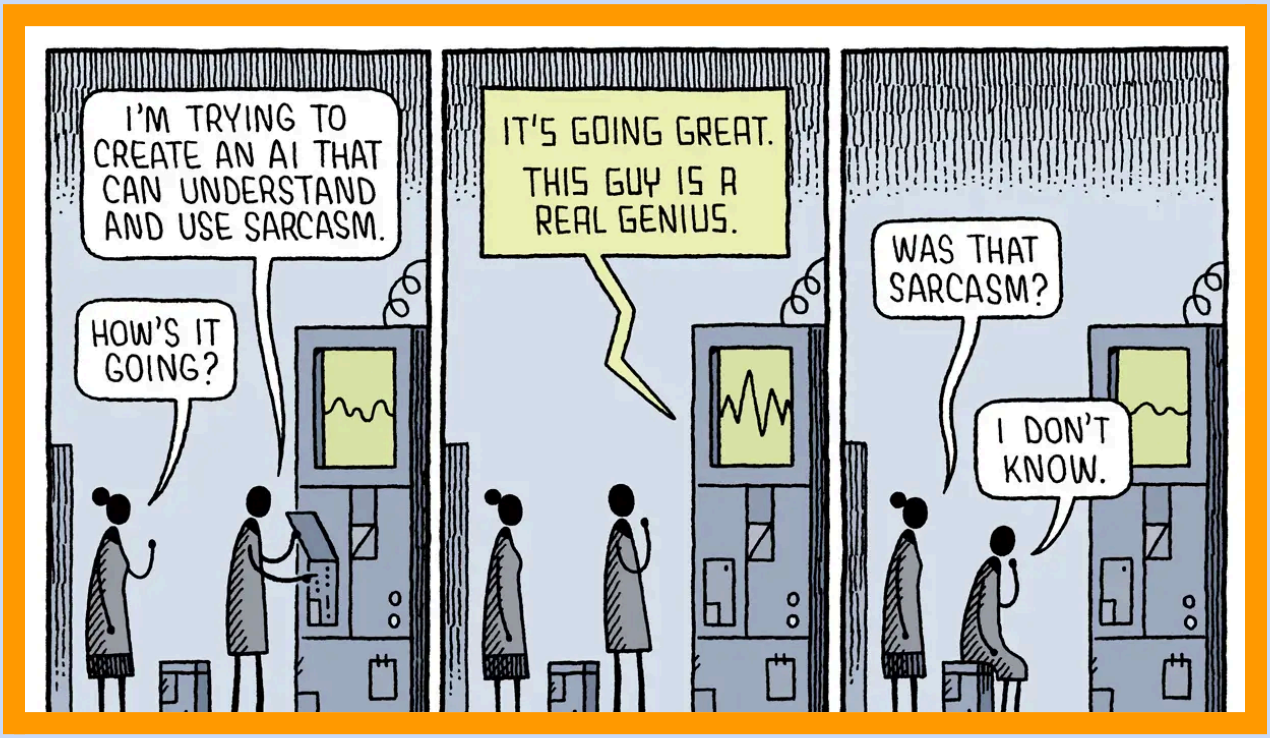


2025-6 Andi's ASP Logbook

Image by Tom Gauld

💜 = meeting w mentor

LINKS (click buttons to go to doc)

Research proposal = research proposal hehe
Background research = bgk reseach
Lit review of studies that have used MUStARD = MUSTARDDDD
Baseline Features I used = baseline features
Oral presentation slides = proposal oral presentationANDI
Methodology = Methodology

Aug+September 🌱	☑ ~~Finish research proposal draft~~

Aug 12 💜
Notes written by Dr A.
	Meeting Notes
	• Topic of project: detect sarcasm in online content (videos)
	• Method: Create code to detect sarcasm based on main acoustic and lexical properties using youtube videos from comedy shows.
	• Project timeline based on my discussion with Dr. Garcia. Please verify with your teacher:
	  • mid Oct = present your proposal
	  • end of Feb = finish project
	  • Mar = science fair
	  • Apr = city science fair (if successful)
	  • May = national science fair (if successful)
	  • end of May = submit paper
	• We'll be meeting every two weeks on Thursdays, 4:30pm alternating between online and in-person meetings.
	• Our next meeting will be on Zoom on August 28th, 4:30pm. I will be sending you a calendar invite with the relevant information.
	• You need to create a Google Drive folder where we can share materials. Please use the email ageliki.athan@gmail.comfor this, but don't email me there, as I don't monitor that email account.
	• You'll prepare for the next meeting:
	  • A file with a list of sources you've identified so far, indicating where you've read them or not.
	  • A file that has a rough research plan based on

	what you presented to me today. ● I'll talk to our PhD student who has worked on a similar project for her Honors and send you her thesis.

Aug 29	The next meeting is on **Sep 3**
	- Took notes for proposal intro
	- Lots of sources on sarcasm on social media talk only about text posts/comments
	- 6 papers due:
	  - Research proposal
	  - Intro
	  - Methodology
	  - Results
	  - Analysis
	  - Final paper
	- Proposal due mid Oct
	- Todo:
	☑ ~~Finish first draft of intro for the research proposal and indicate gaps~~
	☐ Find more information on stress, HNR, one third octave spect
	☐ Email Dr Garcia summary
	☐ Read the pitch accent paper

Aug 31	Add research on the difference between acoustics on social media vs rea life ☑ ~~Finish writing introduction of proposal~~
Sep 1	Added notes on the function of sentiment analysis ☑ ~~Proofread introduction~~ ☑ ~~Start working on methodology~~
Sep 2	☑ ~~Add objectives, research question, and significance of results to proposal~~

Sep 3 💜	Go to the UofC library at least once before the 14th
	● ASP class
	  ○ Research proposal -
	  ○ Find reference manager for google doc: paperpile
	B4 meeting
	☐ Skim the pitch accent paper
	☐ Organize sources into done, not done, and need to get access
	<u>Meeting notes</u>
	● …Need to meet at least 1/wk…
	● Pitch accent paper very technical
	● Research proposal and Introduction by end of nov
	● **OCT 16 Proposal is due!!**
	● Email summary 1/wk
	● HNR

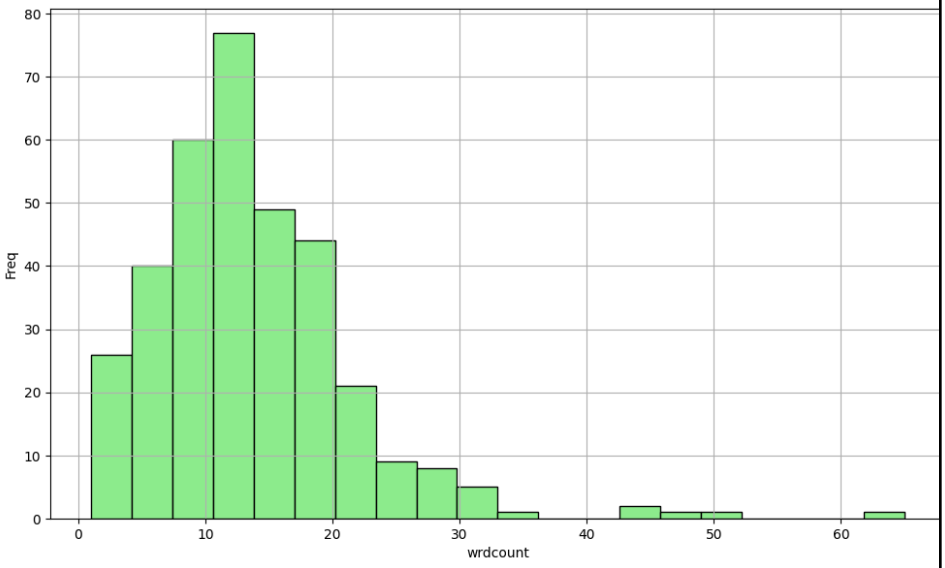
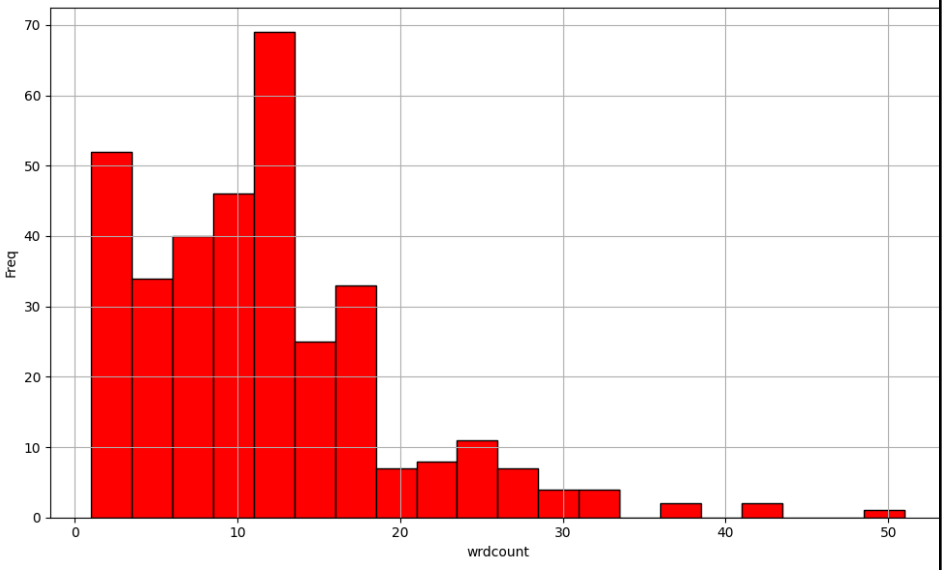
	○ Creak is produced with low tones ○ Regular waveform = ○ Irregular waveform ● ⅓ octave spect ● Declination = f0 decreasing as the sentence progresses ● * = stressed syl ● Inclusion criteria: ○ At least 5-10 minutes ○ At least 10 videos that fit criteria ○ 5050 women men from NA ■ Starndad american english ○ Included transcripts ○ Age??? Not sure yet ● Todo after: ○ Work on inclusion criteria ○ Read MUSTARD methodology ○ Read papers on tone etc ○ Write general points for lit review ■ Specific topics ■ “subheadings” Make sure to CC Ms Kale in the next email!
Sep 4	☑ ~~Send email to Dr A on notes taken~~
Sep 5	☑ ~~Start reading on methodology of MUSTARD~~
Sep 7	☑ ~~Read about kappa and and how to measure interrater reliability~~
Sep 8	☑ ~~School: read 1 paper on another paper classifying sarcasm~~ ☑ ~~Home: read 1 paper on sarcasm phonology~~ ☑ ~~Finish notes on Fleniss kappa statistic~~ ☑ ~~Home: Start brainstorming ideas for inclusion criteria (make a table with the criteria + justification)~~
Sep 9	☑ ~~School: read pitch accent paper!!!~~ ☐ Get more papers from the yeah right paper ☑ ~~Home: write points for lit review~~ ☐ Home: add to inclusion criteria ASP class ● Rough draft of proposal due end of sep
Sep 10	☑ ~~School: read 2 paper on sarcasm phonology~~ ☑ ~~1/2~~ ☑ ~~2/2~~ ☐ Finish justification table for inclusion criteria ☐ Finish lit review points
Sep 11 💜	**Oral presentation is at the end of october**

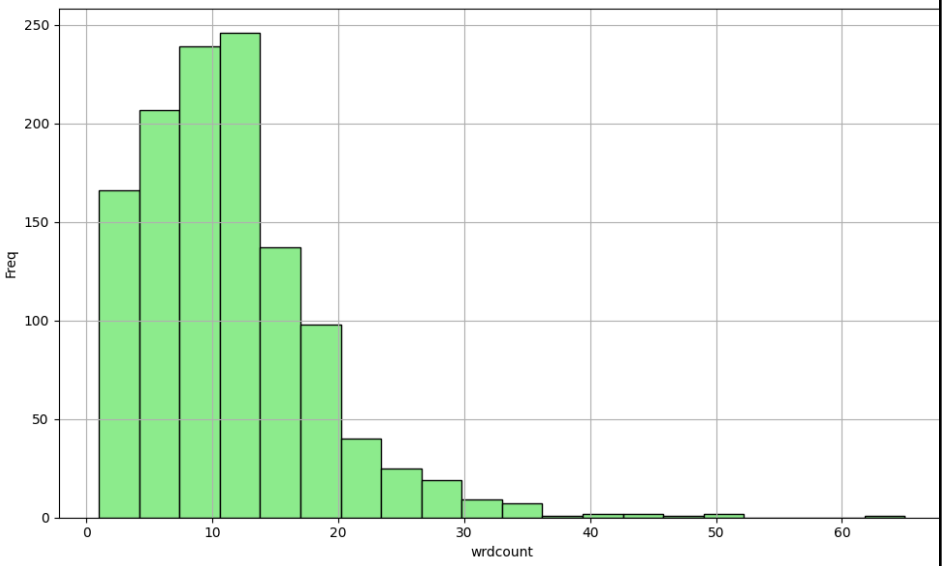
	Proposal introduction due at end of september **Full proposal due oct 16th** ● Installed paperpile today B4 you meet her ☐ Solid idea on inclusion criteria for videos ☐ Read some papers on sarcasm ☐ Write points for the lit review Meeting notes ● Ask for help on features used in yeah right paper ● Used comedy/satire videos instead ● Context all text?? No prosodic features? ● Inclusion criteria arbitrary? ● Cite other researchers that have used the dataset ● Table of paper that have used Mustard ● Montreal forced aligner ● Global acoustic features < local features ● How to remove laughter ? ○ Timestamped laughter ● Try to find other data if this one is not good? ● Older studies b4 2018 Next week Decide on database (Tues) ● Send summary of mustard database ● How to fix laughter Then make data table of the database used Data split : dependent on size
Sep 14	Appointment: library at 2-3pm Booked time at UofC library to get access to papers: ● Listener's ratings and acoustic analyses of voice qualities associated with English and Korean sarcastic utterances - ScienceDirect ● Context and Intonation in the Perception of Sarcasm: Metaphor and Symbol ● Lower, Slower, Louder: Vocal Cues of Sarcasm | Journal of Psycholinguistic Research ● Verbal Irony Use in Face-To-Face and Computer-Mediated Conversations - Jeffrey T. Hancock, 2004 ● Vocal Features of Conversational Sarcasm: A Comparison of Methods | Journal of Psycholinguistic Research
Sep 15	☑ ~~School task: record info on dataset~~ ☑ ~~Read mentor's email~~ ☑ ~~Download dataset~~

	☑ ~~Check to see if laughter is timestamped~~ ☑ ~~Explore~~ ☑ ~~Decide to use it or not~~ ☑ ~~Write report to DR and Brooklyn~~ New search prompt: “local features” Proposal done Oct 16 Oral presentation end of Oct Determine measure of success Send drafts to Dr Garcia and Kale NOTES ON MUSTARD: Laughter: seems to be present at the end of both sarcastic and sincere phrases, but more info is needed (manually annotate?) - Doesn’t seem to be overlapping words\ - Some phrases are a single word long - Bias of phrase length? Sarcastic phrases longer? Need to measure - Laughter NOT transcribed .wav format 690 total videos, Even distribution of TV shows in the two groups, but uneven distribution of shows in general (Friends most numerous) Email: Hi Dr Anthanasopoulou and Brooklyn, ● I downloaded the clips from MUStARD and their labels. As for the laughter, I found both sarcastic and sincere clips had laughter at the end (whew), and it doesn’t really overlap the phrase. I think we can find a way to cut the clip right before the laughter (the laughter is a lot louder than the sentence, maybe cutting when the volume increases suddenly?) ● There is some noise and buzzing especially in the older shows. If I haven’t gotten this wrong, this will be a lot quieter if we normalize it by show? ● Some clips are a single word long excluding the context ● I listen to a couple of clips, I know multimodal makes it more complicated, but it's pretty hard even for me to tell if something’s sarcasm. Could we consider some context or text base features? ● Overall, I think this is a good dataset to work with. It doesn’t have the laughter only on sarcastic clips which is what I was worried about
Sep 16	☑ ~~School: respond to email~~ ☑ ~~Home: start on MUStARD table~~

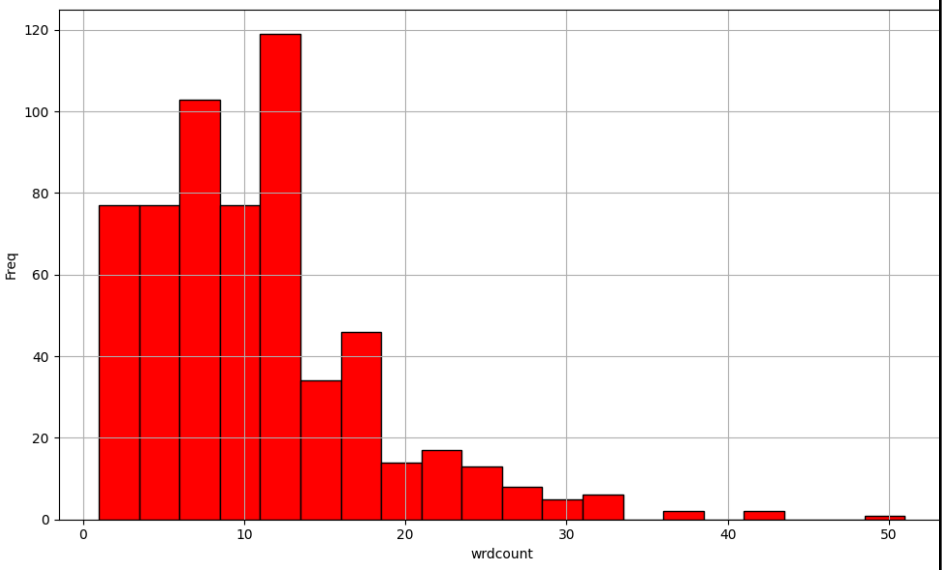
	EMAIL: There are 690 datapoints, evenly divided into sarcastic/sincere. The dataset was originally a .json that I converted into .csv, so it contains a column with the number of a specific clip (ex. “1_60.wav”), the sarcastic label, the speaker of the phrase and also the speakers of the context sentences before it, the actual transcriptions of the phrase and context, and the name of the TV show. Part of the code will probably include finding the label of a data point, finding the name of the file, then finding it in the clips folder. As a side note: I have been coding with Jupyter Notebook for a few years now and am the most comfortable in it. What coding environment is the industry standard? VSCode?
Sep 17	☑ ~~Respond to email~~ ☑ ~~Work on table~~ ☑ ~~Add 10 papers to table~~ ☑ ~~Review 4 of them~~ ☑ ~~1/4~~ ☑ ~~2/4~~ ☑ ~~3/4~~ ☑ ~~4/4~~
Sep 19	☑ ~~Good morning: Find more papers for MUStARD~~ ☑ ~~Respond to email~~ ☑ ~~1/2~~ ☑ ~~Finish summary of all papers~~ ☑ ~~Find example summary table~~ ☐ Check summary of papers and send to dr ASP ● Find project title ● Project classification ● Research Q Email: The label is only sarcastic/sincere. 10 utterances are questions and the rest are statements. For excluding utterances, what’s the minimum number of words an utterance needs to be? About 75% of utterances are 84 characters and below, about 25% are between 3-30 words (one or only a few words). Removing 25% of the datapoints is a lot, so maybe we only remove the utterances with 1 word (33 datapoints)?

