A ZOOM FILTER FOR APPLAUSE AND LAUGHTER

Meeting 15.12.21

Where we left off

Practical Example

- average meeting length: 56min
- average laughter length during meeting: 2:06 min

Mistake:
*tot_transc * recall = tot_pred * prec*

5:03 -> 303s
2:06 -> 126s

303s * 0.2044 = ~62s
126s * 0.3730 = ~46s

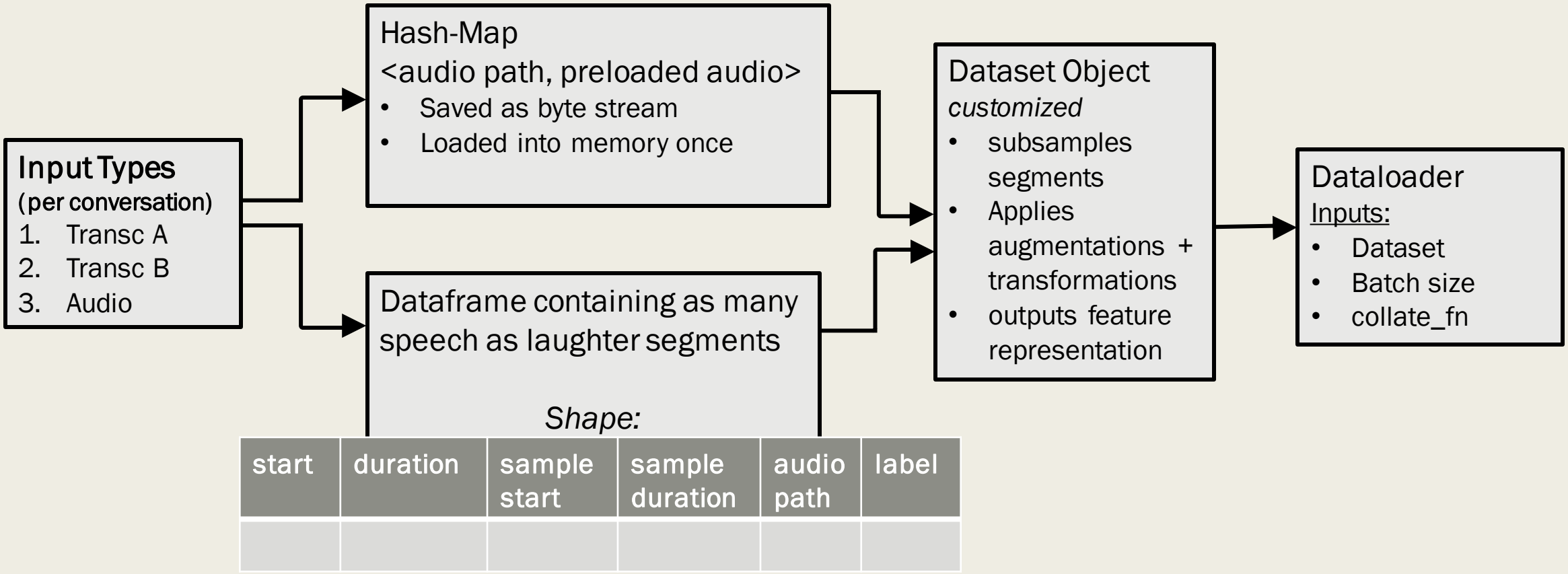
	new method		Laughter in [min:sec]		
threshold	precision	recall	predicted	actual laughter	noise
0.2	20.44%	37.40%	5:03 min	1:02 min	4:01 min
0.4	53.84%	20.68%	1:00 min	0:33 min	0:27 min
0.6	79.68%	8.92%	0:14 min	0:11 min	0:03 min
0.8	90.44%	3.22%	0:04 min	0:04 min	0:00 min

