### PathML Data Munging/Extraction Documentation

This document outlines the scripts and procedures for converting raw Noldus files from different studies into a normalized and standardized format for training PathML models. It assumes the reader is proficient with cloud-based Linux servers.

The process heavily leverages Miller, a declarative data transformation tool primarily intended for CSV files, analogous to the popular jq tool but for CSV.

#### tl;dr

The high level process is:

1- Prepare the Noldus file by converting to CSV and naming unnamed columns 2- Identify unique values that need to be extracted from behavior and modifier columns 3- Prepare a keyfile that specifies how to transform the noldus tagging into our normalized taxonomy 4- Use Miller to create a filelist readable by ML pipelines (NVidia DALI) 5- Convert videos to a format readable by ML pipelines using ffmpeg 6- Repeat steps 1-5 for each study 7- Combine all filelists into a single file 8- Split filelists into balanced train, test and evaluation folds

#### Key Concepts

##### Concept 1: Noldus Files

Noldus files are Excel files containing tagged “behaviors” which may have “modifiers”. Combining these attributes maps to our standardized taxonomies.

There’s only a few columns that are particularily important in most Noldus files that must be extracted. First, the Observation column indicates the video that the line is tagging. This may be a singular column or composed from a few different columns. Second, is the Behavior and Modifier columns.

Another important consideration is identifying when behaviors are active in the dataset. This depends on the type of Noldus file you’re working with. 1. “Standard” - Contains start point, duration, and State start/State end in Event\_Type. 2. “Second by Second” - Each row is 1 second of video. Start may be explicit or implied by row number.

For Standard formatted noldus files, the critical start and stop origination columns are: Time\_Relative\_sf and Duration\_sf which are seconds stored as floats (sf – second floats); analogous to a raw timestamp. The stop time can be easily calculated as $stop = $Time\_Relative\_sf + $Duration\_sf. Time\_Relative\_sf can simply be renamed as start.

For Second-by-Second, use Time\_Relative\_sf if it exists, otherwise track row number and video using the NR metavariable. Increment start/stop by 1 each row, resetting to 0/1 for new videos.

##### Concept 2: Keyfiles

Noldus data points can be labelled with a variety of tags; these tags are not consistent across studies. This could include tags that are misspelled in one source by not the other, inconsistent spacing, different qualifiers, or entirely unequivalent tags. These all need to be converted into our standardized taxonomy. Keyfiles are crucial for standardizing and harmonizing data across different studies.

Keyfiles are used as join tables by Miller to drop irrelevant rows and transform them to the target task. They are simple csv files with 2 or more columns. The first column is the join key. The second is one or more target tasks. Here’s a simple example:

Behavior,Task1  
LA- kneeling/ squatting,sedentary  
LA- stand,active  
LA- stand and move,active  
LA- stand and move with unidentifiable upper body movement,active  
LA- stand and move with upper body movement,active  
LA- stretching,active  
SB- lying,sedentary  
SB-sitting,sedentary  
SP- bike,active

Keyfiles are manually created, first by extracting all the unique values of a particular column or modifier and then assigning them to the defined task one-by-one, deleting lines that should be dropped.

For tasks that have some classes dropped in certain boarder-scoped tasks but retained for more specific ones you can use the label private. Those examples will be filtered out by the pipeline later.

##### Concept 3: Filelists

To be fed into our machine learning classifier, we expect a “filelist”. Filelists are *space delimited* files with one video fragment per line that indicate a start and stop window and a specific class that that fragment belongs to. So it looks like:

video label start stop

The labels must be integers, and the start and stop columns are timestamps, in the same seconds floats format as Noldus. The video should be an on-disc location: typically the post-conversion files.

##### Concept 4: Chi2 Folds

Machine Learning tasks are typically built into three steps: first is the training itself, where the model generalizes off of example data that is fed to it. Second is the testing step, where the machine learning model frequently self-checks against new data it isn’t trained on and reports its accuracy. There is often a prolonged iteration step back and forth between those two steps until the developer achieves sufficiently good performance to continue; sometimes there can be an implicit bias with the testing fold that can build up during this iteration process.