Sep 23	☑ ~~Finish good copy of MUStARD table~~ ☑ ~~Send to Dr~~ ☑ ~~Get started on proposal~~
Sep 25 💜	<u>To talk about:</u> • Find project title • Project classification as experiment or innovation? • Research Q • Ask about good papers to understand legendre polynomial expansion ☑ ~~Take notes on "I did the right thing" paper~~ ☑ ~~Edit research proposal intro~~ Results: For next week: get word distribution histogram Look at ++ Finish reseach proposal intro What new things benefit models on MUStARD
Sep 27-29	☑ ~~Download MUStARD and MUStARD++~~ ☑ ~~Get length of utterance histogram and send to dr~~ ☑ ~~Sarcastic~~ ☑ ~~Sincere~~ ☑ ~~Find MUStARD++ and send info~~ ☑ ~~DUE ON MONDAY: Modify current proposal and add citations!~~ ☑ ~~DUE ON MONDAY: Send email to dr and brooklyn on MUStARD++ info~~ ☑ ~~Install jupyter notebook~~ Email Hi Dr Anthanasopoulou and Brooklyn, I downloaded the MUStARD++ dataset. It has 1202 utterances and is completely balanced. It labels the type of sarcasm but is also just labels if it is sarcastic or not, so I think this is also a good choice for the dataset instead of MUStARD. It also labels the show and speaker of each utterance and has the audio. On the ++ dataset, the distribution for word count is very right skewed and about the same for both sarcastic and sincere clips. I'm not sure how this determines what utterances get removed though, because it's right skewed, should there be a max word count? Sarcastic

	Sincere
MUStARD	Sarcastic  Sincere 
MUStARD++	Sarcastic



Sincere



WARNING: TO OPEN JUPYTER NOTEBOOK, START COMMAND PROMPT WITH "py -m jupyter notebook"

START ALL LINES IN COMMAND PROMPT WITH "py -m"

WARNING FOR HISTOGRAM:

Sarcastic utterances are in GREEN

Sincere utterances are in RED

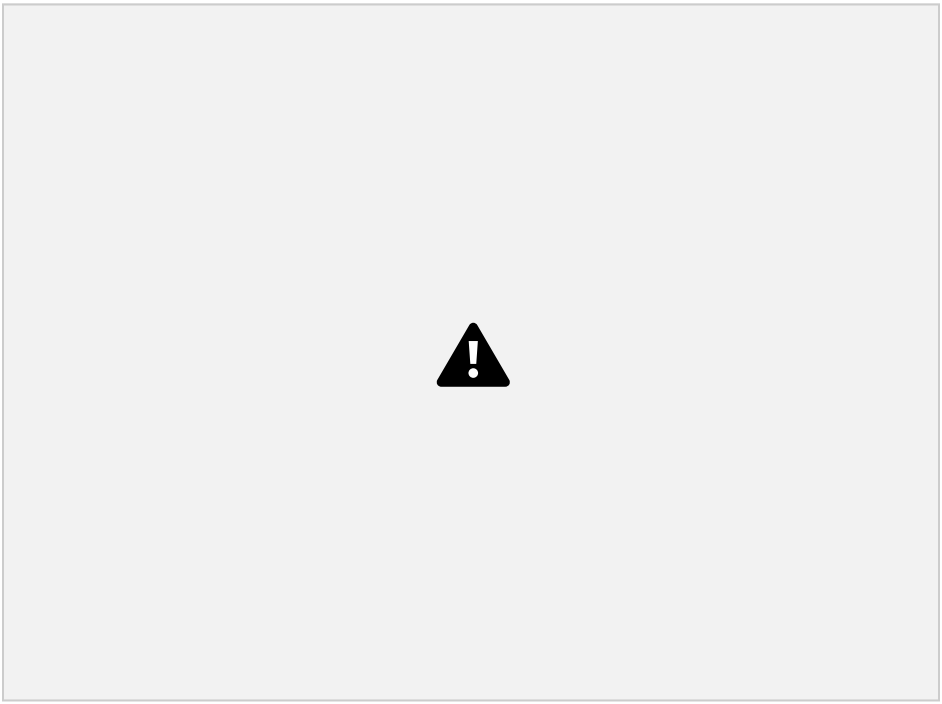
TFDL to borrow:	• Multimodal Sarcasm Detection Method Using RNN and CNN |

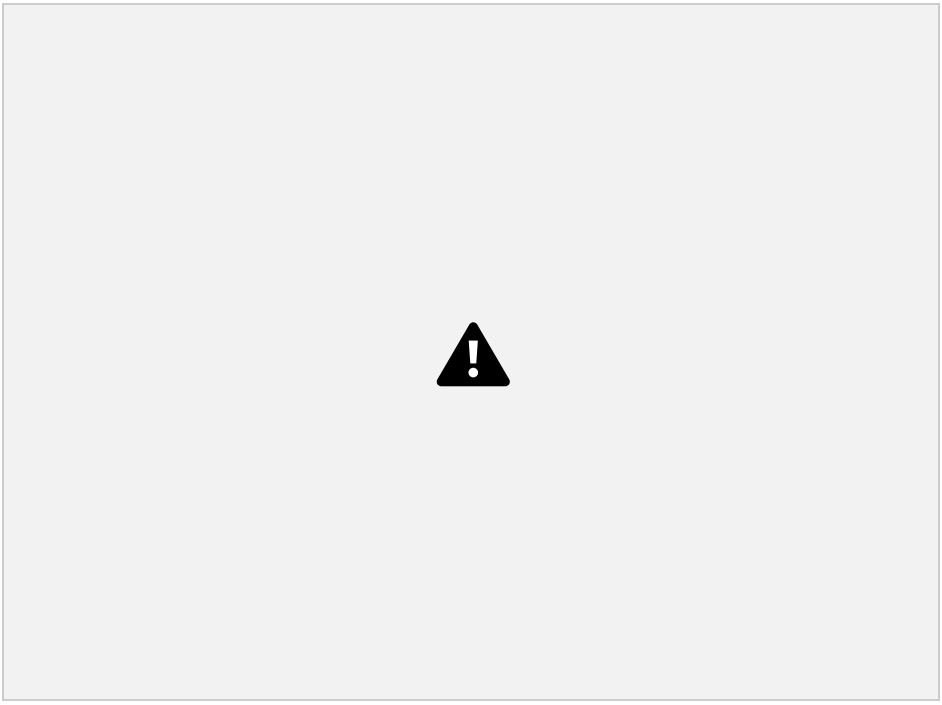
	[IEEE Conference Publication]
October🎃	☑ ~~Finish research proposal~~
Oct 1-2	☑ ~~Finish bgk section of proposal!~~ ☑ ~~Email: "Review Ms Kale"~~ ☑ ~~Check w example proposal~~ ☐ Expand research on applications ☐ Start methodology of proposal ☑ ~~TITLE!!!~~
Oct 3-5	☑ ~~Read brooklyns paper on lengerede coff up to p. 22~~ ☑ ~~Read lengerdre starting from 75 and 97~~ ☑ ~~Focus on the methodology !!!~~ ☑ ~~Add the necessity of project into the proposal intro~~ ☑ ~~Additional app: p76 of lengendre paper, home assistants~~ ☑ ~~Add on why log reg is important to proposal intro~~ ☑ ~~email brooklyn and dr a~~ ☑ ~~Proposal obj~~ ☑ ~~Proposal methodology~~ NEXT EMAIL: • Ask about how to interpret fig 2.1 and 2.2 on p 21 of lengerede paper • I read the thesis, methodology of her paper is not much clearer • Sentence vs phrase • Wav2vec takes a waveform as input, but waveforms only measure amplitude over time??
Oct 6	☑ ~~GIVE DR PROPOSAL~~ ☑ ~~Work on dr garcia feedback~~ Email: Hi Brooklyn and Dr Anthanasopoulou, Thanks for sending me the paper, Brooklyn! It was very helpful and had some info I cited in my proposal. I read the methodology of her unigram and bigram model, but it says exactly the same thing as the conference paper so it's not much clearer. I was wondering if either of you could take a look at the introduction and objective sections of my research proposal to see if I missed anything. research proposal hehe

	See you on Thursday! Andi Feedback Period goes after
Oct 7	☑ ~~Read paper on pitch accent classification with legrendre coef~~ ☑ ~~Finish methodology of proposal~~ ☑ ~~Finish introduction of proposal~~ ☑ ~~Update kale~~
Oct 8	☑ ~~Print out leggrede paper figues -!!~~
Oct 9 💜	☑ ~~Write my version of the methodology (can be changed later)~~ <u>Next meeting:</u> • Need to start talking about methodology • How do linguists determine pitch accent? • What features to measure? • Ask about how to interpret fig 2.1 and 2.2 on p 21 of lengerede paper • Wav2vec takes a waveform as input, but waveforms only measure amplitude over time?? • Grabe paper with british english To get at library: Improving predictive accuracy of logistic regression model using ranked set samples: Communications in Statistics - Simulation and Computation: Vol 46 , No 1 - Get Access Todo: ☑ ~~Label phrases as Q's~~ ☑ ~~10 or 20 as max word count~~ ☑ ~~Fix the transcripts~~ ☐ Pykit learn for legderend coef Python package to label function and content words Measure coef in content words only Find a way to summarise to sentence level to features… Time chunking? (I don't get) • Split into chunks bc some parts are more important than others Measure intensity? • Find super stressed words w intensity Write code to get

	Valence = positive or neg Arousal = extent of emotion Histogram of: ● Speech rate ○ Of each word: package to count syl ■ CMU dictionary ○ Syllables per second ● Just include baseline features ● Change in speaking rate over time ● Speaking rate compared to average speaking rate of person ● Word level leg coef: save for later… ● Arousal and valence and emotion distribution ● How did they classify emotions?? ● Intensity “Neutral networks” Read MUStARD++ paper
Oct 10	☑ ~~Finish methodology~~ ☑ ~~Send update email to a and brooklyn~~ ☑ ~~Dataset:~~ ☑ ~~Cut out utterances that are too long~~ ☑ ~~Lable q and statements~~ ☑ ~~Make histograms of valence, arousal, emotion~~ ☑ ~~Convert mp4 to wav files~~ ☑ ~~Make list of baseline features and start to apply them~~ Hi both, Here are the notes I took from our meeting yesterday: ● Talked about features to use ○ Need to find a way to convert word level features to sentence level features ○ Legendre coef of content words ○ Scipy ○ Splitting utterances into chunks bc some parts of it are more important than others… ● andi heres ur todo: ○ **Rough draft of oral presentation** ○ **Finish research proposal** ○ Histogram of valence, arousal, and emotion distribution ○ Get baseline features found in other papers ■ CMU dictionary ○ Change in speaking rate over time

	○ Label questions vs statements ○ Max 10 or 20 word count for utterances ● Next meetings Oct 16 online and Oct 23 in person Thanks Andi
Oct 11	☑ ~~Finish applying all baseline features~~
Oct 14-15	☑ ~~Apply feedback of research proposal~~ ☑ ~~Make first draft of oral presentation slides~~ ☑ ~~Write bullet points for what I'm gonna talk about~~ ☑ ~~Send research proposal to G and K~~ (1) Paragraph 4 "The inability of individuals to understand sarcasm can lead to miscommunication...." --> I would also include some discussion about online communication, which links to your project's goal. (2) Paragraph 6 "From Table 1, all research focused on sarcasm detection has been with neutral networks...." --> You need to explain the significance of the black-box point a bit more. You say that NN's "decision-making process is hard to interpret when making predictions" and "difficult for users to understand why a model makes a prediction or to find biases and errors." But why is this a bad thing? I don't understand everything about how my computer works, yet that doesn't cause any problems in my work. You need to include here that part of the goal is to understand how language and communication works. So, using these black-boxes isn't helpful. (3) Also, very strong statements like "No current research has ever applied a simple logistic regression model to the MUStARD or MUStARD++ dataset for sarcasm detection" are never a good idea. There might be a paper you haven't read yet. And the significance isn't about applying the models to the specific dataset. In fact, there is a paper that does exactly this: the Rakov paper you cite later in the proposal. Similarly, phrases that talk about proofs, proving, etc. aren't correct. We show evidence for, don't prove.
Oct 16 💜	☑ ~~Submit proposal !!!~~ ☑ ~~Need to define words for unfamiliar audience?~~ ☑ ~~Ask about significance~~

	☑ ~~Ask: how many presentations I need to attend?~~ ☑ ~~4~~
Oct 18	☑ ~~Email about script and slides~~ ☑ ~~Also include can't find papers about sarcasm expression through video, only text online~~ ☑ ~~Took notes on MUStARD++ dataset construction~~
Oct 19	Hi Dr Athanasopoulou and Brooklyn, Apologies for sending the presentation script a bit late, my code took longer than expected to work. Here's my script and presentation: https://docs.google.com/document/d/1ZpzGe0fVU_HXpProgZoXFU7t8xKdABWVVcml2PFVAhk/edit?usp=sharing https://docs.google.com/presentation/d/1ahlBYz8Y4bImhnqw_mD18k4oFC_rbAWJmQMlEzOGUYo/edit?usp=sharing I read the paper for MUStARD++, and data for arousal, valence, and emotion were added by annotators. I made some bar graphs showing the valence and arousal distribution. 

	 Both sarcastic and sincere utterances have a aprox normal distribution of valence and arousal. Sincere utterances have a larger range and high mean valence than sincere ones, which I guess makes since because sarcastic utterances are more negative. Sarcastic utterance had a higher mean arousal, maybe because they are more emotionally charged. Valence and arousal would probably be good training features because of their consistent group difference, but I don't know how that would be applied to real life. If we tested the model on a utterance not part of the MUStARD++ dataset, it wouldn't have a number for valence or arousal and there's no way to automatically measure it. I've been applying the feedback about the research proposal, but I can't find any papers on sarcasm in online video/audio, only through texting apps. Should I include it anyway even if its not related to sarcastic speech? Have a good week Andi
Oct 20	☑ ~~Add more bkg on sarcasm prosody to proposal~~ ☑ ~~Add research proposal feedback~~ ☑ ~~Ask about citing a thesis~~
Oct 21-22	☑ ~~Submit paper~~ ☑ ~~Extend speech by 1 min~~ ☑ ~~Add more images to slides~~

Oct 23 💜	☑ ~~Practice speech at least 5 times until smooth~~ <u>Notes</u> Paselmouth Valence/arousal - prob not add either - Not gonna use these as training features Prosodic pronunciation Talk less about prosodic features Dont talk about positive negative Add Praat picture 7 - add photo of mustard content Sitcoms Process for how audio gets classified How to find an accent on a word? Try to understand as much as I can about the dataset
Oct 24-25	☑ ~~Send recording to g~~ ☑ ~~Do images need to be cited?~~ ☑ ~~Add overview diagram to whole methodology~~ ☑ ~~Okay to talk a lot about significance of project ?~~ ☐ Email: decided to remove mustast image
Oct 27	☑ ~~Apply parseltongue to code instead of Librosa~~ ☑ ~~Send update email~~ Hi Dr Athanasopoulou and Brooklyn, I measured the mean and SD of F0 with Parselmouth again, and they're all positive numbers so I assume that's more reasonable. I also made some modifications to the presentation slides, but it's mostly adding images on every page, changing the colour, and making the text bigger. I'm trying to figure out how to measure syllables per second. I've counted the syllable number with cmudict, but I can't figure out how to trim each audio file to the exact start and end of the sentence. Is this a MFA feature? Andi
Oct 28	☑ ~~Attend 3 oral presentation today~~ ☑ ~~Present to dr garcia at 12 pm today~~ ☐ Practice speech @ home

