Week 2 (September 7-September 14)

Tools used:

Elan (Video annotation application)

PivotHead Camera

Polar Heartbeat sensor

Bluetooth Distance and Accelerometer sensor

Goal:

Objective: In order to collect data from volunteers, we must ensure that data synchronization is as accurate as possible. In order to do so, experimentation must be done to get a feel for how “inaccurate” it can get trying to align video with heartbeat / other signals. For this week, experimentation will be done on trying to synchronize heartbeat and video signals and accelerometer and video signals.

* PivotHead:
  + How to Record
  + How to import video into Elan
  + Ensure proper migration of data
* Come up with alternative solutions to synchronizing data
  + Video - capture recording devices that keep track of Unix TimeStamp and have user input Unix into Python script
  + Record an on-going Unix tracker (on the web) and offset Video time with user input
* How to use Polar Heartbeat sensor
  + How to combine the data

Objective2: Previous HTML parser is not viable for certain Elan outputs, such as handling “TC” and requiring a “Start” tier. Create a new tab-delimited text parser that will more efficiently handle big data and allow a user-friendly approach to entering specific times and unix time / date

Results:

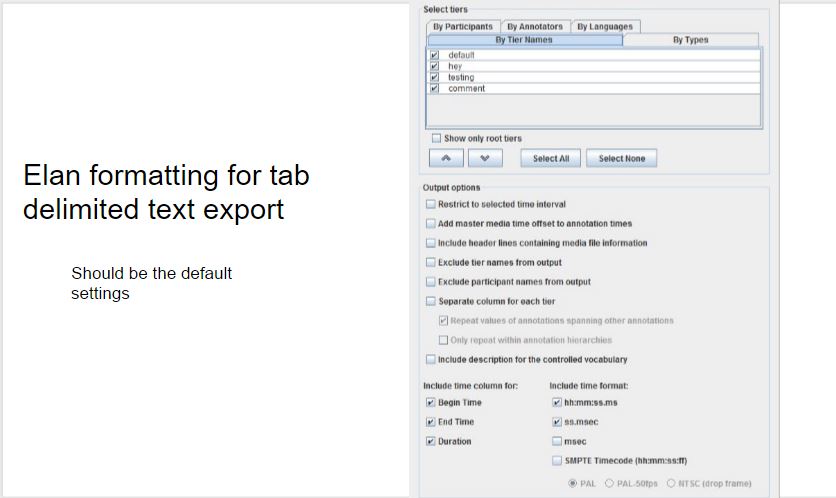
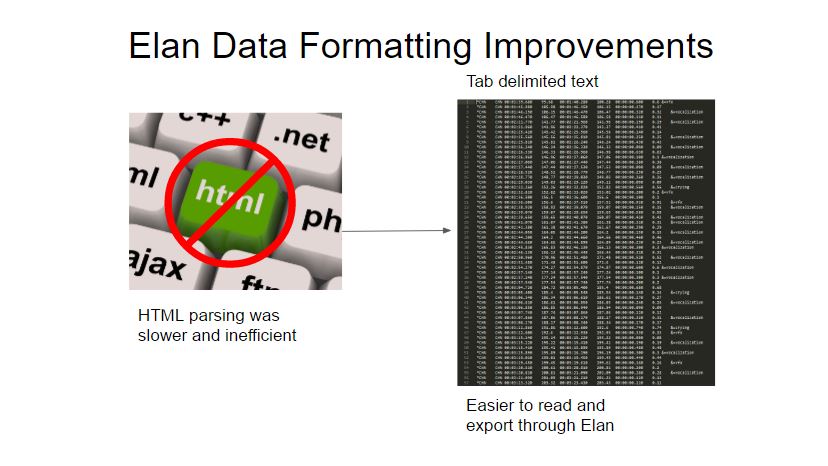
For objective 2:

1) tabparser.py is the new python script that will be able to take in a tab-delimited text file

2) I have added user input functionality where it will try to detect if a "Start" tier and annotation is in the text file. If it is, it will then prompt you to input the date and the time of day for that starting annotation (currently time must be in UTC time zone). If there is no start tier, it will then prompt you to indicate a "start" time (local to the video) and then prompt you for the date and time of day

3) The csv file will now contain all the times starting from the designated start time all the way to the end of the last annotation per tier. It was VERY slow at first, as I was creating a dataframe for each gap in between each annotation per tier, which, after running on your provided Elan file, I noticed that after 300 lines, it would take up to 10 seconds to move on; if projected forward towards the very end, it would have taken hours. I had fixed this using an alternative method, which I was able to perform the same function almost instantaneously.

4) I was able to get the csv files in the right format. I created a function that will identify unique annotations in the tier, create columns for each annotation, and label it as 1 if it is present given the time, or 0 otherwise. Currently, this function will be traversing through each timestamp (which is around 2.5 millions lines for the CHN, CHF, and FAN tiers), and it takes around 5 minutes to complete. I can try to come up with more efficient ways of doing this, but I do not know if this is necessary.



Objectives and Results

Goal: Previous HTML parser was not viable for certain Elan outputs because of handling “TC” tiers and requiring Elan input of a “Start” tier. Create a new tab-delimited text parser that will efficiently handle big data and allow a user-friendly approach to entering specific times and unix time / date

Results:

1. Tabparser.py is the new python script that will be able to take in a tab-delimited text file
2. Script will automatically detect “Start” tier. If not, it will prompt user to input a “start” time, followed by a date and time of day for unix time stamp
3. CSV file will now contain times that span from the designated start time until the last annotation for each tier
4. Each unique annotation has a unique column for each tier. This will allow user to visualize annotations by using a time-series graph

How to use tabparser.py

1. Exported tab-delimited text file using the guidelines (should be default settings) in Elan
2. The text file must be in a folder txt/yourfile.txt with respect to the tabparser.py
3. Indicate the name of the file in the method startParse(yourfile.txt)
4. If “Start” tier is in Elan, then give Date and Time (prompted by script)
5. If no “Start” tier is created, then give the local time of offset followed by Date and Time