This is why there is another round of testing, called evaluation. This uses yet another novel fold of data and is to simulate the expected behavior that the pipeline will have on truly novel data.

In typical machine learning problems, you can build these folds by simply shuffling together the data and splitting them up into a 80%/10%/10% split. However, our video fragments have linear dependency: for example, a 2 hour video of someone mostly sitting on their couch watching television has many similar frames. Shuffling those naively would result in spuriously high accuracy while training the model due to leaking these features across the different folds. Therefore, it would fail to generalize and accurately predict novel data.

##### Concept 5: Directory Layout

In order to orientate, here is a short description of the most important directories on the PathML box.

/data or /mnt/durable in a container /datasets <- video files /fps2\_384 <- converted videos; in this case, 2 fps and 384x??? resolution /Wisconsin <- original raw videos for each study /PathML\_Phase1 /AMstudy /ACT24study /Ground\_truth\_Aim1 <- noldus files /ACT24 Study /AM Study /PathML Study /Wisconsin /training\_data <- keyfiles and filelists /phase2 <- prepared task directories for training /1 <- for task 1, repeat for other tasks (2a, 2b, 3…) training.txt,testing.txt,evaluation.txt <- basic filelist training\_rolled.txt,testing\_rolled.txt,evaluation\_rolled.txt <- filelists with adjecent second-by-second lines collapsed  
testing\_shuf.txt <- testing\_rolled filelist shuffled. improves accuracy. unnecessary for non-training folds. /filelists <- intermediate filelists for each study before they are merged + split /keyfiles <- keyfiles for filtering the data and tagging the noldus data into our final taxonomy labels /mediapipe <- csv for storing skeleton data from mediapipe

#### Process In Detail

This process is describing what steps might be taken to prepare a fresh Noldus dataset for training a machine learning model.

##### Step 1: Prepare Noldus Files

Miller cannot read Excel files, nor does it work with unnamed columns very well. Converting the Excel files can be done using your normal desktop spreadsheet manipulation program, or done in the CLI with the following command:

find . -name "\*.xlsx" -exec sh -c 'ssconvert "$1" "${1%.xlsx}.csv"' \_ {} \;

Then, open it up and name the very first column as 0 or any other throwaway name.

###### Step 2: Unique Behavior Values

In order to write your keyfile, you need to figure out how the behaviors were originally tagged in the Noldus files. In theory, this should be relatively consistent but small typos exist in the observation data and other inconsistencies make this an important step to repeat and carefully review.

In general, the critical columns are Behavior or a Modifier\_ column. You can list out all the values of a given column with the following script, adjusting it slightly to account for the directory (DIR) and the column (Behavior in this case):

find [DIR] -name '\*\*.csv' -print0 | xargs -0 \  
 docker run --rm -v /data:/data jauderho/miller --csv \  
 uniq -f Behavior

Copy these values into a blank text file, which is the start of your keyfile.

###### Step 3: Create Keyfile

The keyfile leverage the join command in Miller which allows you to filter out irrelevant rows for a given task as well as map an input label to a normalized target label. See the concept above for more details. The basic process is to use the unique values in the previous step and then add in the target labels as new columns, one for each sub-task.

Some tasks are dependent on multiple columns; for example, intensity can be part of a modifier that outright describes the intensity, or it could be implicit in the Behavior column; eg, sitting is sedentary. My recommendation there (and what is done in the scripts) to add “sitting” into your keyfile and *copy sitting into the Modifier column, overwriting any existing values (which shouldn’t be the case if it was properly labelled anyway).* This is much simpler than trying to do a multi-join condition.