	☑ ~~Learn about what is precision, recall, F1~~ ☑ ~~Make plan for NOV~~ <u>Notes on presentations:</u> Richard- making copper alloy for nuclear reactors - Combining copper and molybdenum to better transfer heat in nuclear reactors - Nucreal reactors good! - Production stagnating - Bad pubic opinion, high cost, low operational lifespan - Material degradation - wearing down materials - Parts not good at conducting heat What kind of materials used? Can alloy be applied to other fields other than nuclear energy? What is molybdenum? How will making the alloy improve public opinion, high cost, degredation? How long do current reactors operate for? Emma - Child abuse and domestic violence - Both very important health issue - Abuse gets worse with DV - - Domestic violence more commonly happens to women than men? Applications of results? Analysis strategies? Studying different types of abuse individually? Use database? Sample size? What type of abuse most common? Specific type more common w the presence of DV? Jessica - MS - MS disease, destroys myelin sheafs - Bad symptoms :(Have other studies been done on this before? Sample size? <u>TIPS</u> Strict timer at 10 minutes Add more direct sig of project
Oct 30 💜	ORAL PRESENTATION TODAY ON CAMPUS MEETING Sounding less read-y: be more excited ☑ ~~Attend 1 presentation today~~ Notes: Justin - turning off genes - Csp fluid buildup - PRPF4B gene - Big impact on brain

	- Remove the gene PRPF4B PRPF4B gene causes CSF fluid? What is function of gene? Brooklyn out from dec 12 to jan 3 A out start of jan until end of may <u>Meeting notes</u> By next meeting: ● Fin cleaning data w Praat ● Upload data into MFA ● Start writing intro ● Ask for rubric Clean data… Accent detection… Wavelet Extension of rakov Phonation Extrapolate pitch
Nov 31	☑ ~~Email summary~~ ☑ ~~Divide data for cleaning~~ 182/day ☑ ~~Ask for rubric~~ Email Hi everyone, Here is what I will be doing for the next two weeks. Let me know if I missed anything: ● Clean data with Praat ● Create text align files in MFA ● Writing intro to paper Andi
November 🍂	☑ ~~Prepare data for training~~

<table>
<tr>
<td>Nov 1</td>
<td>

☑ ~~Email: saving as wav file cuts off at end

☑ ~~Make screen recording to go along with it…~~

☑ ~~Start writing methodology~~

Email:

Hi everyone,

I starting trimming the clips in Praat, and I notice the following:

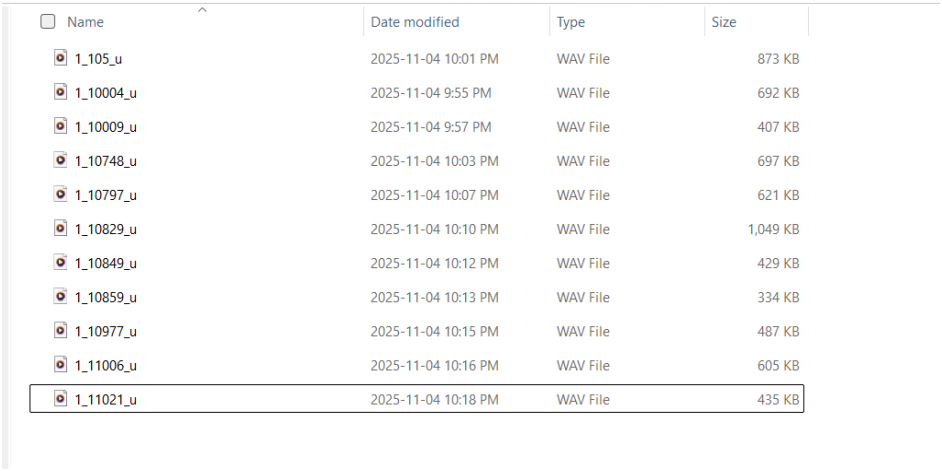
One of the characters has an Indian accent. Should I remove all utterances by him?
Some utterances have laughter in between the speaking. Should I cut the middle laughter out as well?
Not a question but nasals, ʃ, and the end of fricatives are harder to see on the spectrogram. /k/ is now my favourite sound
When I view a clip in Praat, it looks fine but when I save it as a wav file and play it, the end is cut off. I attached a video of what I mean. Any way to fix this?

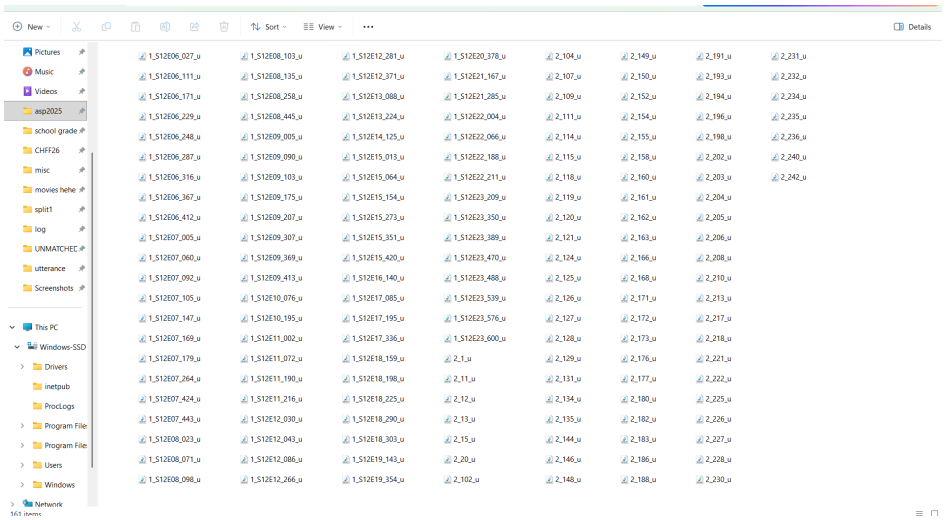
Thanks

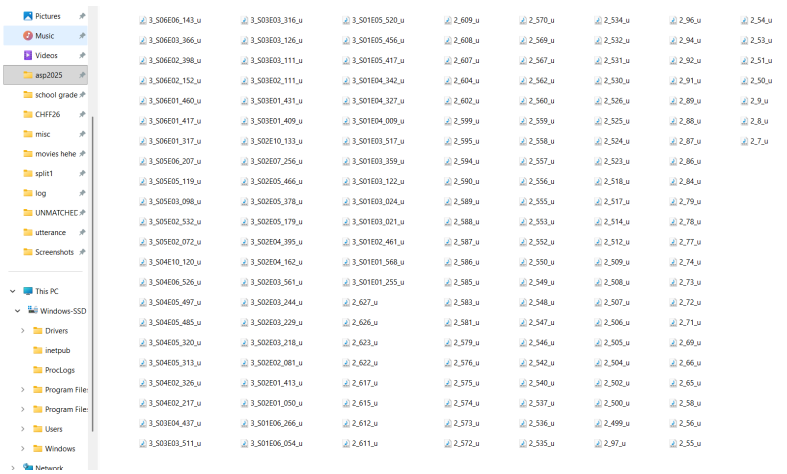
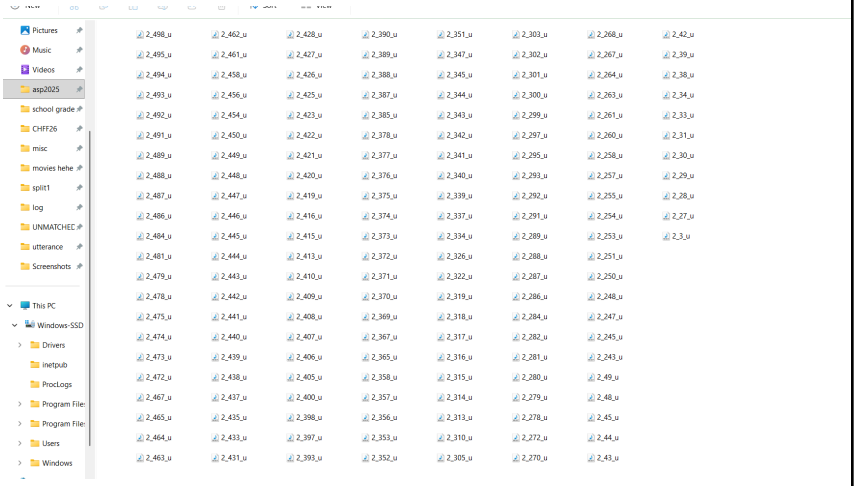
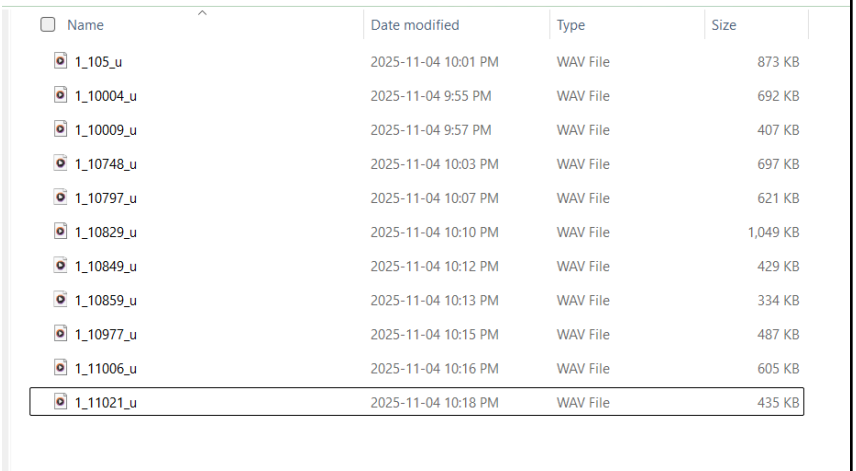
Andi
</td>
</tr>
<tr>
<td>Nov 3-4</td>
<td>

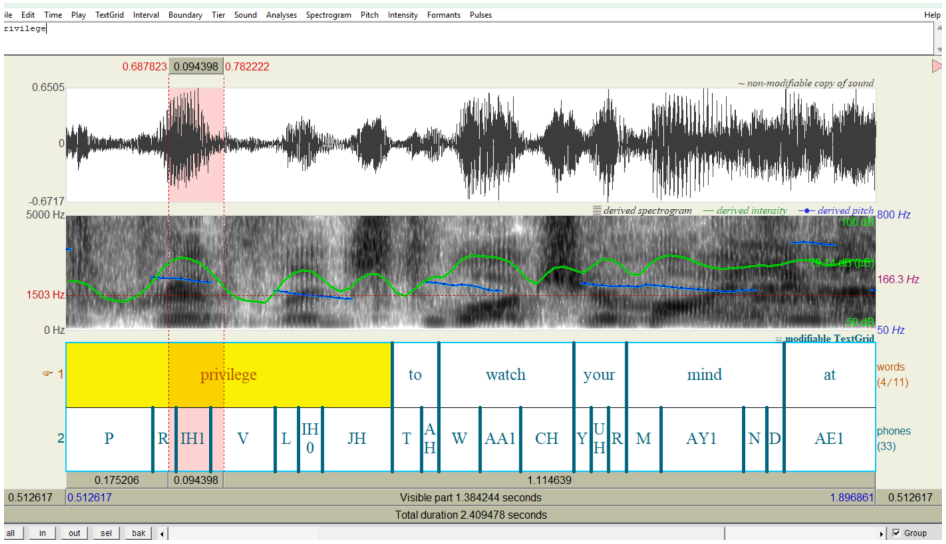
☑ ~~Remove all nonAmerican speakers and write email~~
☑ ~~Go back over set 1 and leave more room/ cut out centre laughter~~
☑ ~~Rerecord video/send dr some of the challenging clips~~

Do these people have american accents?
<table>
<tr><td>Raj</td><td>x</td></tr>
<tr><td>Dinesh</td><td>x</td></tr>
<tr><td>Person1</td><td>Depends</td></tr>
<tr><td>Person3</td><td>Depends</td></tr>
<tr><td>Person</td><td>Depends</td></tr>
<tr><td>Other</td><td>Depends</td></tr>
<tr><td>Moderator</td><td>x</td></tr>
<tr><td>Member-girl</td><td>x</td></tr>
<tr><td>Member-boy</td><td>x</td></tr>
</table>
Email
</td>
</tr>
</table>

	Some character labels apply to multiple characters, so I need to go through those individually to hear the accent. For the people I can definitely remove, we would lose at least 62 utterances (35 sarcastic, 25 sincere). Leaving space after the [k] worked! How much audio should I leave at the end? Is it fine as long as I cut most of the end laughter? I've attached the original and edited clip below. Clips cut:  Name / Date modified / Type / Size 1_105_u — 2025-11-04 10:01 PM — WAV File — 873 KB 1_10004_u — 2025-11-04 9:55 PM — WAV File — 692 KB 1_10009_u — 2025-11-04 9:57 PM — WAV File — 407 KB 1_10748_u — 2025-11-04 10:03 PM — WAV File — 697 KB 1_10797_u — 2025-11-04 10:07 PM — WAV File — 621 KB 1_10829_u — 2025-11-04 10:10 PM — WAV File — 1,049 KB 1_10849_u — 2025-11-04 10:12 PM — WAV File — 429 KB 1_10859_u — 2025-11-04 10:13 PM — WAV File — 334 KB 1_10977_u — 2025-11-04 10:15 PM — WAV File — 487 KB 1_11006_u — 2025-11-04 10:16 PM — WAV File — 605 KB 1_11021_u — 2025-11-04 10:18 PM — WAV File — 435 KB
Nov 5	☑ ~~Email: not due in nov, due sometime september, and going to be pretty similar to the proposal~~ ☑ ~~Also, ask mentor is Dinesh's accent is acceptable~~ ☑ ~~Finish cutting set 1 and 2~~ ☑ ~~Add character exclusion to methodology~~ • Weekly email → progress report, summary of meeting, plan for next week Methodology done: 1. Preprocessing a. Convert all mp4 to wav b. Remove utterances from the following characters (35 sarcastic, 27 sincere) i. Raj ii. Moderator iii. Member-Girl iv. Member-Boy c. Trim audio - open wav files in Praat and cut out laughter by looking at spectogram, save as new wav file Files clipped

	Email: Hi Dr Anthanasopoulou and Brooklyn, The Intro deadline was actually pushed back to mid December because Dr Garcia wanted to give us all of November to focus on the project :) There is another character that I think doesn't have an American accent but I just wanted to confirm before I remove him. Examples of his utterances are files 3_S01E03_056.wav and 3_S01E03_249.wav I cut the 11120.wav file like the images, but I can't hear [k] when I play it in Praat or when I open it in Window Media Player. Interestingly, when I open the wav file in Microsoft Clipchamp, I can hear [k] so maybe it's a problem with the audio player itself? Is the sound cut off in the huh.wav file on your end? Andi
Nov 6	☑ ~~Cut set 3~~ 
Nov 7	• Talk about long term plan w Dr A and B • Next due date mid Dec for intro
Nov 9	☑ ~~Cut set 4~~

	
Nov 10	☑ ~~Cut set 5~~ 
Nov 11	☑ ~~Cut set 6~~  ☐
Nov 12	☑ ~~Follow tutorial for aligning textgrid file in MFA~~

	☑ ~~Start with a single recording~~ 
Nov 13 💜	☑ ~~Load all clips into MFA and get textgrid file for all of them~~ ☐ Add textgrid prep to methodology Email: Hi Dr Anthanasopoulou and Brooklyn, I've clipped all the utterances in Praat, I've removed about 70 because the speakers didn't have an American accent and 40 for having really loud music/ringing/beeping in the background. In total, I kept 981 of the 1092 utterances. Something I noticed is for a lot of the utterances, the transcript doesn't match the audio. The audio cuts out words at the start or end, but I suppose I can fix this in the TextGrid. MFA took me a while to figure out, but I've generated the TextGrid for 953 of the 981 files. I took a look at the other 28 audio clips and they have words that aren't in the MFA dictionary (ie Leonard, crayon). I read that I can add my own words and pronunciation into the dictionary, but I don't know how. Also, for the meeting today, are you available to have it from 4:20-5:20? I have a different meeting after this one. Andi <u>Notes from meeting</u> ● Do MFA storted by speaker ● Handle unknown words ○ Fix typos

<table>
<tr><td></td><td>

Send alt pronunciation to a and b

Convert txt to lab
Make table on removed utterances and reasons
Might not meet next week

</td></tr>
<tr><td>Nov 14-16</td><td>

☑ ~~Do MFA storted by speaker~~
☑ ~~Handle unknown words~~

☑ ~~Fix typos~~

☑ ~~Make table on removed utterances and reasons~~

File of OOV: oov_pronounciation_1

Email:

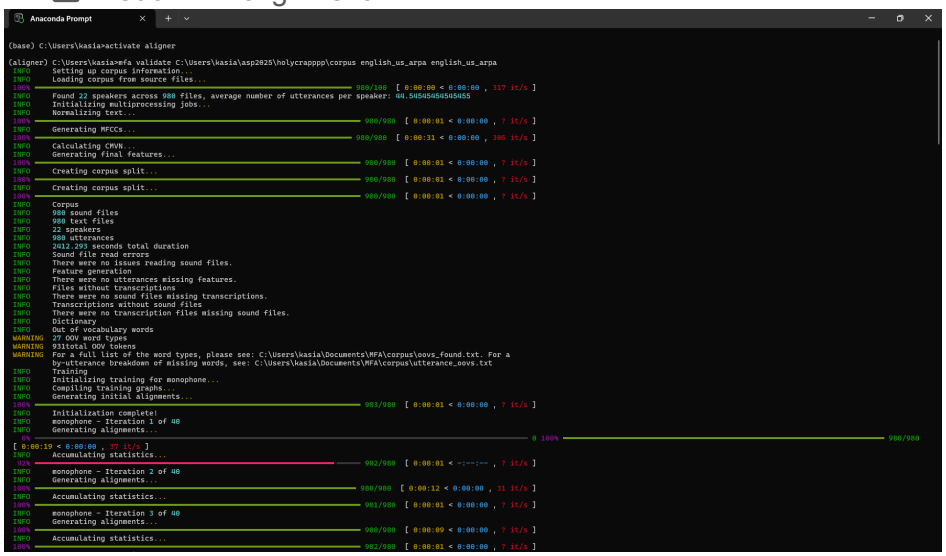
Hi Dr Anthanasopoulou and Brooklyn,

I fixed the typos in the transcript. For the other words, should I highlight what I think the ARPABET is for each word in the text file and send it to you to double check?

