FINAL-TERM ASSIGNMENT REPORT

INTELLIGENT SIGNAL PROCESSING COURSEWORK EXERCISE 3

Contents

File Structure:

When the shared lab link is opened, we will see the files for both exercise 2 and 3. The files that will be used for exercise 3 are ‘Exercise3_Films’ folder and ‘Exercise 3.ipynb’ After running the application, the ‘OutputFiles’ folder and ‘file_analysis_report.txt’ file will be generated.

The ‘OutputFiles’ folder must be empty when running the application or the following error will be thrown:

```
File '/home/jovyan/work/OutputFiles/Cosmos_War_of_the_Planets_formatOK.mp4' already exists. Overwrite ? [y/N] Not overwriting - exiting
```

This happens as the program does not have direct permission to override all existing files.

Program Sequence:

Install necessary files

```
In [1]: !pip install ffprobe-python
        !pip install ffmpeg-python
```

Required libraries for this application: os, pathlib, ffmpeg and ffprobe

import all necessary libraries and get ffmpeg library files

```
In [2]: from ffprobe import FFProbe
        from os import listdir
        from os.path import isfile, join
        import ffmpeg
        import pathlib

        # current folder path
        curr_path = pathlib.Path().resolve()
```

Once all libraries have been installed and imported, we will define the current folder path by using the pathlib to do so. This path will be used to read the various files and folders of this application.

ffprobe:

Retrieving files from folder

In [4]:
```
# get all files that are present in the Exercise3_Films folder in current path directory
files = [f for f in listdir(f"{curr_path}/Exercise3_Films") if isfile(join(f"{curr_path}/Exercise3_Films", f))]

# open the txt file to write the results of the analysis
result_file = open('file_analysis_report.txt', 'w')

for file in files:
    print(f"\nfile: {file}")
    try:
        metadata=FFProbe(f"{curr_path}/Exercise3_Films/{file}")

        # metadata field names can be found in the following link:
        # https://trac.ffmpeg.org/wiki/FFprobeTips
        # print(metadata)

        # retrieve stream information
        video_stream = metadata.streams[0]
        audio_stream = metadata.streams[1]

        # assign stream fields to variables
        file_format = file.split('.')[1]
        v_codec_name = video_stream.codec_name
        a_codec_name = audio_stream.codec_name
        v_frame_rate = float(video_stream.nb_frames) / float(video_stream.duration)
        v_aspect_ratio = video_stream.display_aspect_ratio
        v_resolution = f"{video_stream.width} x {video_stream.height}"
        v_bitrate = int(video_stream.bit_rate) / 1000000
        a_bitrate = int(audio_stream.bit_rate) / 1000
        a_channel_layout = audio_stream.channel_layout
        a_channels = audio_stream.channels
```

Next, we will retrieve all files from the ‘Exercise3_Films’ folder by looping through all items present in that folder and checking if the ‘combined path + file name’ is a file in that directory.

We will then open a txt file in the write mode which will be used to indicate which files are adheres to the correct format and which of those fields are problematic.

After opening that file, we will iterate through the files in ‘Exercise3_Films’ folder. We will attempt to use ffprobe library to read the metadata of the file and extract the video and audio stream data from the metadata.

These video and audio stream data can be used to retrieve the specified format information of each film. More information on the types of format information available are accessible in the link provided in the program comments. All format information retrieved from those stream data are provided in the form of string value and thus, wherever calculation is required, we will need to convert the string value to the respective data type. Video bitrate is divided by 1000000 as we will want to read the bitrate by Mb units while audio is divided by 1000 as we are reading it in kb units.

```
        print(f"Video format (container): {file_format}")
        print(f"Video codec: {v_codec_name}")
        print(f"Audio codec: {a_codec_name}")
        print(f"Frame rate: {format(v_frame_rate, '.2f')} FPS")
        print(f"Aspect ratio: {v_aspect_ratio}")
        print(f"Resolution: {v_resolution}")
        print(f"v_bitrate: {format(v_bitrate, '.2f')}Mb/s")
        print(f"a_bitrate: {format(a_bitrate, '.2f')}kb/s")
        print(f"channel layout: {a_channel_layout}")
        print(f"channels: {a_channels}")
```

Next, we will simply print all the format information for display purposes.