###### Step 4: Create Miller scripts

Miller scripts usually have the following structure: 1: concatenate all relevant csv files 2: cut relevant columns 3: filter out rows (eg, only include rows with a “State start”) 4: transform start and duration columns into a “start” and “stop” 5: join your behavior column with your keyfile 6: output only relevant columns into the filelist format

The full documentation is here: https://miller.readthedocs.io/en/6.12.0/

Miller is currently installed as a docker container, which means it doesn’t have to be installed separately. A convienent alias for miller (and what is used in this documentation) is:

alias mlr='docker run --rm -v /data:/data -v /mnt/ephemeral:/mnt/ephemeral jauderho/miller'

Some scripts might have the weird-looking “pwd”: this is to expose the current working directory to docker, which doesn’t have access to that by default.

###### Step 5: Convert Videos

Videos need to be re-encoded into a specific orientation (384 pixels on the long edge). This is done using ffmpeg. In addition, we downsample the video stream to a lower fps (currently, 1 or 2). We also crop off a little bit of the farthest left and right part of the image, with the assumption that the observation study is focused on the center and the edges often contain irrelevant information.

Finally, we need to index the video stream with something called “keyframes” which are full-resolution frames that can be trivially seeked to. This is essential for being able to seek between different file lists. This is the general command, adjust the initial location [DIR] to target the study you need to convert

find [DIR] -name '\*.mp4' -exec bash -c "docker run --rm --runtime=nvidia -v /data:/data -v /mnt/ephemeral:/mnt/ephemeral jrottenberg/ffmpeg:5.1-nvidia2004 -hwaccel cuvid -hwaccel\_output\_format cuda -c:v h264\_cuvid -y -crop 0x0x140x140 -i {} -vf fps=fps=2:0,scale\_npp=384:-1:force\_original\_aspect\_ratio=decrease -c:v h264\_nvenc -g 8 -preset slow /mnt/ephemeral/fps2\_384/\$(basename {})" {} \;

###### Step 7: Merge Studies

After creating multiple per-study filelists, you need to merge them together into one large file, while normalizing inconsistencies, and dropping private rows. This can be done with a relatively complex miller script which is contained in the script directory as merge\_filelists.sh. Note that this needs to be run **per task**. See towards the end how i have task# capitalized. change those to what it should be. eg, task2a.

##### Step 8: Create Folds

###### 8a: Generate Splits

See previous concept about Chi2 difference. There is a julia script contained in the script folder called splitter.jl. This outputs various suggestions for how to create folds from the merged filelist file created in the previous step. This will print out 5 possible candidates. In general, for well-balanced datasets where all classes have high incidence and occur in many videos, the first one is clearly the best.

However, for datasets with uneven class incidence, you may need to pick a “worse” candidate that has better distribution of the minority classes in all three of the train/test/evaluation folds. Prioritize a good # of train candidates paired with roughly 10% as many examples in the test fold. If possible, have a good evaluation fold count but that is the least important when trying to balance larger, more complex taxonomies that might have multiple underrepresented classes. Save these files in a “\_splits.txt” file.

###### 8b: create task folders

From the raw splits, you must create the appropriate task folders from all \_splits and \_filelist\_final files. The contained script create\_splits\_from\_split\_files.sh demonstrates how to do this and has documentation how to use it.

#### Appendix

###### NVMe drive

Amazon AWS typically uses remote NAS drives called “EBS” – Elastic Block Storage. The advantage of this is that they are easy to back up and move between different instances. However, they are much slower because data must be retrieved across the warehouse rather than physically plugged into a colocated drive. Some AWS instances, including the G4 instance the box is currently on, offers “ephemeral” storage that is associated with the physical device. This storage is not backed up and lost each time the instance restarted. On the other hand, it is substantially faster.

For this reason, we move our videos to NVMe drives for training, while keeping a back up copy on our slow EBS drives. This is mounted to /mnt/ephemeral. Inside of a Docker container, we remap /data to be /mnt/durable to make this distinction obvious. Intermediate caching files (such as a swapfile) are also put on the ephemeral drive.

###### NVidia DALI

Currently for training we use NVidia DALI to load GPU tensors from our videos. This is mostly a programming detail. Here’s the documentation: https://docs.nvidia.com/deeplearning/dali/user-guide/docs/operations/nvidia.dali.fn.readers.video.html