Andi

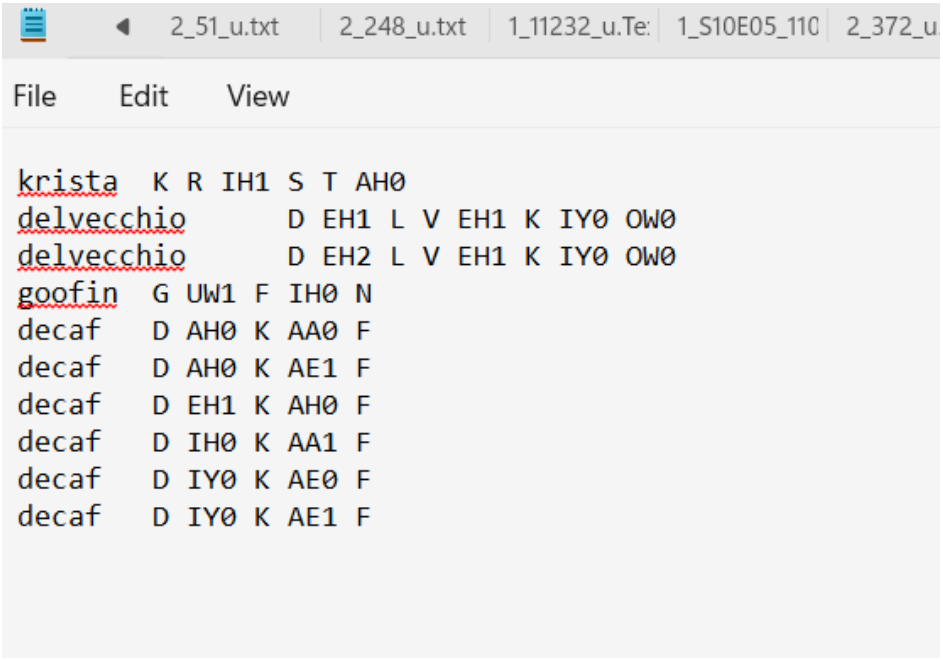
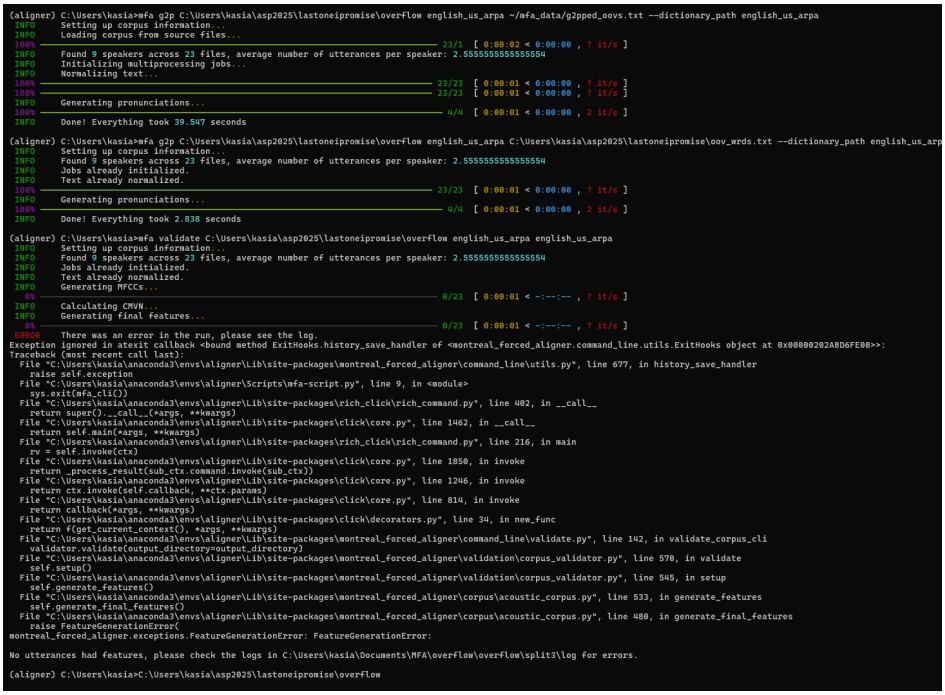
<table>
<tr><td>Reason</td><td>Sarcastic</td><td>Sincere</td></tr>
<tr><td>Not an American accent</td><td>42</td><td>32</td></tr>
<tr><td>Loud</td><td>1</td><td>1</td></tr>
<tr><td>Background music/sound</td><td>21</td><td>9</td></tr>
<tr><td>Too much cut off/too short</td><td>1</td><td>3</td></tr>
</table>
</td></tr>
<tr><td>Nov 17</td><td>

☑ ~~Redo MFA alignment~~



☑ ~~Print out words not found in MFA dictionary~~

</td></tr>
</table>

	File of OOV:  Notes • Planning/todo and evidence of work in logbook • MFA aligned 957 of the 980 files, so **23** files are unaligned ○ In the “overflow” folder
Nov 21	☑ ~~Run MFA on overflowed files~~ Error  ☑ ~~Send pronunciations to A and B~~

	☑ ~~Brainstorm features~~ > To detect elongation → maybe measure how much slower (in sly/s) the slowest word is pronounced in the sentence compared to the mean/median syl/s of the whole sentence > > Find legendre coef of each content word → could consider clustering like ravok Email: Hi Dr Anthanasopoulou and Brooklyn, I reran MFA on the whole corpus and had 23 utterances that weren't aligned. I printed out 4 OOV, but obviously most of the unaligned utterances had perfectly fine transcripts so I'm not sure why MFA didn't work. I've attached the .txt file with the OOV. Once I add the OOV to the dictionary and run MFA on utterances again, perhaps I should just manually make the Textgrid files for the unaligned files. I can't think of anything else to do. Also, I've created a document to put ideas about what the word level features should be. I hope the conference is going well Andi
Nov 23	☑ ~~Select OOV pronunciations and add to dict~~

```
krista  K R IH1 S T AH0
delvecchio      D EH2 L V EH1 K IY0 OW0
goofin  G UW1 F IH0 N
decaf   D IY0 K AE1 F
```

- [x] ~~Align the overflow files again~~

```
Anaconda Prompt

(aligner) C:\Users\kasia>mfa align C:\Users\kasia\asp2025\lastoneipromise\overflow english_us_arpa english_us_arpa C:\Us
ers\kasia\asp2025\lastoneipromise\overflow_output --cleaned
 INFO     Setting up corpus information...
 INFO     Loading corpus from source files...
  23% ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23/100  [ 0:00:02 < -:--:-- , ? it/s ]
 INFO     Found 9 speakers across 23 files, average number of utterances per speaker: 2.5555555555555554
 INFO     Initializing multiprocessing jobs...
 INFO     Normalizing text...
 100% ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23/23  [ 0:00:01 < 0:00:00 , ? it/s ]
 INFO     Generating MFCCs...
 100% ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23/23  [ 0:00:02 < 0:00:00 , ? it/s ]
 INFO     Calculating CMVN...
 INFO     Generating final features...
 100% ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23/23  [ 0:00:01 < 0:00:00 , ? it/s ]
 INFO     Creating corpus split...
 100% ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23/23  [ 0:00:01 < 0:00:00 , ? it/s ]
 INFO     Compiling training graphs...
 INFO     Performing first-pass alignment...
 INFO     Generating alignments...
 100% ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23/23  [ 0:00:01 < 0:00:00 , ? it/s ]
 INFO     Collecting phone and word alignments from alignment lattices...
  35% ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8/23  [ 0:00:05 < -:--:-- , ? it/s ]
 INFO     Analyzing alignment quality...
  35% ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8/23  [ 0:00:01 < -:--:-- , ? it/s ]
 INFO     Exporting alignment TextGrids to C:\Users\kasia\asp2025\lastoneipromise\overflow_output...
  35% ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8/23  [ 0:00:03 < 0:00:03 , 5 it/s ]
 INFO     Finished exporting TextGrids to C:\Users\kasia\asp2025\lastoneipromise\overflow_output!
 INFO     Done! Everything took 68.566 seconds
```

```
  File "C:\Users\kasia\anaconda3\envs\aligner\Lib\site-packages\montreal_forced_aligner\alignment\pretrained.py", line 4
21, in align
    super().align()
  File "C:\Users\kasia\anaconda3\envs\aligner\Lib\site-packages\montreal_forced_aligner\alignment\base.py", line 443, in
 align
    self._align()
  File "C:\Users\kasia\anaconda3\envs\aligner\Lib\site-packages\montreal_forced_aligner\alignment\pretrained.py", line 4
14, in _align
    super()._align()
  File "C:\Users\kasia\anaconda3\envs\aligner\Lib\site-packages\montreal_forced_aligner\alignment\base.py", line 412, in
_align
    self.align_utterances()
  File "C:\Users\kasia\anaconda3\envs\aligner\Lib\site-packages\montreal_forced_aligner\alignment\mixins.py", line 312,
in align_utterances
    raise NoAlignmentsError(self.num_current_utterances, self.beam, self.retry_beam)
montreal_forced_aligner.exceptions.NoAlignmentsError: NoAlignmentsError:

There were no successful alignments for 15 utterances.
The current set up used a beam of 10 and a retry beam of 40.
You can try rerunning with a larger beam (i.e. "mfa align ... --beam 100 --retry_beam 400").
If increasing the beam size does not help, then there are likely issues with the corpus, dictionary, or acoustic model,
and can be further diagnosed with the "mfa validate" command
```

```
(aligner) C:\Users\kasia>mfa align C:\Users\kasia\asp2025\lastoneipromise\thesejusthatemeidk english_us_arpa english_us_
arpa C:\Users\kasia\asp2025\lastoneipromise\ahhhhhhhh --cleaned --beam 100
 INFO     Setting up corpus information...
 INFO     Found 7 speakers across 15 files, average number of utterances per speaker: 2.142857142857143
 INFO     Jobs already initialized.
 INFO     Text already normalized.
 INFO     Features already generated.
 INFO     Compiling training graphs...
 INFO     Performing first-pass alignment...
 INFO     Generating alignments...
 100% ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15/15  [ 0:00:01 < 0:00:00 , ? it/s ]
 INFO     Collecting phone and word alignments from alignment lattices...
 100% ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15/15  [ 0:00:05 < 0:00:00 , ? it/s ]
 INFO     Analyzing alignment quality...
 100% ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15/15  [ 0:00:02 < 0:00:00 , ? it/s ]
 INFO     Exporting alignment TextGrids to C:\Users\kasia\asp2025\lastoneipromise\ahhhhhhhh...
 100% ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15/15  [ 0:00:04 < 0:00:00 , 7 it/s ]
 INFO     Finished exporting TextGrids to C:\Users\kasia\asp2025\lastoneipromise\ahhhhhhhh!
 INFO     Done! Everything took 22.200 seconds
```

```
]: #count num files in folder
file_count = sum(len(files) for _, _, files in os.walk(r'C:\Users\kasia\asp2025\lastoneipromise\full_output'))
print(file_count)
```

```
2943
```

- [x] ~~Finish generating textgrids for all files~~

FIND ALL TEXTGRIDS HERE: textgrids

- [x] ~~Write code to find F0 mean, SD, range for women and men~~

```
plus['pitchmean'] = None
plus['pitchrange'] = None
plus['pitchsd'] = None

def measurePitch(sound, f0min, f0max, unit):
    duration = call(sound, "Get total duration")
    pitch = call(sound, "To Pitch", 0.0, f0min, f0max)
    meanF0 = call(pitch, "Get mean", 0, 0, unit)
    stdevF0 = call(pitch, "Get standard deviation", 0, 0, unit)
    minF0 = call(pitch, "Get minimum", 0, 0, unit, "Parabolic") #check this
    maxF0 = call(pitch, "Get maximum", 0, 0, unit, "Parabolic")
    f0range = maxF0 - minF0

    return duration, meanF0, stdevF0, f0range

base_dir = r'C:\Users\kasia\asp2025\textgrids'
#MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN
for i, key in enumerate(plus['SCENE']):
    try:
        wav_path = None
        for root, dirs, files in os.walk(base_dir):
            target_file = f"{key}.wav"
            if target_file in files:
                wav_path = os.path.join(root, target_file)
                break

        if wav_path is None:
            raise FileNotFoundError(f"{key}.wav not found in subfolders")

        sound = parselmouth.Sound(wav_path)
        (duration, meanF0, stdevF0, f0range)=measurePitch(sound, 98.4, 591, "Hertz")
        #make diff one for women

        plus.at[i, 'pitchmean'] = meanF0
        plus.at[i, 'pitchsd'] = stdevF0
        plus.at[i, 'pitchrange'] = f0range
    except Exception as e:
        print(f"oh noses u have a problem in {wav_path}: {e}")
```

- [x] ~~Find research on min and max pitch for men and women to determine cutoff~~

[Mean lowest and highest frequencies and phonational range for men and... | Download Scientific Diagram]

	Men (N = 18)		*Women (N = 14)*	
	M	*SD*	*M*	*SD*
Lowest frequency (Hz)	98.4	11.6	170.0	20.1
Highest frequency (Hz)	591.1	131.1	823.0	194.4
Phonational range (in semitones)	30.9	3.8	36.7	4.0

Email:

Hi Brooklyn and Dr Anthanasopoulou

I managed to generate Textgrids for all utterances. For the 4 OOV words, I looked at an IPA to Arpabet table and chose the best pronunciation (I only chose the pronunciation for two words).

A few weeks ago I showed you the script for getting F0 mean, SD, and range. I'm setting the min and max F0 values to the mean low and high pitch range for men and women from this table.

The mean pitch I'm getting for men is about 150-200 Hz and the range is from like 150-400 Hz. The pitch range matches the table, but is this a reasonable value?

Andi

Nov 23

```
plus["GENDER"] = plus["SPEAKER"].isin(['SHELDON', 'HOWARD',
                                       'LEONARD', 'STUART', 'CHANDLER', 'JOEY', 'ROSS','GILFOYLE', 'RICHARD', 'ERLICH']).map({True: "M", False: "F"})
plus["GENDER"] = plus["SCENE"].isin(['1_4639_u', '1_4789_u','1_7494_u', '1_7504_u', '1_7953_u', '1_8136_u', '1_9963_u', '1_11051_u','1_11885_u'
                                    ,'2_55_u','2_150_u','2_193_u','2_228_u','2_245_u','2_278_u','2_295_u','2_341_u','2_415_u','2_530_u','2_548_u'
                                    ,'2_609_u','2_627_u','1_6645_u','2_42_u','1_S10E01_225_u','1_S11E08_351_u','1_S12E02_274_u','1_S12E04_204_u'
                                    ,'1_S12E04_233_u','1_S12E11_072_u','1_S12E20_378_u','3_S01E02_461_u','3_S01E04_327_u']).map({True: "M", False: "F"
plus = plus.reset_index()
```

1023

[23]:
```
root_folder = r"C:\Users\kasia\asp2025\textgrids"
wav_files = []
for dirpath, _, filenames in os.walk(root_folder):
    for f in filenames:
        if f.lower().endswith(".wav"):
            wav_files.append(f)

wav_set = set(wav_files)
wav_set = {os.path.splitext(f)[0] for f in wav_set}
plus = plus[plus["SCENE"].str.strip().isin(wav_set)]
```

☑ ~~Add gender labels to data~~

[212]:
```
#WEOMENWEOMENWEOMENWEOMENWEOMENWEOMENWEOMENWEOMENWEOMENWEOMENWEOMENWEOMENWEOMENWEOMENWEOMENWEOMENWEOMENWEOMENWEOMENWEOMEN
for idx, key in wmen['SCENE'].items():
    try:
        wav_path = None
        for root, dirs, files in os.walk(base_dir):
            target_file = f"{key}.wav"
            if target_file in files:
                wav_path = os.path.join(root, target_file)
                break

        if wav_path is None:
            raise FileNotFoundError(f"{key}.wav not found in subfolders")

        sound = parselmouth.Sound(wav_path)
        duration, meanF0, stdevF0, f0range = measurePitch(sound, 170, 823, "Hertz")

        plus.loc[idx, ['pitchmean', 'pitchsd', 'pitchrange', 'duration']] = [
            meanF0, stdevF0, f0range, duration
        ]

    except Exception as e:
        print(f"oh noses u have a problem in {wav_path}: {e}")
```

[*]:
```
#MEENMEENMEENMEENMEENMEENMEENMEENMEENMEENMEENMEENMEENMEENMEENMEENMEENMEEN

for idx, key in men['SCENE'].items():
    try:
        wav_path = None
        for root, dirs, files in os.walk(base_dir):
            target_file = f"{key}.wav"
            if target_file in files:
                wav_path = os.path.join(root, target_file)
                break

        if wav_path is None:
            raise FileNotFoundError(f"{key}.wav not found in subfolders")

        sound = parselmouth.Sound(wav_path)
        duration, meanF0, stdevF0, f0range = measurePitch(sound, 98.4, 591, "Hertz")

        plus.loc[idx, ['pitchmean', 'pitchsd', 'pitchrange', 'duration']] = [
            meanF0, stdevF0, f0range, duration
        ]

    except Exception as e:
        print(f"oh noses u have a problem in {wav_path}: {e}")
```