ffmpeg:

```
        # all field filters and conversions are available at: https://ffmpeg.org/ffmpeg-filters.html
        # input current file to ffmpeg input stream
        stream = ffmpeg.input(f"{curr_path}/Exercise3_Films/{file}")
        video_stream = stream.video
        audio_stream = stream.audio

        # check if variable fields have the correct format
        # if incorrect, convert them to the correct format
        problematic_fields, video_stream, audio_stream = find_problematic_fields(video_stream, audio_stream)

        # write lines to file_analysis_report.txt based on findings
        # if no problematic fields no files need to be generated
        if problematic_fields == "":
            result_file.write(f"{file} - no issues found\n")
        else:
            result_file.write(f"{file} - {problematic_fields}\n")
            output_filename = f"{curr_path}/OutputFiles/{file.split('.')[0]}_formatOK.mp4"
            stream = ffmpeg.output(video_stream, audio_stream, output_filename, format='mp4', vcodec='h264', acodec='aac', video_
            stream = ffmpeg.run(stream, capture_stdout=True, capture_stderr=True)

    except ffmpeg.Error as e:
        print('stdout:', e.stdout.decode('utf8'))
        print('stderr:', e.stderr.decode('utf8'))
        raise e

# close file after analysis
result_file.close()
```

Now, we will begin to filter and convert the films as required.

Similarly, we will first input the current iteration's file to the ffmpeg library and retrieve the audio and video stream data.

The video and audio stream data is then passed to a function that I have written, called find_problematic_fields() which we will go into further details at the later section. In summary, this function finds all problematic fields and writes them into a single string and returns them as the variable, problematic_fields. Some filters are applied to the video or audio stream while in the function while others will only be done at a later stage.

Once the problematic fields have been identified and necessary filters are applied to their respective stream data, we will begin to generate the report and necessary converted films.

If there are no problematic fields, the report txt file will write the file name along with a text indicating that no issues are found with the format of the film. However, if there are problematic fields, all problematic fields will be listed in the report txt file and we will begin to generate the converted format film file.

We will first define the filename as shown above. file.split('.')[0] retrieves the filename of the original file without the extension.

The output file is given the parameter of:

video_stream, audio_stream, output_filename, format='mp4', vcodec='h264', acodec='aac', video_bitrate='2.5M', audio_bitrate='256k', aspect='16:9'

These specific formats are the formats specified. Channel layout does not need to be specified as inputting a video and an audio data will assign the file with a stereo layout.

This process is repeated until all files are checked and converted as necessary.

The output files are unable to be played on the lab environment and must be downloaded and played on local machine.

Functions to be used

```
def find_problematic_fields(video_stream, audio_stream):
    '''
    Takes in the ffmpeg stream as input parameter which will be used to filter the video input.
    Only filter for video framerate and resolution will be performed, other field settings will be done when
    the ffmpeg output video file is made.

    Returns problematic_fields (String), video stream and audio stream
    '''
    problematic_fields = ""
    # file format setting will be set along with output file
    if file_format != "mp4":
        problematic_fields += "file_format "

    # video codec setting will be set along with output file
    if v_codec_name != "h264":
        problematic_fields += "video_codec "

    # audio codec setting will be set along with output file
    if a_codec_name != "aac":
        problematic_fields += "audio_codec "

    if v_frame_rate != "25":
        problematic_fields += "video_frame_rate "
        # section 11.90 fps of the link provided
        video_stream = ffmpeg.filter(video_stream, 'fps', fps=25, round='near')

    # aspect ratio setting will be set along with output file
    if v_aspect_ratio != "16:9":
        problematic_fields += "aspect_ratio "

    if v_resolution != "640 x 360":
        problematic_fields += "resolution "
        video_stream = ffmpeg.filter(video_stream, 'scale', w='640', h='360')
```

find_problematic_fields() function checks the specific fields to see if they match the specified format and if not, they will be added to the problematic_fields. For the video frame rate and resolution, the addition ffmpeg filter will be applied here instead of it being in the output parameter.

Format of original files and results of finding problematic fields

```
file: Cosmos_War_of_the_Planets.mp4
Video format (container): mp4
Video codec: h264
Audio codec: aac
Frame rate: 29.97 FPS
Aspect ratio: 314:177
Resolution: 628 x 354
v_bitrate: 2.99Mb/s
a_bitrate: 317.10kb/s
channel layout: stereo
channels: 2
```

```
file: Last_man_on_earth_1964.mov
Video format (container): mov
Video codec: prores
Audio codec: pcm_s16le
Frame rate: 23.98 FPS
Aspect ratio: 16:9
Resolution: 640 x 360
v_bitrate: 9.29Mb/s
a_bitrate: 1536.00kb/s
channel layout: stereo
channels: 2
```

```
file: The_Hill_Gang_Rides_Again.mp4
Video format (container): mp4
Video codec: h264
Audio codec: aac
Frame rate: 25.00 FPS
Aspect ratio: 16:9
Resolution: 640 x 360
v_bitrate: 7.54Mb/s
a_bitrate: 253.27kb/s
channel layout: stereo
channels: 2
```

```
file: Voyage_to_the_Planet_of_Prehistoric_Women.mp4
Video format (container): mp4
Video codec: hevc
Audio codec: mp3
Frame rate: 29.97 FPS
Aspect ratio: 16:9
Resolution: 640 x 360
v_bitrate: 8.04Mb/s
a_bitrate: 320.00kb/s
channel layout: stereo
channels: 2
```

```
file: The_Gun_and_the_Pulpit.avi
Video format (container): avi
Video codec: rawvideo
Audio codec: pcm_s16le
Frame rate: 25.00 FPS
Aspect ratio: 0:1
Resolution: 720 x 404
v_bitrate: 87.44Mb/s
a_bitrate: 1536.00kb/s
channel layout: unknown
channels: 2
```

```
1 Cosmos_War_of_the_Planets.mp4 - video_frame_rate aspect_ratio resolution channels_layout
2 Last_man_on_earth_1964.mov - file_format video_codec audio_codec video_frame_rate video_bitrate channels_layout
3 The_Hill_Gang_Rides_Again.mp4 - video_frame_rate video_bitrate channels_layout
4 Voyage_to_the_Planet_of_Prehistoric_Women.mp4 - video_codec audio_codec video_frame_rate video_bitrate channels_layout
5 The_Gun_and_the_Pulpit.avi - file_format video_codec audio_codec video_frame_rate aspect_ratio resolution video_bitrate channels_layout
```

