Lab 2 – MATH 240 – Computational Statistics

Ben Horner
Colgate University
Math Department
bhorner@colgate.edu

Febuary 6, 2025

Abstract

Keywords: Batch File, Data Processing, Music, JSON

1 Introduction

In 2018, two of Professor Cipolli's favorite bands - The Front Bottoms and Manchester Orchestra - released a song they collaborated on called Allen Town. In a statement to Noisey - the music arm of Vice - Andy Hull of Manchester Orchestra recalled that the creation of this track started when Nate Hussey of All Get Out sent him the first four lines of the track. Andy Hull worked out the melody and music and shared it with Brian Sella of The Front Bottoms, who then helped develop the chorus.

This brings us to an interesting question: which band contributed most to the song?

1.1 Task

To attempt to answer this question, we first need to create a batch file for data processing, aiming to use Essentia - an open-source program for music analysis, description, and synthesis - to create data about what each band's tracks "sound like." In this Lab, we will begin by creating the code for building this batch file using a set of example .wav files. Then we will process the .JSON output and extract key descriptors.

2 Methods

We divide our overall goal in to two tasks. **Task 1**, where we build the batch files for data processing, and **Task 2**, where we process the data. For this Lab, we use test files created by Professor Cipolli located in a directory labeled "MUSIC."

2.1 Task 1: Building a Batch File for Data Processing

To write the code to create the desired batch file, we will need to convert them from the format of `[track number]-[artist]-[track name].wav` to `[artist name]-[album name]-[track name].json`.

1. First, we install the `stringr` package for `R` (Wickham, 2023). We will be using it to split the file name apart to isolate the artist and album and to change the file to `.json`.

```
library(stringr)
```

2. Next, we isolate the subdirectories (artists and albums) and files (songs) within the `MUSIC` directory using the `list_dirs, list_files` and the `str_count()` function to count the number of forward slashes.

```
artist.dirs = c()
album.dirs = c()
for (directory in music.dirs){
  dir.level <- str_count(directory, pattern = "/")
  if (dir.level == 1){
  artist.dirs = append(artist.dirs, directory)
  }
  else if (dir.level == 2){
    album.dirs = append(album.dirs, directory)
  }
}
```

3. For each album, we isolated each `.wav` file using `str_count()` to subset all `.wav` files from the current album subdirectory. For each file (song) we used `str_split()` to extract just the track name, which we then pasted (`paste()` function) together with the artist name, track name, and .json to create an object of `[artist name]-[album name]-[track name].json`. We also pasted `streaming_extractor_music.exe` to the file name to create the command line prompt for the current track.

```
json.files = c()
for (album in album.dirs){
  ###Splits the string of the directory into Music, Artist, Album
  split.string = str_split(album, pattern = "/", simplify = TRUE)
  album.name = split.string[1, 3]
  artist.name = split.string[1, 2]
  art.n.album = paste(artist.name, album.name, sep = "-") #artist and albun
                                                          #comined in desir


  ##Lists the music files in each album, goes through them, creates the fin
  ##desired string of [artist name]-[album name]-[trackname].json
  music.files <- list.files(album)
  for (music.file in music.files){
    if (str_count(music.file, pattern = ".wav")){
      #Makes sure we are only dealing with .wav files
      file.path = str_split(music.file, pattern = ".wav", simplify =  TRUE)
      file.path = str_split(file.path, pattern = "-", simplify = TRUE)
      song.name = file.path[1, 3]
      final.path = paste(art.n.album, "-", song.name, ".json", sep = "")
      json.files = append(wav.files, final.path) #will contain all of the .
    }
  }
}

code.to.process = paste("streaming_extractor_music.exe ", json.files)
```

4. Finally, using the `writeLines()` function, we wrote the file names to a .txt file called batfile.txt.

```
writeLines(code.to.process, "batfile.txt")
```

2.2 Task 2: Process JSON Output

After being provided the .JSON output for the song "Au Revoir (Adios)," we analyze it for the key descriptors of `average_loudness`, `mean` of `spectral_energy`, `danceability`, `bpm`, `key_key` (musical key) `key_scale` (musical mode), and `length` (duration in seconds). We do this by installing the `jsonlite` package for `R` (Ooms, 2014), using `str_split` to extract the artist, album, and track from the file name. And finally loading the JSON file into `R` via `fromJSON()`, before extracting the key descriptors.

```
#Step 0
library(jsonlite)

#Step 1
json.file = (list.files())[8]
#gives us the file we want. In the future, we can just loop through the .json
#files that we have in a list or folder
split.file = str_split(json.file, "-", simplify = TRUE)
artist.name = split.file[1, 1]
album = split.file[1, 2]
song.name = split.file[1, 3] #this still has .json on it
song.name = str_split(song.name, ".json", simplify = TRUE)[1,1]

#Step 2
json.data = fromJSON(json.file)

#Step 3
average.loudness = json.data$lowlevel$average_loudness
mean.spectral.enrgy = json.data$lowlevel$spectral_energy$mean
danceability = json.data$rhythm$danceability
bpm = json.data$rhythm$bpm
musical.key = json.data$tonal$key_key
musical.mode = json.data$tonal$key_scale
#duration = json.data ####in seconds, cant find in file
```

3 Results

To answer our question of which band contributed most to a song, we need to be able to analyze the data of the song to see what it "sounds like." Following this lab, we will be using Essentia to do this. Via our creation of the batch file and processing the .JSON output, we now have the code to turn other songs into Essentia ready file commands. For the already processed song "Au Revoir (Adios)", we now can examine key descriptors of it.

4 Discussion

In this Lab, we laid the ground work for analyzing many more songs, as now we have the code to process them into .json files to be output and analyzed for key descriptors.

References

Ooms, J. (2014). The jsonlite package: A practical and consistent mapping between json data and r objects.

Wickham, H. (2023). *stringr: Simple, Consistent Wrappers for Common String Operations*. R package version 1.5.1, https://github.com/tidyverse/stringr.