Nov 26

☑ ~~Write gender distribution in methodology~~

1. Use different min and max F0 based on speaker gender. (.. for males and … for females base off research)

Male		Female	
Sarcastic	Sincere	Sarcastic	Sincere
11	22	470	477

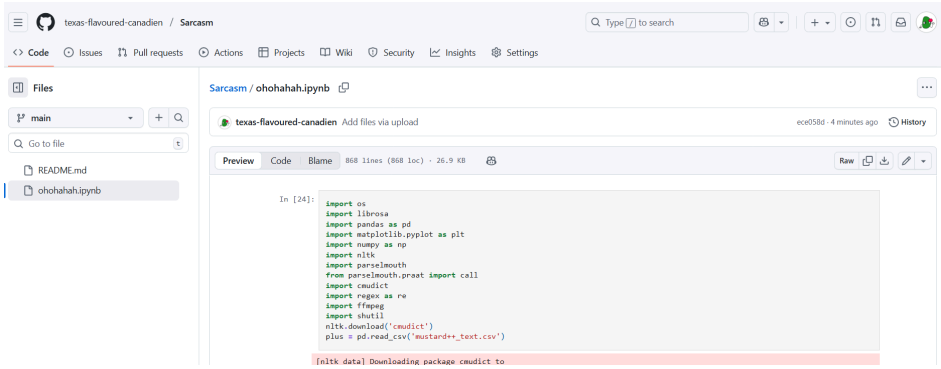
```
6]: haha = (plus['GENDER'].str.contains('F')) & (plus['Sarcasm'] == 0)
    haha.sum()

6]: np.int64(228)
```

☑ ~~Debug gender labelling~~

```
meen = plus["SPEAKER"].isin([
    'SHELDON','HOWARD','LEONARD','STUART',
    'CHANDLER','JOEY','ROSS','GILFOYLE',
    'RICHARD','ERLICH'
])
print(meen.value_counts())
plus["SPEAKER"] = plus["SPEAKER"].str.strip().str.upper()
plus["GENDER"] = meen.map({True: "M", False: "F"})


plus = plus.reset_index()
```

```
SPEAKER
True     591
False    432
Name: count, dtype: int64
```

☑ ~~Backup Jupyte notebook on github~~



☑ ~~Write MFA alignment in methodology~~

c. Trim audio - open wav files in Praat and cut out laughter by looking at spectogram, save as new wav file
d. Create transcript → Create a .txt file of the same name for each utterance with the transcript in all caps
e. Install and run MFA g2p to find OOV words. Print out their predicted pronounciation and add the words to the dictionary
f. Separate the audio files and the text files into folders base on speaker
g. Run MFA validate and align to generate .TextGrid files for the utterances

2. Baseline features
   a. Pitch
      i. Label all utterances by speaker gender. For ambiguous speaker categories (ie "Person1"), listen to each one and manually add gender
      ii. Run Paselmouth code to find F0 mean, SD and range
         1. Use different min and max F0 based on speaker gender. (.. for males and … for females base off research)

Nov 27 💜

☑ ~~Brainstorm global features for leg coef~~

To detect elongation → maybe measure how much slower (in sly/s) the slowest word is pronounced in the sentence compared to the mean/median syl/s of the whole sentence

Stress word → find elongated louder word

Find legendre coef of each content word → could consider clustering like ravok

Trigram or bigram

Or find coef on stressed word

Volume increase → find slope of intensity over utterance

- [x] ~~Update playtime on df~~

```
def measurePitch(sound, f0min, f0max, unit):
    duration = call(sound, "Get total duration")
    pitch = call(sound, "To Pitch", 0.0, f0min, f0max)
    meanF0 = call(pitch, "Get mean", 0, 0, unit)
    stdevF0 = call(pitch, "Get standard deviation", 0, 0, unit)
    minF0 = call(pitch, "Get minimum", 0, 0, unit, "Parabolic") #check this
    maxF0 = call(pitch, "Get maximum", 0, 0, unit, "Parabolic")
    f0range = maxF0 - minF0

    return duration, meanF0, stdevF0, f0range

base_dir = r'C:\Users\kasia\asp2025\textgrids'
#MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN MEN
for i, key in enumerate(men['SCENE']):
    try:
        wav_path = None
        for root, dirs, files in os.walk(base_dir):
            target_file = f"{key}.wav"
            if target_file in files:
                wav_path = os.path.join(root, target_file)
                break

        if wav_path is None:
            raise FileNotFoundError(f"{key}.wav not found in subfolders")

        sound = parselmouth.Sound(wav_path)
        (duration, meanF0, stdevF0, f0range)=measurePitch(sound, 98.4, 591, "Hertz")
        #make diff one for women

        plus.at[i, 'pitchmean'] = meanF0
        plus.at[i, 'pitchsd'] = stdevF0
        plus.at[i, 'pitchrange'] = f0range
        plus.at[i, 'duration']=duration
    except Exception as e:
        print(f"oh noses u have a problem in {wav_path}: {e}")
```

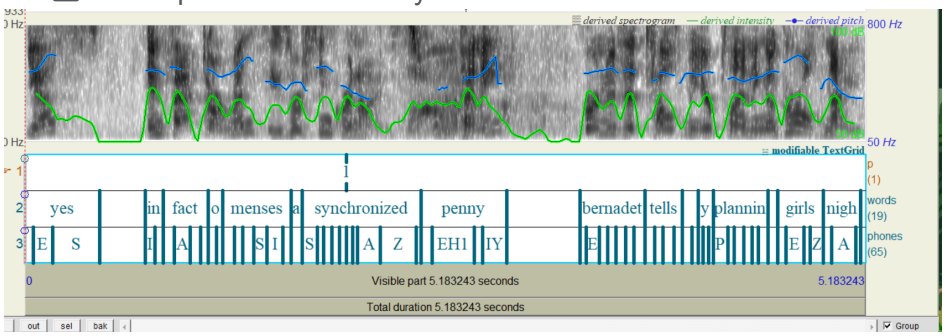
- [x] ~~Read Booklyns github repo~~

	``python
	import parselmouth
	from scipy.special import eval_legendre
	import scipy.stats
	import cmudict
	import numpy as np
	import os
	import math
	import statistics
	import tgt
	
	# Load cmudict. Used to get number of syllables and word rate
	pron_dict = cmudict.dict()
	
	def to_sound(wav_path):
	    """
	    Convert wav file to Praat sound object
	    :param wav_path: full path to wav file
	    :return: wav file as sound object
	    """
	    return parselmouth.Sound(wav_path)
	
	# Get intensity and pitch contours
	def get_contours(sound, start_time, end_time, keep_zeros):
	    """
	    Extract F0 and intensity contours from wav file, within a given time frame
	    :param wav_path: Full path to a single wav file
	    :param keep_zeros: True if you want to keep zeros of F0 contour (i.e., keep spaces without an F0 me
	    :return: the F0 and intensity contours as lists
	    """
	    sound_interval = sound.extract_part(from_time=start_time, to_time=end_time)
	    try: # This deals with if the extracted part is shorter than the window length
	        pitch_contour = sound_interval.to_pitch().selected_array['frequency']
	        intensity_contour = sound_interval.to_intensity().values[0]
	``
	
	Email:
	
	Hi Dr Anthanasopoulou and Brooklyn,
	I wanted to confirm if we were having a meeting today as I don't see it on the Google calendar.
	
	Andi
	
	To talk about in meeting
	- Min word or time cutoff? Some very short
	- Prob
	
	Meeting notes:
	- Wavelet find prominence and cross examine with own annotations
	- Every word exaggerated, or monotone, or one word accented
	-

Nov 29	☑ ~~ERROR IN CODE: code to assign gender overwrites previous code, fix so it doesn't overrun!~~

	8]:
	``
	man = [
	    '1_4639_u','1_4789_u','1_7494_u','1_7504_u',
	    '1_7953_u','1_8136_u','1_9963_u','1_11051_u',
	    '1_11885_u','2_55_u','2_150_u','2_193_u','2_228_u',
	    '2_245_u','2_278_u','2_295_u','2_341_u','2_415_u',
	    '2_530_u','2_548_u','2_609_u','2_627_u','1_6645_u',
	    '2_42_u','1_S10E01_225_u','1_S11E08_351_u','1_S12E02_274_u',
	    '1_S12E04_204_u','1_S12E04_233_u','1_S12E11_072_u','1_S12E20_378_u',
	    '3_S01E02_461_u','3_S01E04_327_u'
	]
	men = [
	    'SHELDON','HOWARD','LEONARD','STUART',
	    'CHANDLER','JOEY','ROSS','GILFOYLE',
	    'RICHARD','ERLICH'
	]
	meeen = plus["SCENE"].isin(man) | plus["SPEAKER"].isin(men)
	print(meeen.value_counts())
	plus["GENDER"] = meeen.map({True: "M", False: "F"})
	``
	``
	True 624
	False 399
	Name: count, dtype: int64
	``
	☑ ~~Update gender distribution NEW STATS!~~
	males and … for females base off research)
	
	| Male
			Female	

---	---	---	---
Sarcastic	Sincere	Sarcastic	Sincere
293	297	188	202	 Email: Hi Dr Anthanasopoulou and Brooklyn, Here's a summary of what I'm doing this week: - Find the prominent word manually and then with Wavelet to compare - Extract legendre coef for the prominent word and all words in the utterance I'm not clear on manually determining prominence. Do I mark it in the Textgrid file? Andi
December ❄️	☑ ~~Finish intro section~~
Dec 1	☑ ~~Mark 3 utterances and email A and B to verify~~ ☐ Use parselmouth to measure syl/sec Using following package: Shahabks/my-voice-analysis: My-Voice Analysis is a Python library for the analysis of voice (simultaneous speech, high entropy) without the need of a transcription. It breaks utterances and detects syllable boundaries, fundamental frequency contours, and formants.

	NOTE: add ms kale and dr garcia as mentors in proposals Class note: ask about schedule over winter break Introduction still due on the 11? YES Email: Hi Dr Anthanasopoulou and Brooklyn, For manually marking prominent words, I decided to just type the word into the csv file for each utterance. I've attached three examples of prominent words to double check. 2_104 → prominent word is sun 2_115 → don't 2_120 → amusing Right now I'm listening for the most emphasized word. Andi
Dec 2	☑ ~~Mark prominence in Amys and chandler's utterances~~  ☐ ~~Install Wavelet and record prominence according to wavelet~~ ~~ASK FOR HELP IN MEETING~~ ~~Installation instructions: wavelet_prosody_toolkit/INSTALL.rst at master · asuni/wavelet_prosody_toolkit~~
Dec 3	Notes: • New intro due date on 15th • Move obj sec into the intro • Record all materials used • Add
Dec 4 💜	☑ ~~Write list of all packages used~~

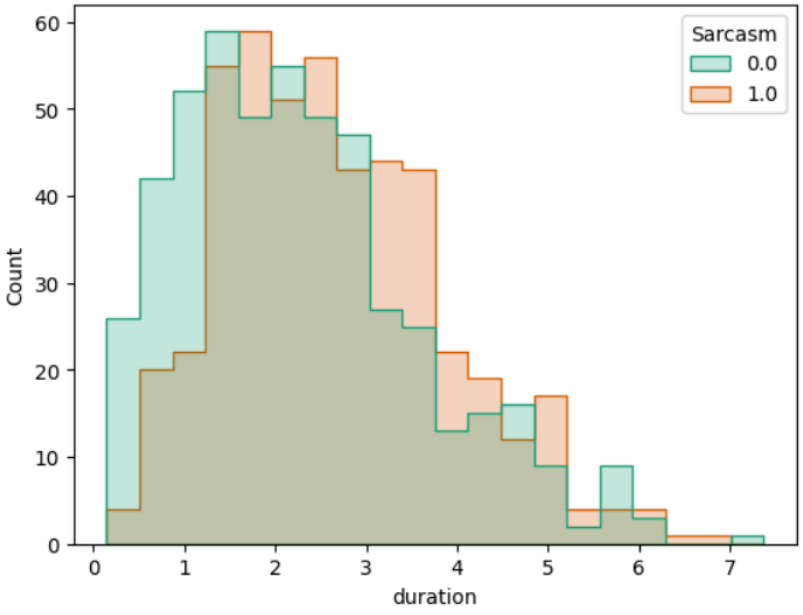
List of all packages:

- Jupyter notebook
- Packages
  - Pandas
  - Parselmouth
  - Madplotlib
  - Scipy
  - my-voice-analysis
- Software
  - Praat
  - Wavelet
  - MFA

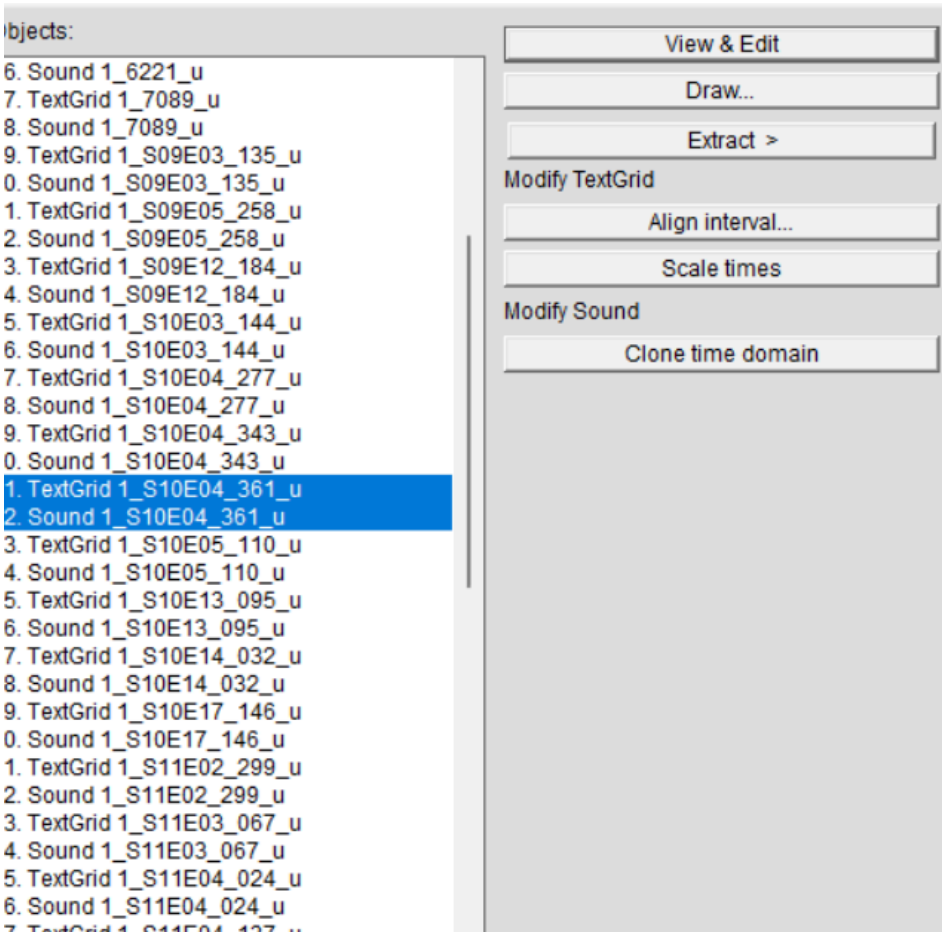
☑ ~~Take notes on Attardo paper~~

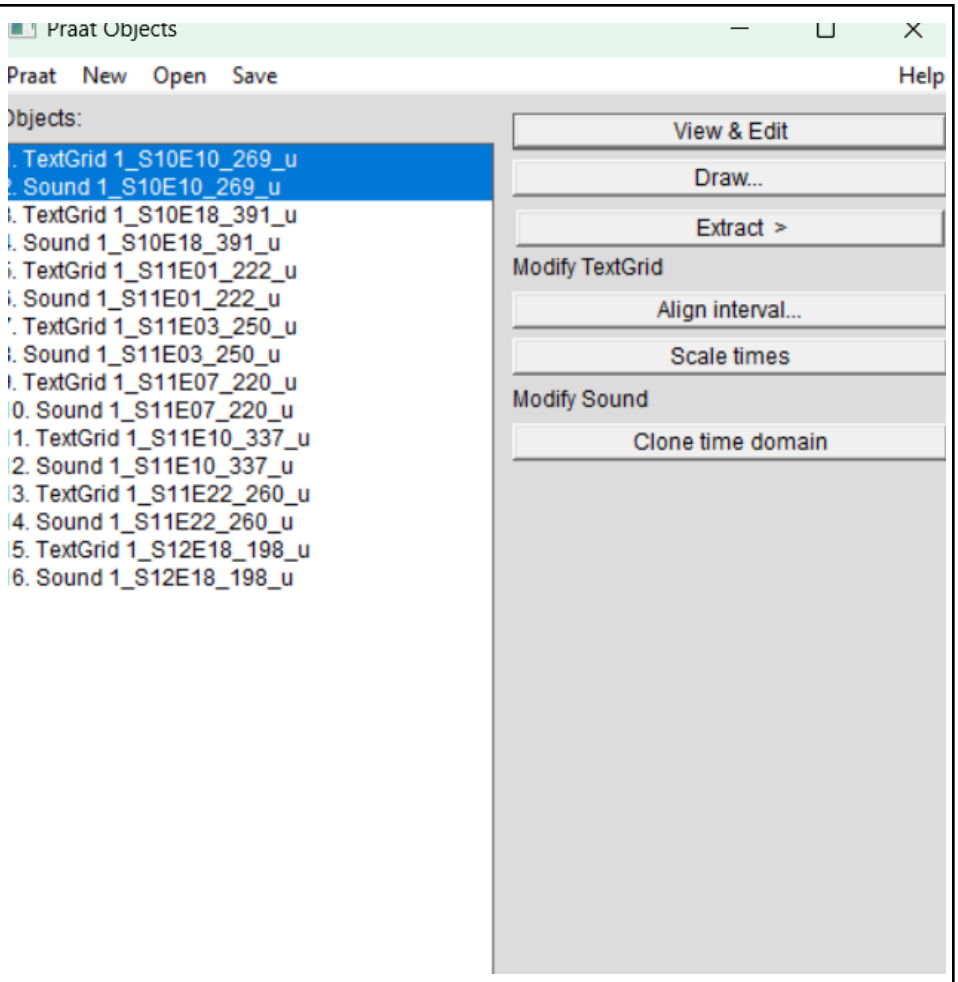
Dec 4	Multimodal_markers_of_irony_and_sarcasm.pdf
	• Pitch marks of sarcasm, three main patterns
	• Within statement contrast
	  ○ Contrast in pitch between initial phrase and next phrase
	  ○ Large pitch range in first part followed by reduced pitch range
	• Low pitch range in whole utterance
	• Pitch accents on all content words and many syllables
	  ○ Elongation and highly rhythmic

