
import sys
import numpy as np

def calcPlayl(plId, songs) :
    n = len(songs)
    if n == 0 or n == 1:
        return
    print (plId, "len", n)
    r = range(0, n)
    for i in r :
        if matrix[songs[i], 0] is None :
            matrix[songs[i], 0] = {}
        for j in range(i + 1, n) :
            otherId = songs[j]
            if otherId in matrix[songs[i], 0] :
                matrix[songs[i], 0][otherId] += 1
            else :
                matrix[songs[i], 0][otherId] = 1


DATA_DIR = "./data/data/"
numSongs = 1389689 + 1 # max song id , starts from 0
matrix = np.empty([numSongs, 1], dtype=np.dtype('O'))

cnt = 0
MAX = 233000
runId = int(sys.argv[1])
startPl = runId * MAX
endPl = startPl + MAX
with gzip.open(DATA_DIR + "preproc.csv.gz", mode="rt") as f:
    print("file", f)
    csvobj = csv.reader(f, delimiter=',', quotechar="'")
    first = True
    prevPl = -1
    songs = []
    for line in csvobj:
        if first :
            first = False
            continue
        if len(line) == 0 :
            continue
        songId = int(line[2])
        plId = int(line[5])
        if plId < startPl :
            continue
        if plId > endPl :
            break
        if plId != prevPl :
            calcPlayl(prevPl, songs)
            songs = []
        songs.append(songId)
        prevPl = plId

print("done from ", startPl, "to", endPl)
with gzip.open(DATA_DIR + str(runId) + "_simsong_calc_.csv.gz", 'wt') as fz:
    writer = csv.writer(fz, delimiter=',')
    writer.writerow(['songid', 'simsongid', 'count'])
    for i in range(0, matrix.shape[0]) :
        if matrix[i, 0] is None :
            continue
```

In [ ]:
```
# Song similarity merge
import gzip
import csv
import sys

def empty(line1) :
    return line1 is None or len(line1) == 0

def allEmpty(dones) :
    for done in dones :
       if not done:
           return False
    return True

def writeSims(writer, songId, simSongs):
    for key, value in sorted(simSongs.items(), key=lambda kv: kv[1], reverse=Tr
ue) :
        # cutoff ?
        writer.writerow([songId, key, value])

#return false if song ids don't match
def addSongInfo(songId, line, simSongs):
    # print("add", songId, line, simSongs)
    newSongId = int(line[0])
    if songId == newSongId :
        simId = int(line[1])
        cnt = int(line[2])
        if simId in simSongs :
            simSongs[simId] += cnt
        else :
            simSongs[simId] = cnt
        return True
    return False

#return empty list if end of file is reached, next-song line otherwise
def fetchSongInfo(songId, prevLine, curIter, simSongs) :
    # print("fetch", songId, prevLine)
    if not empty(prevLine) :
        match = addSongInfo(songId, prevLine, simSongs)
        if not match :
            return prevLine
    while True :
        line = next(curIter, None)
        if empty(line):
            return []
        match = addSongInfo(songId, line, simSongs)
        if not match :
            return line

DATA_DIR = "./data/data/"

numSongs = 1389689 + 1 # max song id , starts from 0
numFiles = 2
files = [None]*numFiles
csvobj = [None]*numFiles
iters = [None]*numFiles
prevLine  = [None]*numFiles
doneFile = [False]*numFiles
simSongs = {}
fCnt = 0
file0 = sys.argv[1]
file1 = sys.argv[2]
with gzip.open(DATA_DIR + file0 + "_" + file1 + "_simsong_calc_.csv.gz", 'wt')
```