RTF – Real Time Factor

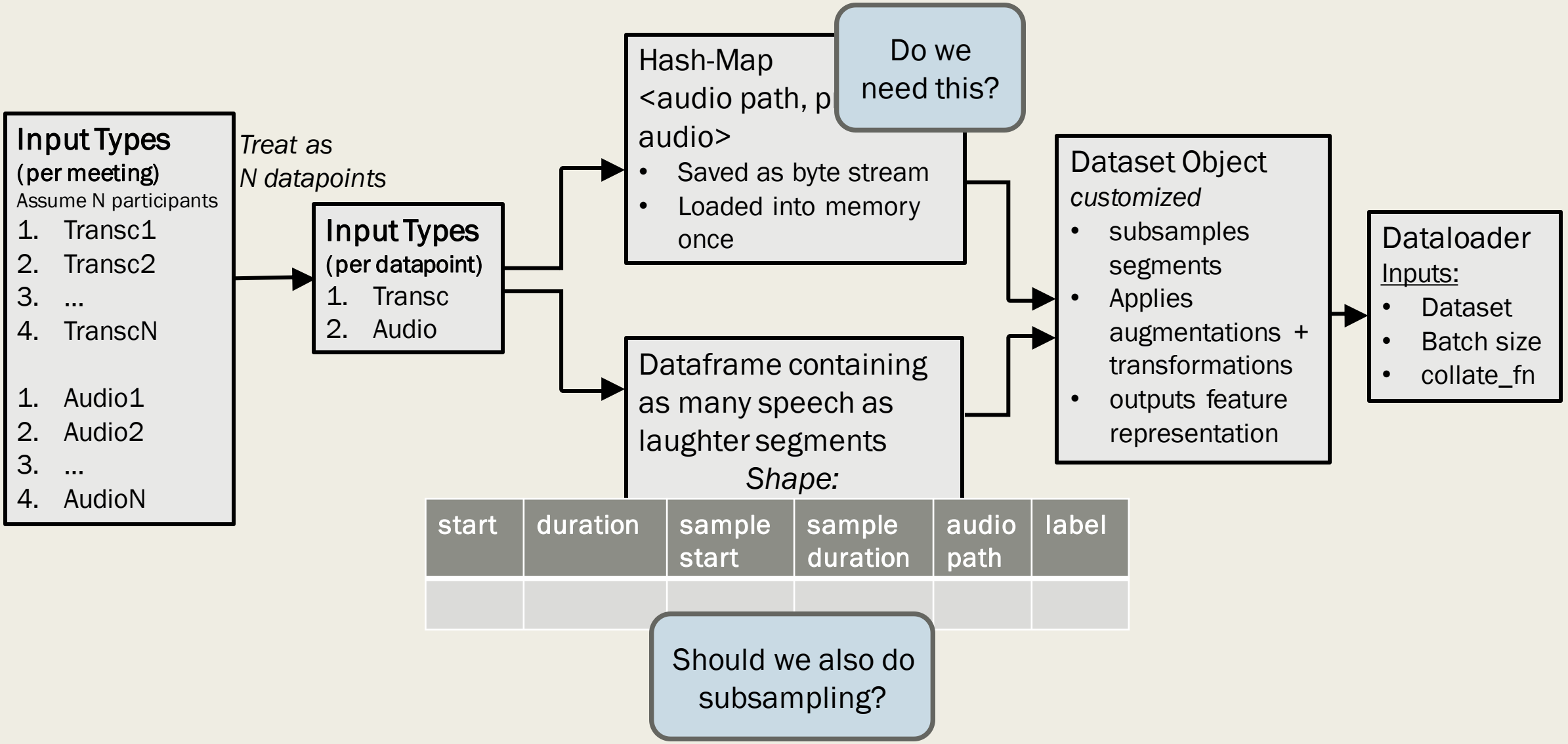
- Calculated RTF or current system on different machines
  - Reference value for future models
  - RTF = (time to process the data) / (duration of the data)

Audio Duration	Iterations run	Average RTF
CPU - i5-6500 CPU @ 3.20GHz		
3s	20	1.31
30s	20	1.41
120s	10	1.49
CPU AT - AMD EPYC 7302 16-Core Processor		
3s	20	0.63
30s	20	0.84
120s	10	0.81
GPU AT - NVIDIA GeForce GTX 1060 6GB		
3s	20	0.14
30s	20	0.10
120s	20	0.10
300s	10	0.10

Data Pipeline used by Gillick et al.



Data Pipeline for our project



Questions about the implementation

- What's the point of global step?
  - why use this and a while loop instead of epochs?
  - train_df is created every time – seems inefficient
- collate_fn

Next steps

- Correct Evaluation Method
- Create Data Pipeline
- Train Gillick et al.'s Model on ICSI (subset?)
- Train and evaluate more efficient models