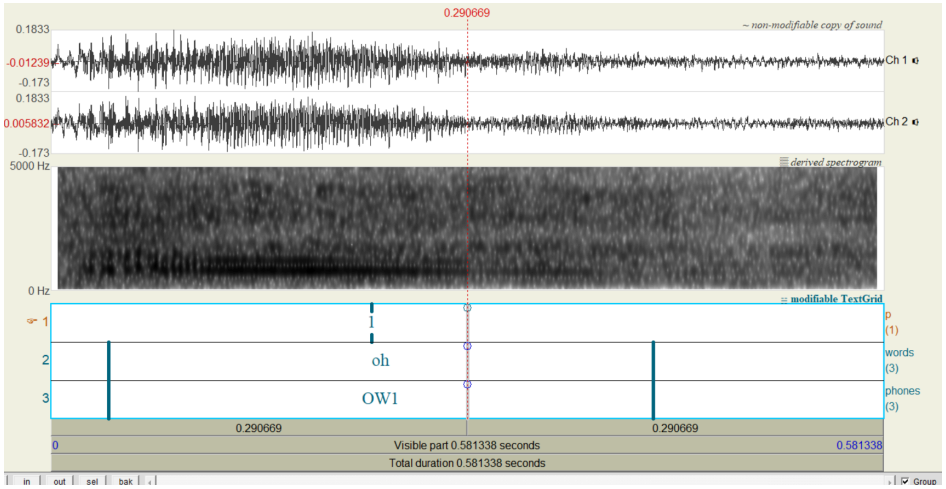
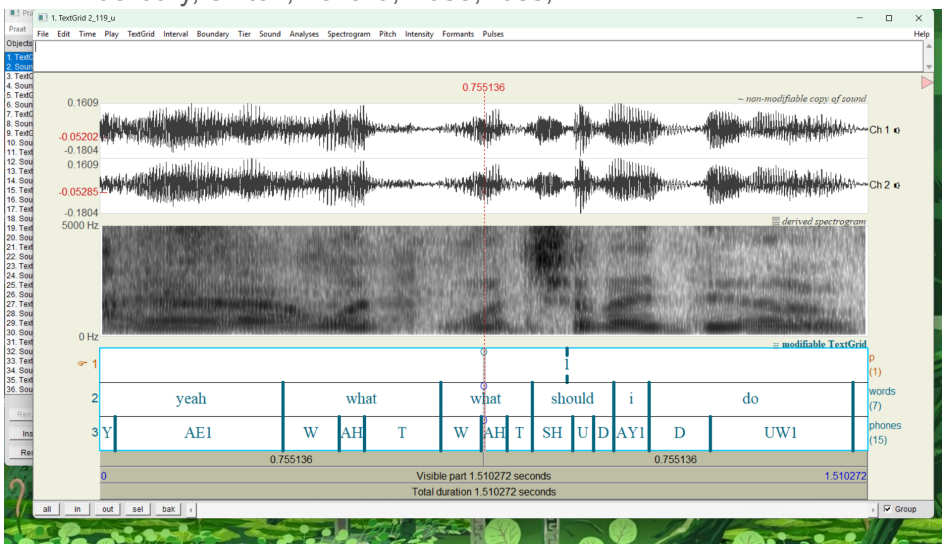
☑ ~~Display utterance length distribution*~~

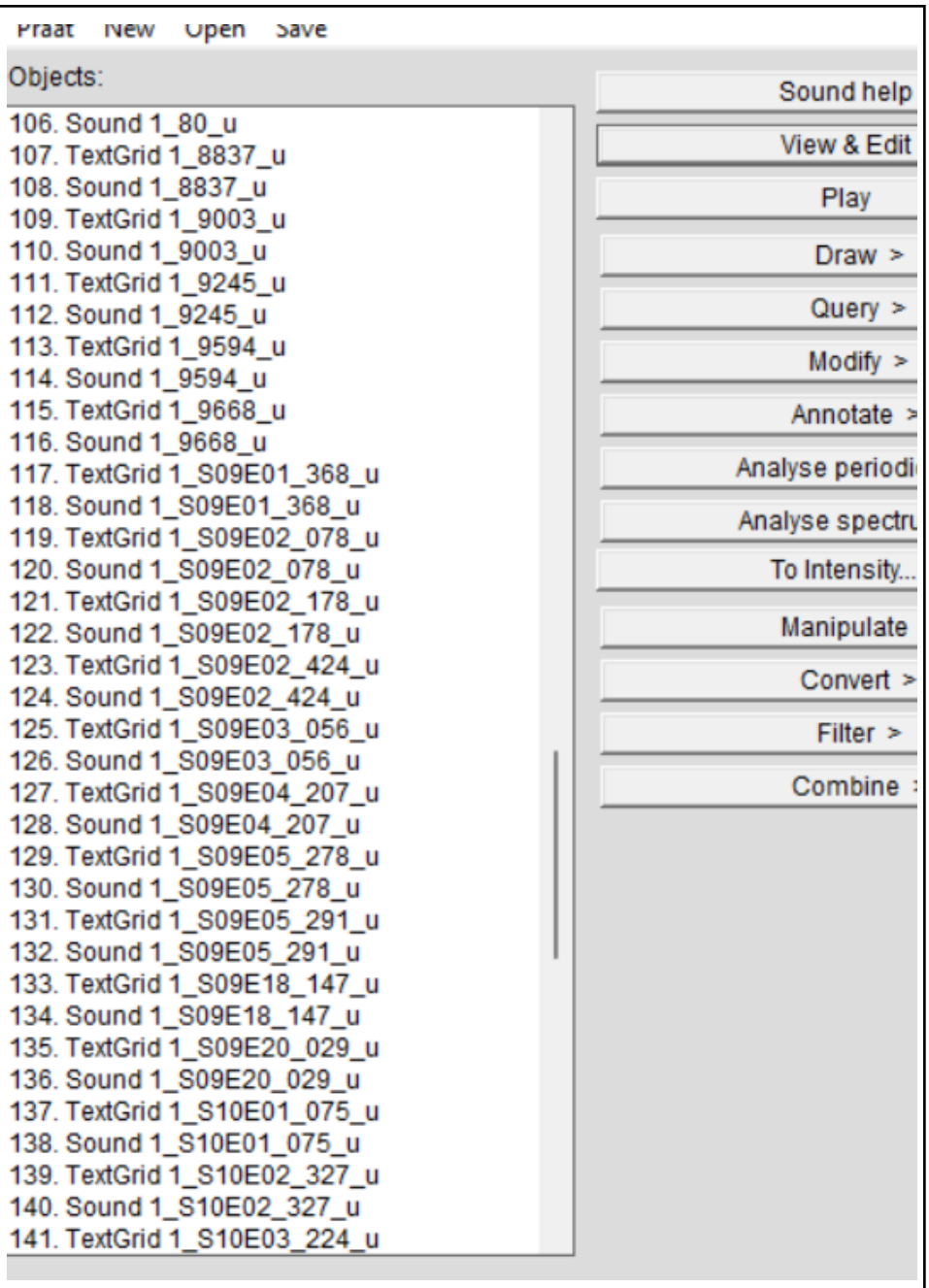


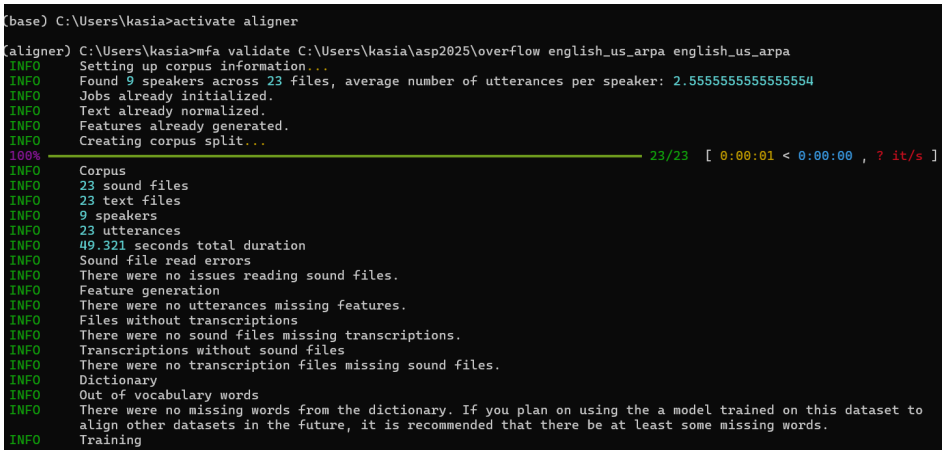
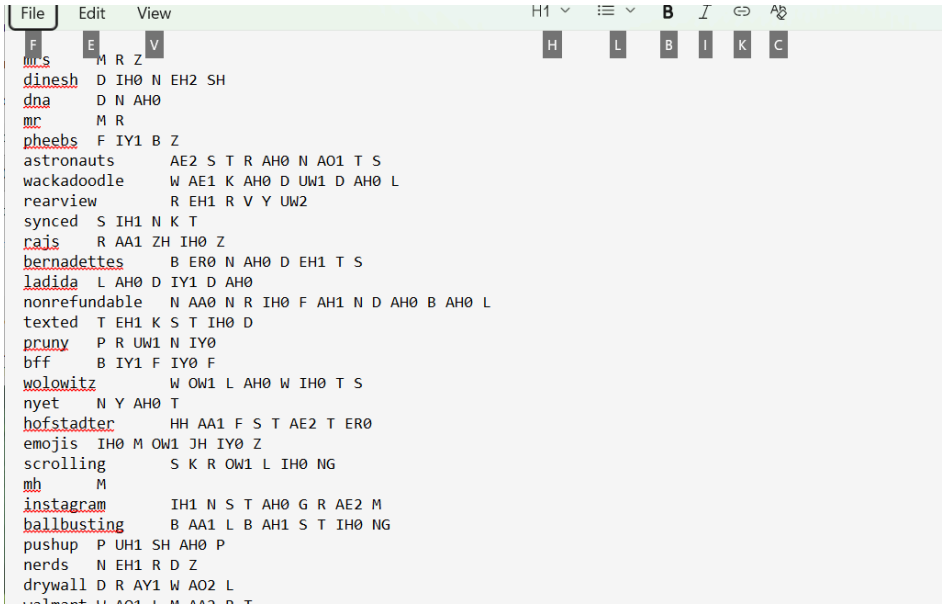
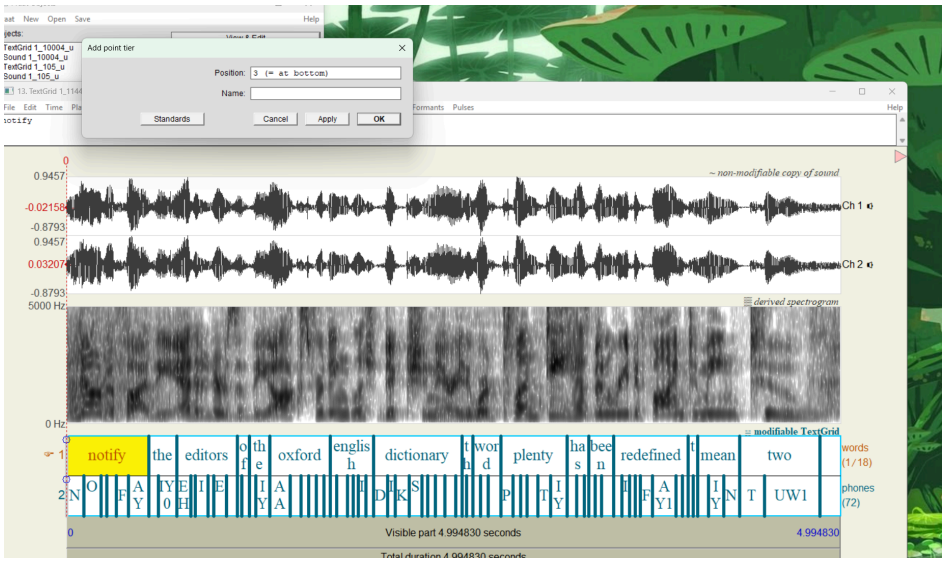
	☑ ~~Add prom for bernadettes and stuarts utterances~~



	 Notes: ● May need help marking prom ● Ask about how to debug my-voice-analysis code ● Meeting scheduling during winter break ○ B available by email, A available regularly ● Cant install wavelet ● Write intro section ● Prob not gonna use wavelet bc of trouble installing ● Remove utterances w only 1 monosyllabic words ● Try wavelet on mac ○ Create conda env w py 3.11 Summary of week: ● PRIORITY: FINISH marking utterances ● Rerun MFA and mark those utterances ● Split utterances into three parts
Dec 5	☑ ~~Mark for person1, phoebe, and rachel~~

	
Dec 6	☑ ~~Mark gilfoyle,other, Joey, leonard and monica and person3, dorothy, elritch, richard, rose, ross,~~ 
Dec 7	☑ ~~Mark howard, peenny and sheldon~~

	
Dec 8	☑ ~~Rerun MFA and mark all utterances not marked~~

	  
Dec 9	☐ Split utterance into three parts and gather pitch and intensity

stats for each section

```
7]: sought_row = plus[plus['SCENE'].str.contains("1_60_u")]
segment_length = sought_row.iloc[0]['duration']
```

```
8]: segment_length
```

```
8]: 2.4094784580498865
```

```
9]: segment_length = segment_length/3 #split into thirds, this is in seconds
#HEY! only run the above once or bad things will happen!!
```

```
0]: from pydub import AudioSegment
import math

def split_audio(file_path, segment_length=segment_length*1000):
    # Load the audio file
    audio = AudioSegment.from_file(file_path)

    # Get the total length of the audio file
    total_length = len(audio)

    # Calculate the number of segments needed
    num_segments = math.ceil(total_length / segment_length) #this number should always be three
    print(num_segments)
    # Loop through and create each segment
    for i in range(num_segments):
        start_time = i * segment_length
        end_time = min((i + 1) * segment_length, total_length)  # Ensure the last segment does not exceed total length
        segment = audio[start_time:end_time]

        # Generate the output file name
        output_file = f"{file_path[:-4]}_part{i+1}.wav"

        # Export the segment as an MP3 file
        segment.export(output_file, format="wav")
        print(f"Exported: {output_file}")
```

```
1]: split_audio(r"C:\Users\kasia\asp2025\testy\1_60_u.wav")
```

```
[41]: split_audio(r"C:\Users\kasia\asp2025\testy\1_60_u.wav")
```

```
3
Exported: C:\Users\kasia\asp2025\testy\1_60_u_part1.wav
Exported: C:\Users\kasia\asp2025\testy\1_60_u_part2.wav
Exported: C:\Users\kasia\asp2025\testy\1_60_u_part3.wav
```

```
0]: a=0
for idx, row in plus.iterrows():
    filename = row['SCENE']
    speaker = row['SPEAKER']
    fullpath = rf"C:\Users\kasia\asp2025\testy\{speaker}\{filename}.wav"
    segment_length = row['duration']
    print(segment_length)
    print(filename)
    print(speaker)
    segment_length = segment_length/3
    def split_audio(file_path, segment_length=segment_length*1000):
    # Load the audio file
        audio = AudioSegment.from_file(file_path)

    # Get the total length of the audio file
        total_length = len(audio)

    # Calculate the number of segments needed
        num_segments = 3 #this number should always be three
        print(num_segments)
    # Loop through and create each segment
        for i in range(num_segments):
            start_time = i * segment_length
            end_time = min((i + 1) * segment_length, total_length)  # Ensure the last segment does not exceed total length
            segment = audio[start_time:end_time]

        # Generate the output file name
            output_file = f"{file_path[:-4]}_part{i+1}.wav"

        # Export the segment as an MP3 file
            segment.export(output_file, format="wav")
            print(f"Exported: {output_file}")

    split_audio(fullpath)
    a=a+1

print(a)
```

- [x] ~~Add the syl/sec column~~

```
55]: plus['slypersec'] = None
     plus['slypersec'] = plus['syl_count']/plus['duration']
```

```
57]: plus = plus.drop(columns=['cmudict'])
```

