# Starpower AI Ads Product Data Creation Requirements

## Font Recognition

**Input:** Video (format, resolution, duration can vary)

**Output:** Data (format tbd -- what is most convenient for being able to pull from for an AI model? JSON or CSV?)

1. For each video
   1. Video Resolution **(width, height)**
   2. Video Duration **(second round to 5th decimal number)**
   3. Amount of Text Blocks in Video
   4. % of video with any text on screen **(by duration)**
   5. % of video covered by text **(% of video frame across duration)**
2. For each section (needs the section data block to be completed first, TBD)
   1. Input
      1. Section Name (section\_x)
      2. Section Time Start (**start\_time**)(second)
      3. Section Time End (**end\_time**)(second)
      4. Section Duration Time (iii - ii)
   2. Expected Output
      1. Text box still active from previous section? **(any block)**
      2. Amount of Text Blocks in Section
      3. % of video with any text on screen (by duration) during this section
      4. % of video covered by text (% of video frame across duration) during this section
3. For each text block in the video
   1. Timing information
      1. Time the text is first shown (**start\_time**)(second)
      2. Time the text is last shown (**end\_time**)(second)
      3. Duration time from entry to exit **(ii-i )**(second)
      4. Duration of text as % of video length **(iii / video length)**
   2. Text information
      1. Text capitalized and punctuated as shown
      2. WPM (number of words shown divided by duration time from entry to exit)
   3. Size (objective), rectangle **(width, height)**
   4. Size (% of video frame)
   5. Location (objective) **(box coors: top-left, top-right, bottom-right, bottom-left coordinates)**
   6. Location (section of video frame) (map to be included) **(frame indexes)**
   7. Font details
      1. Font style (closest match to font)
      2. Font size (in pt)
      3. Serif or Sans Serif
      4. Font weight (bolded, unbolded)
      5. Capitalization style
         * This is a **standard case** capitalization style.
         * This Is A **Start Case** Capitalization Style.
         * THIS IS AN **ALL CAPS** CAPITALIZATION STYLE.
         * this is a **no caps** capitalization style.
      6. Font color
      7. Font outline (yes or no). If yes,
         * font outline color
      8. font outline weight
   8. Is the text natural in video or added on top?

--- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

# Starpower AI Ads Product Data Creation Requirements

## Speaker Recognition

**Inputs:**

* Video (format, resolution, duration can vary)
* Audio Data

**Output:** Data (format tbd -- what is most convenient for being able to pull from for an AI model? JSON or CSV?)

1. For each video
   1. Video Resolution
   2. Video Duration
   3. Amount of Speakers in Video
      1. Each speaker’s ID
      2. Gender
      3. Est. age range
      4. % of video covered by each speaker ID (by duration)
      5. % of video covered by each speaker ID (% of video frame across duration)
   4. % of video covered by speakers (% of video frame across duration)
   5. Number of people: Include all detected people in video.
2. For each section
   1. Section Duration
   2. Amount of Speakers in Section
      1. Each speaker’s ID
      2. Gender
      3. Est. age range
      4. % of video covered by each speaker ID (by duration)
      5. % of video covered by each speaker ID (% of video frame across duration)
   3. % of Section covered by speakers (% of video frame across duration)
3. For each speaking segment
   1. Speaker information
      1. Speaker ID (if the same speaker is shown multiple times, should be the same ID)
      2. Gender
      3. Est. age range
      4. Speaking?
   2. Timing information
      1. Time the speaker is first shown
      2. Time the speaker is last shown
      3. Duration time from entry to exit
      4. Duration of speaker on video as % of video length
   3. Size (objective), rectangle (biggest object as represent)
   4. Size (% of video frame)
   5. Location (objective) (biggest object as represent)
   6. Location (section of video frame) (map to be included)
   7. What is shown of speaker
      1. Shoulders up
      2. Waist up
      3. Full body

**PIPELINE CODE:  
1. Font recognition:**

1. **Detect** pyscene\_score\_diff ([link](https://github.com/Breakthrough/PySceneDetect)) using mode **AdaptiveDetector.**
2. **Detect “text on screen”** using [Paddle OCR](https://github.com/PaddlePaddle/PaddleOCR)with pipeline:
   1. Detect raw text in each frame and filter text by **confidence\_score: 0.8**
   2. Tracking text objects with format below:
      1. Text origin.
      2. Normalized text: Lower all letter, replace all specials characters except [“d”, “$”, “/”, “@”, “.” ]. (these use for date, email, dollar unit)
      3. Get text difference score using **difflib.SequenceMatcher()**
   3. Merge separately text boxes as vertical and horizontal axes. Compare nearest edge of box (threshold = 30.0) and center distance(smaller than higher box’s edge)
   4. Tracking merge boxes by:
      1. Check text difference with threshold = 0.8 (**difflib.SequenceMatcher()**) and IoU of boxes during video frames.
      2. If it pass step i, check inner\_text or inner box. Inner\_text mean a part of string was detected separately: detect by checking continuously each letter in text with merge objects with text\_difference >= 0.8 and distance of 2 adjacent character < 2 letters
      3. Unique frame numbers of each text objects and remove any object has number of frame less than 5.
3. Detect scene: diff\_text\_sore and pyscene\_score\_diff => model detect scene change timestamp
4. Export output from step 3 to JSON file with requirements format above, with some condition notes below:
   1. Any field that has **“percentage”**  should save in range [0,1].
   2. **"Size-objective":** save the biggest face box size from step 4 in format [width, height] as represent value.
   3. **"Size-percentage-of-frame"**: average of face\_boxes/frame\_size.
   4. **"Location-objective"**: save the biggest face box coordinates ratio in range [0,1] with frame size.
   5. **"Location-sector"**: the same with Speaker recognition
   6. “**Text capitalized and punctuated as shown**”: use string function as: istitle(), isupper(), islower().
   7. Font information:
      1. **“Color”**: crop text box and use Pillow.Image.getcolors() to get the color with highest pixel.
   8. **“Bold”, “fontsize”:** use tesseract\_model.WordFontAttributes() (v3.05.02).
   9. **“Font name”:** using [Whatfontis API](https://www.whatfontis.com/API-identify-fonts-from-image.html)(free liscene has limit 200 requests/day)