Verifying output files have the correct format

Check format of converted files

In [5]:
```
files = [f for f in listdir(f"{curr_path}/OutputFiles/") if isfile(join(f"{curr_path}/OutputFiles/", f))]

for file in files:
    print(f"file: {file}")
    metadata=FFProbe(f"{curr_path}/OutputFiles/{file}")
    # retrieve stream information
    video_stream = metadata.streams[0]
    audio_stream = metadata.streams[1]

    # assign stream fields to variables
    file_format = file.split('.')[1]
    v_codec_name = video_stream.codec_name
    a_codec_name = audio_stream.codec_name
    v_frame_rate = float(video_stream.nb_frames) / float(video_stream.duration)
    v_aspect_ratio = video_stream.display_aspect_ratio
    v_resolution = f"{video_stream.width} x {video_stream.height}"
    v_bitrate = int(video_stream.bit_rate) / 1000000
    a_bitrate = int(audio_stream.bit_rate) / 1000
    a_channel_layout = audio_stream.channel_layout
    a_channels = audio_stream.channels
    print(f"Video format (container): {file_format}")
    print(f"Video codec: {v_codec_name}")
    print(f"Audio codec: {a_codec_name}")
    print(f"Frame rate: {format(v_frame_rate, '.2f')} FPS")
    print(f"Aspect ratio: {v_aspect_ratio}")
    print(f"Resolution: {v_resolution}")
    print(f"v_bitrate: {format(v_bitrate, '.2f')}Mb/s")
    print(f"a_bitrate: {format(a_bitrate, '.2f')}kb/s")
    print(f"channel layout: {a_channel_layout}")
    print(f"channels: {a_channels}\n")
```

```
file: Cosmos_War_of_the_Planets_formatOK.mp4
Video format (container): mp4
Video codec: h264
Audio codec: aac
Frame rate: 25.00 FPS
Aspect ratio: 16:9
Resolution: 640 x 360
v_bitrate: 2.47Mb/s
a_bitrate: 245.59kb/s
channel layout: stereo
channels: 2
```

```
file: Last_man_on_earth_1964_formatOK.mp4
Video format (container): mp4
Video codec: h264
Audio codec: aac
Frame rate: 25.00 FPS
Aspect ratio: 16:9
Resolution: 640 x 360
v_bitrate: 2.57Mb/s
a_bitrate: 240.95kb/s
channel layout: stereo
channels: 2
```

```
file: The_Hill_Gang_Rides_Again_formatOK.mp4
Video format (container): mp4
Video codec: h264
Audio codec: aac
Frame rate: 25.00 FPS
Aspect ratio: 16:9
Resolution: 640 x 360
v_bitrate: 2.48Mb/s
a_bitrate: 214.11kb/s
channel layout: stereo
channels: 2
```

```
file: Voyage_to_the_Planet_of_Prehistoric_Women_formatOK.mp4
Video format (container): mp4
Video codec: h264
Audio codec: aac
Frame rate: 25.00 FPS
Aspect ratio: 16:9
Resolution: 640 x 360
v_bitrate: 2.38Mb/s
a_bitrate: 246.17kb/s
channel layout: stereo
channels: 2
```

```
file: The_Gun_and_the_Pulpit_formatOK.mp4
Video format (container): mp4
Video codec: h264
Audio codec: aac
Frame rate: 25.00 FPS
Aspect ratio: 16:9
Resolution: 640 x 360
v_bitrate: 2.41Mb/s
a_bitrate: 251.32kb/s
channel layout: stereo
channels: 2
```

Shareable Link: https://hub.labs.coursera.org:443/connect/sharedeafwwcot?forceRefresh=false