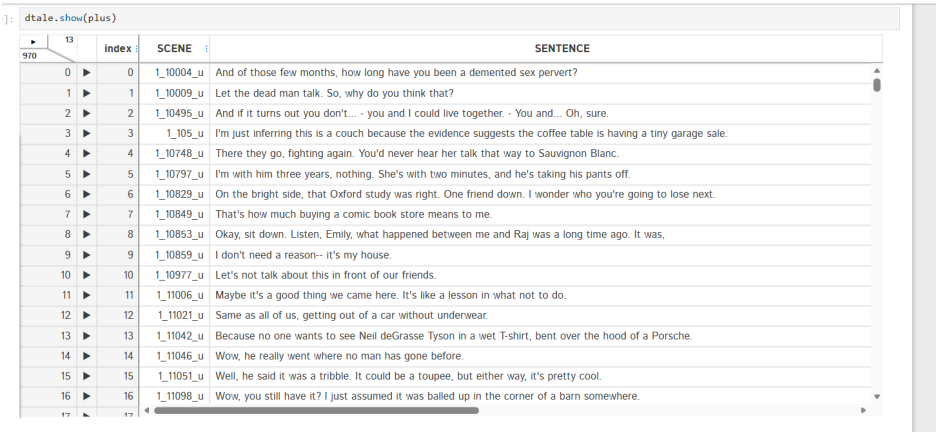
- [x] ~~Update removed utterance chart and gender distribution chart~~

	1. Use different min and max F0 based on speaker gender. (.. for males and … for females base off research) 	Male		Female	
---	---	---	---
Sarcastic	Sincere	Sarcastic	Sincere
290	297	187	196 | | Background music/sound	21	9
---	---	---
Too much cut off/too short	5	9
Unclear who speaker is	0	1	 ☑ ~~Add attardo paper to intro~~ those with a monotonal (H*) accent (Sheppard & Winters, 2022). Finally, sarcastic utterances often contain either compressed pitch, pronounced pitch, or within statement contrast (Attardo et al. 2003). Pitch range is suppressed throughout the utterance, or pitch accents are placed on every word and sometimes multiple syllables within a word. Alternatively, utterances with strong within statement contrast contain a phrase with high pitch range and a phrase with reduced pitch range. This pitch pattern usually manifests as a phrase with high pitch range followed by a phrase with extremely low pitch range (Attardo et al. 2003). There are various applications for automatic sarcasm detection, including in ☑ ~~Add dr g and kale to acknowledgments~~ Notes: ● Intro basically the same as the proposal except moving the objectives into the intro ● Summarize winter break meeting schedule in email ● Clear idea of tasks over winter break
Dec 10	☐ Get global intensity stats ☐ Write list of all global features
Dec 11 💜	Meeting notes: ● Idea for feature, in what third of the utterances does prominence appear? ● Mark different types of sarcasm (1,2,3) ● Multiclass model? ● Pitch → into semitone ● Intensity → to db ● Speech rate → ● Duration of prom word ○ Normalize by speaker → how fast is this word ● Praat funt to find prom

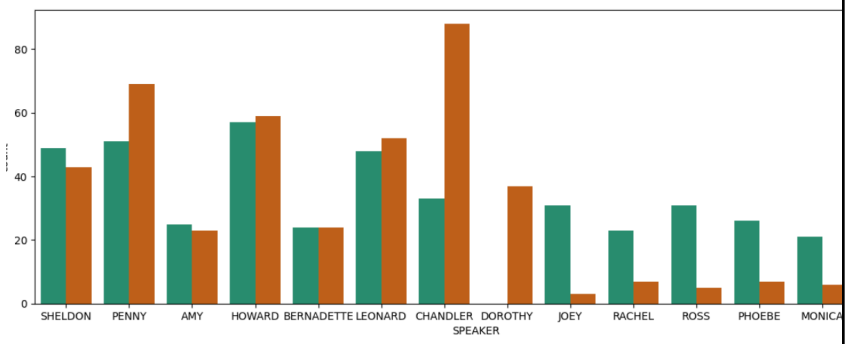
<table>
<tr>
<td></td>
<td>

Loop through word tier and get start and end time
Get timing of point and see if it falls into the word range
Call = loop through words (see brooklyns code!)

F0 experpolate function
.interpolate()
Print out pitch contour on test file
Normalize w z score in scipy
How many utterances per speaker
Delete generic speakers/ not enough utterances
Distribution of sarcasm type across speakers
Dtale exploratory data analysis auto display
Before after prom word
Ask set date of school fair

</td>
</tr>
<tr>
<td>Dec 12-14</td>
<td>
☑ ~~Install Dtale~~

dtale.show(plus)
<table>
<tr><th></th><th></th><th>index</th><th>SCENE</th><th>SENTENCE</th></tr>
<tr><td>0</td><td>▶</td><td>0</td><td>1_10004_u</td><td>And of those few months, how long have you been a demented sex pervert?</td></tr>
<tr><td>1</td><td>▶</td><td>1</td><td>1_10009_u</td><td>Let the dead man talk. So, why do you think that?</td></tr>
<tr><td>2</td><td>▶</td><td>2</td><td>1_10495_u</td><td>And if it turns out you don't... - you and I could live together. - You and... Oh, sure.</td></tr>
<tr><td>3</td><td>▶</td><td>3</td><td>1_105_u</td><td>I'm just inferring this is a couch because the evidence suggests the coffee table is having a tiny garage sale.</td></tr>
<tr><td>4</td><td>▶</td><td>4</td><td>1_10748_u</td><td>There they go, fighting again. You'd never hear her talk that way to Sauvignon Blanc.</td></tr>
<tr><td>5</td><td>▶</td><td>5</td><td>1_10797_u</td><td>I'm with him three years, nothing. She's with two minutes, and he's taking his pants off.</td></tr>
<tr><td>6</td><td>▶</td><td>6</td><td>1_10829_u</td><td>On the bright side, that Oxford study was right. One friend down. I wonder who you're going to lose next.</td></tr>
<tr><td>7</td><td>▶</td><td>7</td><td>1_10849_u</td><td>That's how much buying a comic book store means to me.</td></tr>
<tr><td>8</td><td>▶</td><td>8</td><td>1_10853_u</td><td>Okay, sit down. Listen, Emily, what happened between me and Raj was a long time ago. It was,</td></tr>
<tr><td>9</td><td>▶</td><td>9</td><td>1_10859_u</td><td>I don't need a reason-- it's my house.</td></tr>
<tr><td>10</td><td>▶</td><td>10</td><td>1_10977_u</td><td>Let's not talk about this in front of our friends.</td></tr>
<tr><td>11</td><td>▶</td><td>11</td><td>1_11006_u</td><td>Maybe it's a good thing we came here. It's like a lesson in what not to do.</td></tr>
<tr><td>12</td><td>▶</td><td>12</td><td>1_11021_u</td><td>Same as all of us, getting out of a car without underwear.</td></tr>
<tr><td>13</td><td>▶</td><td>13</td><td>1_11042_u</td><td>Because no one wants to see Neil deGrasse Tyson in a wet T-shirt, bent over the hood of a Porsche.</td></tr>
<tr><td>14</td><td>▶</td><td>14</td><td>1_11046_u</td><td>Wow, he really went where no man has gone before.</td></tr>
<tr><td>15</td><td>▶</td><td>15</td><td>1_11051_u</td><td>Well, he said it was a tribble. It could be a toupee, but either way, it's pretty cool.</td></tr>
<tr><td>16</td><td>▶</td><td>16</td><td>1_11098_u</td><td>Wow, you still have it? I just assumed it was balled up in the corner of a barn somewhere.</td></tr>
</table>

☐ Find min sample size for normalization
☑ ~~Update table on removed speakers~~
<table>
<tr><td>Generic speaker</td><td>24</td><td>48</td></tr>
<tr><td>Too small for normalization (<10)</td><td>9</td><td>8</td></tr>
</table>
<table>
<tr><th>Name</th><th>Sarcastic</th><th>Sincere</th><th>Total</th></tr>
<tr><td>Elrich</td><td>2</td><td>2</td><td>4</td></tr>
<tr><td>Other</td><td>13</td><td>16</td><td>29</td></tr>
<tr><td>Person</td><td>11</td><td>29</td><td>40</td></tr>
<tr><td>Person1</td><td>0</td><td>2</td><td>2</td></tr>
<tr><td>Person3</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>Richard</td><td>2</td><td>2</td><td>4</td></tr>
<tr><td>Rose</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>Stuart</td><td>4</td><td>4</td><td>8</td></tr>
<tr><td>Total</td><td>33</td><td>56</td><td>89</td></tr>
</table>
☐ Graph the following:
</td>
</tr>
</table>

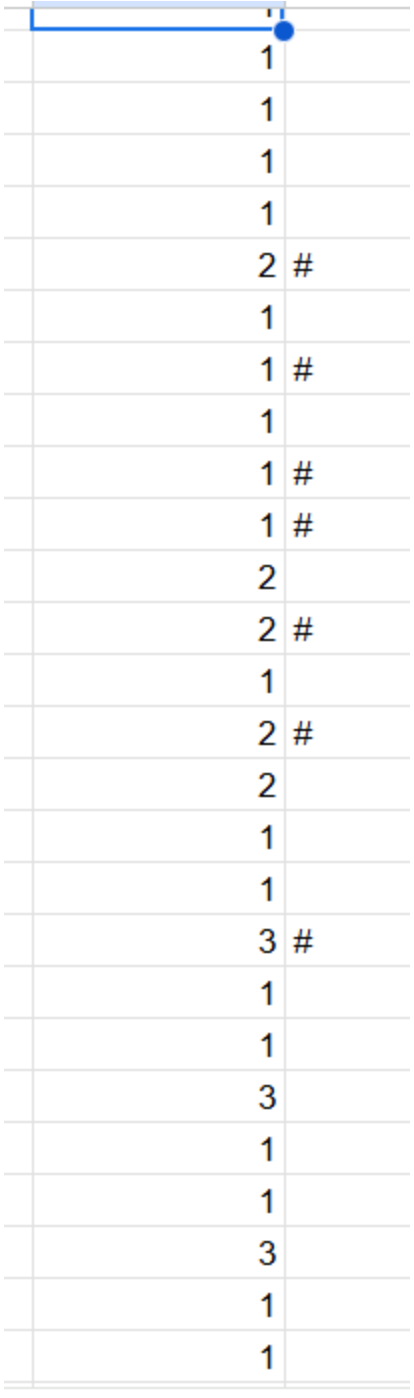
	 ☑ ~~Sarcasm/sincere utterances by speaker~~ ☐ Sarcasm type by speaker ☐ ☑ ~~Write a plan to annotate sarcastic utterances by type of sarcasm~~ 444 total sarcastic speakers, so 222 utterances over two days Email: Hi Brooklyn and Dr Anthanasopoulou Just wanted to confirm these are all the tasks I will complete over these next few weeks: 1. Remove speakers with only a few utterances and generic speakers 2. Mark the three different types of sarcasm 3. Graph distribution of sarcasm type 4. Measure pitch as semitones and interpolate it 5. Normalize global stats by speaker 6. Write a function to find the time range of the prominent word 7. Extract legendre coefficients for prominent words Have a good winter break! Andi Actually on vacation: Dec 27-jan 3
Dec 14	☑ ~~Submit intro~~ ☐ Scan paper notes into logbook
Dec 15	Notes • Log book jan 6 • School fair date: March 2 Email:

	● I removed all the generic speaker labels and any speakers with less than 10 utterances. This adds up to 89 utterances, with nearly twice as many sincere ones removed than sarcastic ones (33 vs 56) ● Distribution of sarcastic and sincere utterances overall is pretty even, 444 sarcastic and 438 sincere ○ Within speakers, distribution is less even ● For sarcasm type distribution ● The school fair is on March 2
Dec 16	☑ ~~Annotate 222 sarcastic utterances~~

	J	K
	TYPE	
	1	
	3	#
	1	
	3	#
	1	#
	1	
	2	#
	1	#
	1	#
	1	
	2	
	1	#
	2	
	2	#
	2	
	2	#
	1	#
	3	#
	1	
	1	#
	2	#
	2	#
	2	#
	2	#
	1	#
	2	

	J	
	2	
	2	
	1	
	2	
	1	
	1	
	1	
	2	
	1	
	2	#
	1	#
	2	#
	1	
	1	#
	2	
	1	
	1	#
	2	#
	1	#
	1	
	1	
	1	
	1	#
	1	#
	1	
	1	

J	
2	#
2	#
1	
1	
1	#
1	
2	#
2	#
2	#
1	
1	
1	
2	
1	
1	#
1	
2	
3	#
1	
1	#
2	#
2	#
2	#
2	
1	
1	

	 ☐ Edit function to measure pitch in semitones ☐ Normalize pitch stats with scipy
Dec 17	☑ ~~Annotate 222 sarcastic utterances~~

	1	
	1	
	3	
	1	
	1	
	1	
	2	#
	1	
	1	
	1	
	1	
	0	
	1	#
	1	
	1	
	1	
	1	
	1	
	1	
	1	
	1	
	1	
	2	
	1	
	1	
	1	

	1	
	1	
	1	
	1	
	1	
	2	#
	1	
	1	#
	2	#
	1	
	1	
	2	
	1	
	1	
	1	
	1	
	3	#
	1	
	1	
	1	#
	1	
	1	
	1	
	1	
	1	
	1	#
	1	

	J	K
	1	
	1	
	2	
	1	#
	2	
	1	#
	1	
	1	
	1	
	1	
	1	#
	1	#
	1	#
	1	
	1	#
	2	#
	1	
	2	#
	1	#
	1	
	1	
	3	#
	1	
	2	#
	1	
	1	
	2	#

	2 #
	1
	1
	1
	1
	1
	1
	1
	1
	1
	1
	1
	1
	1 #
	1 #
	1
	1
	1
	1
	1
	1 #
	2 #
	1
	1
	1
	1
	1

1	
1	
2	
1	
1	
2	#
2	
1	
1	
1	#
1	
1	#
1	
1	
1	
1	
2	
3	
1	#
1	
1	
1	
3	
1	
1	
1	

	2	
	2	#
	1	
	1	
	1	
	3	#
	3	
	3	#
	1	#
	3	
	3	
	2	#
	3	
	1	
	1	
	3	#
	3	
	3	
	3	
	3	
	3	#
	3	
	1	#
	1	#

FIND HERE: data_w_type

- [x] ~~Rewrite transcripts to show true transcripts from txt~~

[43]:
```
plus['TRANSCRIPT'] = None
base_dir = r'C:\Users\kasia\asp2025\textgrids'

for idx, key in plus['SCENE'].items():
    try:
        path = None
        for root, dirs, files in os.walk(base_dir):
            target_file = f"{key}.txt"
            if target_file in files:
                path = os.path.join(root, target_file)
                break

        if path is None:
            raise FileNotFoundError(f"srry bb I can't find {key}.wav idk...")

        with open(path, 'r') as file:
            text = file.read()
            plus.loc[idx, 'TRANSCRIPT'] = text

    except Exception as e:
        print(f"oh noses u have a problem in {path}: {e}")
```

- ☑ ~~Email: ask about single word questions~~
  - ☑ ~~Note: Sheldons style was the hardest!~~
- ☑ ~~Email: double check unsure types~~

Dec 18

- ☑ ~~Write code to input sarcasm type into the original df~~

[64]:
```
all_1 = []
df = pd.read_csv('data_w_type.csv')
ones = df[df['TYPE'] == 1]

for idx, key in ones['SCENE'].items():
    scene = df.loc[idx, 'SCENE']
    all_1.append(scene)
    #print(scene)
#print(all_1)

for idx, key in plus['SCENE'].items():
    scene_true = plus.loc[idx, 'SCENE']
    if scene_true in all_1:
        plus.loc[idx, 'TYPE'] = 1

##########################################################################
all_2 = []
df = pd.read_csv('data_w_type.csv')
twos = df[df['TYPE'] == 2]

for idx, key in twos['SCENE'].items():
    scene = df.loc[idx, 'SCENE']
    all_2.append(scene)
    #print(scene)
#print(all_1)

for idx, key in plus['SCENE'].items():
    scene_true = plus.loc[idx, 'SCENE']
    if scene_true in all_2:
```

```
all_2 = []
df = pd.read_csv('data_w_type.csv')
twos = df[df['TYPE'] == 2]

for idx, key in twos['SCENE'].items():
    scene = df.loc[idx, 'SCENE']
    all_2.append(scene)
    #print(scene)
#print(all_1)

for idx, key in plus['SCENE'].items():
    scene_true = plus.loc[idx, 'SCENE']
    if scene_true in all_2:
        plus.loc[idx, 'TYPE'] = 2
################################################################################
all_3 = []
df = pd.read_csv('data_w_type.csv')
threes = df[df['TYPE'] == 3]

for idx, key in threes['SCENE'].items():
    scene = df.loc[idx, 'SCENE']
    all_3.append(scene)
    #print(scene)
#print(all_1)

for idx, key in plus['SCENE'].items():
    scene_true = plus.loc[idx, 'SCENE']
    if scene_true in all_3:
        plus.loc[idx, 'TYPE'] =

plus.fillna(0)
plus.head()
```

☑ ~~Update removed table to add the 4 removed utterances~~

Reason	Sarcastic	Sincere
Not an American accent	42	32
Loud	1	1
Background music/sound	25	9
Too much cut off/too short	5	9
Unclear who speaker is	0	1
Generic speaker	24	48
Too small for normalization (<10)	9	8

Email:

Hi Dr Anthanasopoulou and Brooklyn

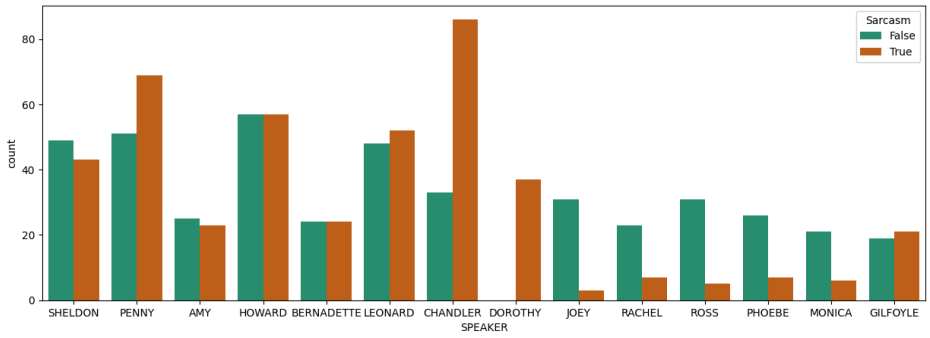
I removed 89 utterances from speakers with <10 utterances or are general labels. I have labeled the three types of sarcasm, where 1 is within phrase contrast, 2 is where all words have pitch accents, and 3 is monotone. 0 is for sincere utterances.