**2. Speaker recognition:**

1. Detect scene: diff\_text\_sore and pyscene\_score\_diff => model detect scene change timestamp
2. Get face data: Scan face through video frames
   1. Use Retinaface[[model link](https://github.com/peteryuX/retinaface-tf2?tab=readme-ov-file#Data-Preparing)] detect face (confidence score >= 0.5)
   2. Tracking face box from step 2.a by IoU algorithm (threshold > 0.7 is the same location else otherwise save as separate face)

Output format: {

"face\_1": { #face key

"0": [ #frame index of face

358.06109619140625, #x-coor of top-left box

197.44886779785156, #y-coor of top-left box

472.6702575683594, #x-coor of bottom-right box

325.4707946777344]} #y-coor of bottom-right box

1. Filter straight face by face angle(yaw, pitch, roll): Scan each face from output Step 2
   1. Filter face keys has number of frame more than X = 0.5s x FPS of video
   2. Detect 468 keypoints by mediapipe.facemesh(min\_detection\_confidence=0.2, min\_tracking\_confidence=0.2, max\_num\_faces=50)
   3. Cal yaw, pitch, roll (in degrees) from 5 points have indexes [34,264,168,33, 263]
   4. Filter straight face by yaw, pitch in range: [-40,40] for yaw, [-45, 45] for pitch. If face key has more than **30%** straight face => legal.

Output sample:

"face\_1": {

"0": {

"box": [

358,

197,

472,

325

],

"yaw\_pitch\_roll": [

0,

-11,

6]}}

1. Combine the same faces and detect gender / age:
   1. Find 5 most straight face and best straight face by yaw, pitch, roll with straight face target yaw, pitch, roll in [0,10,0]
   2. Verify face and get speakerID:
      1. Extend the best straight face with padding = 20 pixels.
      2. Compare 2 face crop image by DeepFace.verify(detector\_backend="skip")[[model link](https://github.com/serengil/deepface/blob/fbc6d8d184f3877e2fdd739d974b1d9f457e8256/deepface/DeepFace.py#L64)]
      3. Update new best straight face image of existing speaker if current face has better yaw, pitch, roll.
   3. Detect gender, age[[model link](https://github.com/wildchlamydia/mivolo)] for each speakerID by average of *5 most straight face* from step 4.a

Output sample:

[ {

"speakerID": "speaker-1",

"gender": "Male",

"age": "(8-12)",

"frames\_infos": {

"0": {

"box": [358,197,472,325],

"yaw\_pitch\_roll": [0,-11,6]}}]

1. Export output from step 4 to JSON file with requirements format above, with some condition notes below:
   1. Any field that has **“percentage”**  should save in range [0,1].
   2. **"Size-objective":** save the biggest face box size from step 4 in format [width, height] as represent value.
   3. **"Size-percentage-of-frame"**: average of face\_boxes/frame\_size.
   4. **"Location-objective"**: save the biggest face box coordinates ratio in range [0,1] with frame size.
   5. **"Location-sector"**: find all sectors in frame of biggest face box corresponding in map here:
   6. **"Speaker-show**":Save % speaker body was shown in video in 3 status: “Full\_Body”, “Waist\_up” and “Shoulder\_Up”. Using model Mediapipe Pose to detect [33 poses](https://ai.google.dev/static/mediapipe/images/solutions/pose_landmarks_index.png) on the body and calculate average visibility score to detect. Step produce:

1, Scan each frame and check if any speakerID has existed in the frame by checking the frame index in **“frames\_infos”** from Step 2-4.  
2, Run mediapipe Pose to detect poses.  
3, Matching pose with existing speakerIDs(*5-f-1*) by check % face poses( 0 > 10-th poses) was in the speaker face box(detected by RetinaFace) => Select highest score as matched speaker.  
4. Detect % speaker shown-up by comparing average visibility scores with threshold 0.5, here are poses for 3 status:

* + - 1. Full body: poses [25,26,27,28]
      2. Waist Up: poses [23,24]
      3. Shoulder Up: any detected poses and matched speakerID

5, Remove the person out of the image by extracting the detected **segmentation mask** and replacing it with white color.

6, Run mediapipe pose with blanket image to clean memory of model with current images.

7, Repeat from step *5-f-2* with processed image from step *5-f-4*. Because mediapipe Pose only supports single person pose detection.  
\*Note: model mediapipe Pose could detect the same person although we replace white color from step *5-f-4*. So we need to add a condition to compare different current **seg\_mask** and previous mask (use MSE algorithm) to exit the loop.

* 1. **"Speaking"**: Detect speaker has speaking or not by using model [Lip-movement](https://github.com/sachinsdate/lip-movement-net):
     1. Get list 6 points from step (**2-3-b**) [index: 61, 62, 63, 65, 66, 67]
     2. Cal average distance of 3 pairs: 61-67, 62-66, 63-65 for each frame
     3. Detect lip-movement for each sequence (25-frame) with 1 of 2 modes: separately(sequence step = 25) or continuous (sequences step = 1).
     4. Get final result by calculating % speaking results..