There are a couple utterances that I'm unsure about, I was wondering if

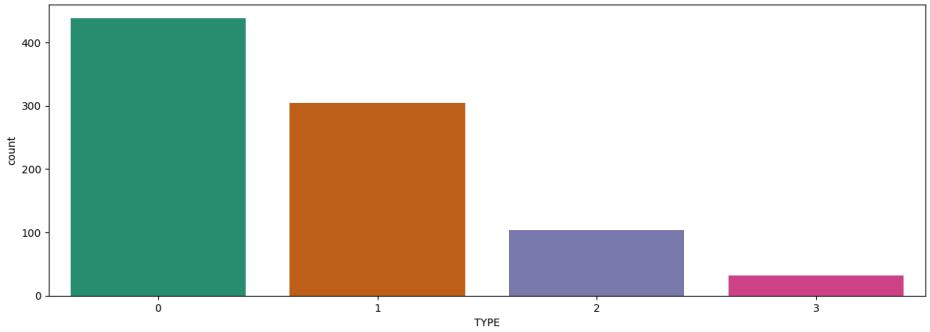
either of you could double check some.

1_105_u	3
1_10829_u	3
1_11006_u	1
1_11098_u	2
1_11177_u	1
1_11224_u	1
1_11322_u	1
1_11439_u	2
1_11697_u	2
1_11773_u	1

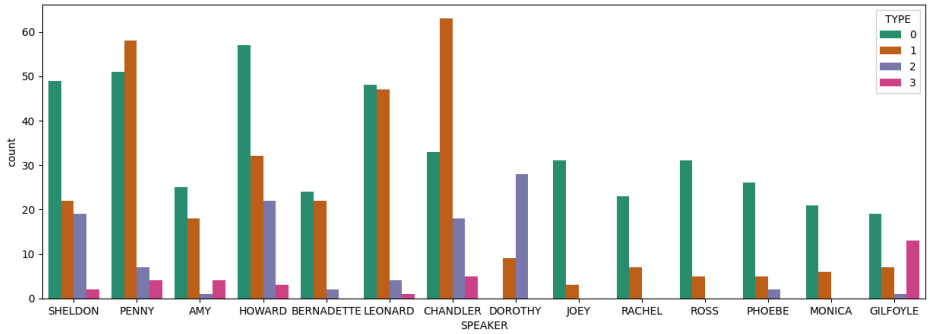
The overall sarcasm distribution is not that even across speakers, but the overall balance of sarcastic/ sincere is very even, 440 sarcastic and 438 sincere.



In terms of sarcasm type, within phrase contrast is the most common, which I expected.



Speakers also appear to have their own style? Within phrase contrast is the most common type in all speakers except for Gilfoyle and heavily accented with Dorothy.

	 This doesn't show on the data, but it sounds like most within contrast sarcasm has one strong pitch accent in one sentence, while Sheldon follows a formula of "Single sarcastic sounding word", a pause, and then "sincere statement" For normalization, I'm adapting Brooklyn's code to fit my own. I'm not sure if I understand semitones correctly, but converting pitch from Hz to semitones sorta "normalizes" it because I'm providing a different mean pitch for each speaker, right? I also updated the transcripts to reflect the actual Textgrid because some of the old ones had words that weren't said. Now the speaking rate should be more accurate (yikes, glad I caught that one). Next, I will work on getting the time range of prominent words and normalizing speaking rate by speaker. Should I also make a measurement of the normalized speech rate of just the prominent word? Andi
Dec 18-20	☑ ~~Normalize hz to semitones~~

5]:
```
#finds mean pitch in hz for each speaker
from statistics import mean
speaker_listUnique = plus['SPEAKER'].unique()
speaker_list = plus["SPEAKER"]
speaker_meanF0 = plus["mean_F0"]

speakerTotal = {}
speakeravgSum = {}
speakeravg = {}

for i in speaker_listUnique:
    speakerTotal[i] = 0
    speakeravgSum[i] = 0
print(speaker_list)
print(len(speaker_list))
for i in range(len(speaker_list)):
    for a in speaker_listUnique:
        if speaker_list[i] == a:
            speakerTotal[a] += 1
            speakeravgSum[a] += int(speaker_meanF0[i])
for i in speaker_listUnique:
    speakeravg[i] = speakeravgSum[i]/speakerTotal[i]

print(speakeravg)
```

```
0       SHELDON
  3, 'PHOEBE': 311.3939393939394, 'MONICA': 305.40740740740
```

[41]:
```
plus["speaker_mean_f0"] = None
for idx, key in plus['SCENE'].items():
    speaker = plus.loc[idx, "SPEAKER"]
    #g = np.where(speaker_listUnique == speaker)[0]
    mean = speakeravg[speaker]
    plus.loc[idx, "speaker_mean_f0"] = mean
```

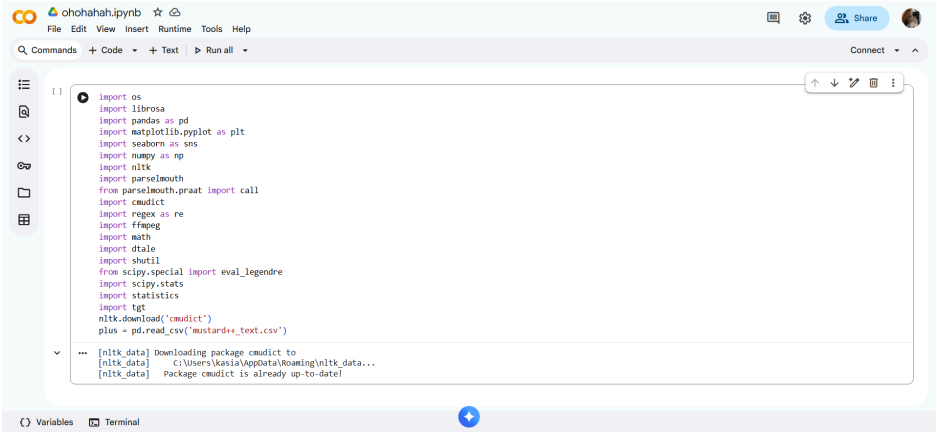
- [x] ~~Modify Brooklyn's code to find the range of prominent words~~

```
]: for idx, key in plus['SCENE'].items():
    try:
        wpath = None
        for root, dirs, files in os.walk(base_dir):
            target_file = f"{key}.TextGrid"
            if target_file in files:
                path = os.path.join(root, target_file)
                break

        if path is None:
            raise FileNotFoundError(f"srry bb I can't find {key} idk...")

        ###find time interval of all words in textgrid
        word_intervals = get_times_from_textgrid(path, "words")
        prom_intervals = get_times_from_textgrid(path, "p")
        prom_timepoint = prom_intervals[0][1]
        #print(path)
        #print(prom_intervals)
        #print(prom_timepoint)
        #print(word_intervals)

        for word in word_intervals:
            start = word[1]
            end = word[2]
            if start < prom_timepoint < end:
                prom_start = start
                prom_end = end
                break
        print(f"start of prom word: {prom_start}")
        print(prom_end)
        print(path)
```

☑ ~~Interpolate pitch in hz~~

```
try: # this deals with if the extracted part is shorter than the window length
    pitch_contour = sound_interval.to_pitch().interpolate().selected_array['frequency']
```

☑ ~~Upload code to google collab~~



Dec 21-22	☑ ~~Add prominence tier for unmarked utterances~~

```
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\PENNY\1_3649_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\PENNY\1_3649_u.TextG
rid does not have a tier called "p".
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\PENNY\1_3660_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\PENNY\1_3660_u.TextG
rid does not have a tier called "p".
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\PENNY\1_5166_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\PENNY\1_5166_u.TextG
rid does not have a tier called "p".
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\PENNY\1_537_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\PENNY\1_537_u.TextGri
d does not have a tier called "p".
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\PENNY\1_5617_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\PENNY\1_5617_u.TextG
rid does not have a tier called "p".
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\PENNY\1_6188_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\PENNY\1_6188_u.TextG
rid does not have a tier called "p".
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\PENNY\1_6427_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\PENNY\1_6427_u.TextG
rid does not have a tier called "p".
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\PENNY\1_S09E23_420_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\PENNY\1_S09E23
_420_u.TextGrid does not have a tier called "p".
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\LEONARD\1_S11E08_305_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\LEONARD\1_S1
1E08_305_u.TextGrid does not have a tier called "p".
```

```
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\CHANDLER\2_104_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\CHANDLER\2_104_u.T
extGrid does not have a tier called "p".
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\CHANDLER\2_109_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\CHANDLER\2_109_u.T
extGrid does not have a tier called "p".
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\CHANDLER\2_115_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\CHANDLER\2_115_u.T
extGrid does not have a tier called "p".
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\CHANDLER\2_120_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\CHANDLER\2_120_u.T
extGrid does not have a tier called "p".
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\CHANDLER\2_124_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\CHANDLER\2_124_u.T
extGrid does not have a tier called "p".
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\CHANDLER\2_134_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\CHANDLER\2_134_u.T
extGrid does not have a tier called "p".
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\ROSS\2_152_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\ROSS\2_152_u.TextGrid
does not have a tier called "p".
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\CHANDLER\2_235_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\CHANDLER\2_235_u.T
extGrid does not have a tier called "p".
oh noses u have a problem in C:\Users\kasia\asp2025\textgrids\CHANDLER\2_464_u.TextGrid: Textgrid C:\Users\kasia\asp2025\textgrids\CHANDLER\2_464_u.T
extGrid does not have a tier called "p".
```

☑ ~~Convert intensity measurements to z-score~~

```
###################################
db_z = norm_intensity(intensity_contour)
mean_db_z = get_mean(db_z)
range_db_z = get_range(db_z)
sd_db_z = get_sd(db_z)

plus.loc[idx, ['mean_F0', 'sd_F0', 'range_F0', 'duration', 'mean_db_z','range_db_z','sd_db_z']] = [
    mean_F0, sd_F0, range_F0, duration, mean_db_z,range_db_z,sd_db_z
]
```

```
plus['intensity_contour'] = plus['intensity_contour'].astype(object)
for idx, key in plus['SCENE'].items():
    try:
        wav_path = None
        for root, dirs, files in os.walk(base_dir):
            target_file = f"{key}.wav"
            if target_file in files:
                wav_path = os.path.join(root, target_file)
                break

        if wav_path is None:
            raise FileNotFoundError(f"srry bb I can't find {key}.wav idk...")

        sound = parselmouth.Sound(wav_path)

        duration = findDuration(sound)
        pitch_contour, intensity_contour = get_contours(sound, 0.0, duration, keep_zeros=False)
        mean_F0 = get_mean(pitch_contour)
        range_F0 = get_range(pitch_contour)
        sd_F0 = get_sd(pitch_contour)
        ###################################
        mean_db = get_mean(intensity_contour)
        sd_db = get_sd(intensity_contour)
        db_z = norm_intensity(intensity_contour)
        mean_db_z = get_mean(db_z)
        range_db_z = get_range(db_z)
        sd_db_z = get_sd(db_z)

        plus.loc[idx, ['mean_F0', 'sd_F0', 'range_F0', 'duration', 'mean_db_z','range_db_z','sd_db_z', 'mean_db', 'sd_db']] = [
            mean_F0, sd_F0, range_F0, duration, mean_db_z,range_db_z,sd_db_z, mean_db, sd_db
        ]
        plus.at[idx, 'intensity_contour'] = intensity_contour
```

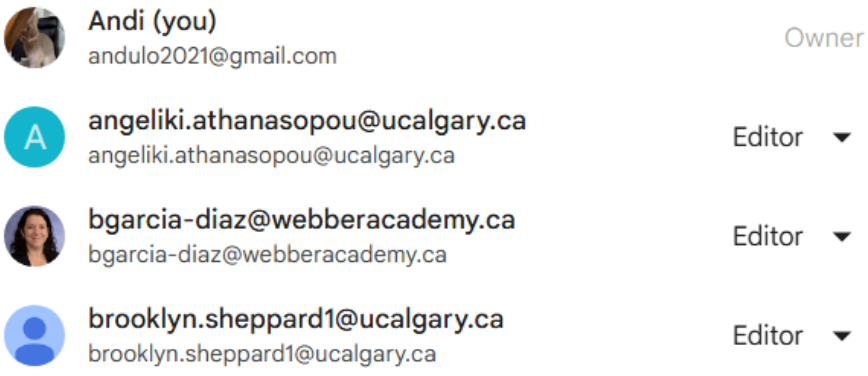
```
speaker_contours = {} #ANDI, don't print this out! It will crash Jupyter notebook!!!! >:(
speaker_contours_means = {}
speaker_contours_sds = {}
for idx, row in plus.iterrows():
    speaker = row["SPEAKER"]
    contour = row["intensity_contour"]

    if speaker not in speaker_contours:
        speaker_contours[speaker] = []

    speaker_contours[speaker].extend(contour)

for speaker in speaker_contours:
    ind_contour = speaker_contours[speaker]
    mean = sum(ind_contour)/len(ind_contour)
    sd = np.std(ind_contour, ddof=0) # 0 bc this is the population mean
    speaker_contours_means[speaker] = mean
    speaker_contours_sds[speaker] = sd
```

```
print(speaker_contours_means)
```

```
{'SHELDON': np.float64(66.01762795464742), 'PENNY': np.float64(61.01437839107055), 'AMY': np.float64(59.994150307421634), 'HOWARD': np.float64(60.3063923248959
7), 'BERNADETTE': np.float64(58.25437145334782), 'LEONARD': np.float64(61.741101159326995), 'CHANDLER': np.float64(62.745867011299204), 'DOROTH
Y': np.float64(80.30339754053833), 'JOEY': np.float64(59.43885235257127), 'RACHEL': np.float64(59.37214743968287), 'ROSS': np.float64(58.5704515640799
3), 'PHOEBE': np.float64(59.47445572668255), 'MONICA': np.float64(58.55259316341727), 'GILFOYLE': np.float64(63.48519826492458)}
```

```
plus['mean_db_fix'] = None
plus['sd_db_fix'] = None
plus['range_db_fix'] = None

for idx, key in plus['SCENE'].items():
    try:
        wav_path = None
        for root, dirs, files in os.walk(base_dir):
            target_file = f"{key}.wav"
            if target_file in files:
                wav_path = os.path.join(root, target_file)
                break

        if wav_path is None:
            raise FileNotFoundError(f"srry bb I can't find {key}.wav idk...")

        sound = parselmouth.Sound(wav_path)

        duration = findDuration(sound)
        pitch_contour, intensity_contour = get_contours(sound, 0.0, duration, keep_zeros=False)
        speaker = plus.loc[idx, 'SPEAKER']
        db_z_fix = (intensity_contour - speaker_contours_means[speaker])/speaker_contours_sds[speaker]
        mean_db_fix = get_mean(db_z_fix)
        range_db_fix = get_range(db_z_fix)
        sd_db_fix = get_sd(db_z_fix)

        plus.loc[idx, ['mean_db_fix','range_db_fix', 'sd_db_fix']] = [mean_db_fix, range_db_fix, sd_db_fix]

    except Exception as e:
        print(f"oh noses u have a problem in {wav_path}: {e}")
```

- [x] ~~Add Brooklyn as collaborator~~



- [x] ~~Find speaking rate of only prom word~~

```
1]: def get_times_from_textgrid(path_to_textgrid, tier_name):
        """
        Extract bounadary times for each interval and its annotation
        :param path_to_textgrid: full path to the textgrid file
        :param tier_name: the tier you want to extract annotations from
        :return: List of lists: each list if os the form [annotation, start_time, end_time] for each interval in the textgrid
        """
        textgrid = tgt.io.read_textgrid(path_to_textgrid)
        tier = textgrid.get_tier_by_name(tier_name)
        intervals = []
        for interval in tier:
            curr_interval = []
            curr_interval.append(interval.text)
            curr_interval.append(interval.start_time)
            curr_interval.append(interval.end_time)
            intervals.append(curr_interval)
        return intervals

    # Use Legendre polynomials to model pitch and intensity
    def get_legendres(contour, n=3): # This operates at word-level. Get a legendre for each word
        """
        Perform Legendre poplynomial expansion on a contour of any acoustic measure
        :param contour: A list of values representing the contour (e.g., pitch or intensity) you want Legendre coefficients for
        :param n: order of legendre polynomials you want coefficients for
        :return: List of n legendre coefficients (i.e. list[0] = 0th legendre coefficient)
        """
        all_legendres = []

        num_xs_contour = len(contour)
        x_vals_contour = np.linspace(-1, 1, num_xs_contour)
        contour_sampled_legendres = [eval_legendre(i, x_vals_contour) for i in range(n)]
        fitted_contours = [2*np.dot(contour, contour_sampled_legendres[i])/num_xs_contour for i in range(len(contour_sampled_legendres))]
```

```
        all_legendres += fitted_contours
        return all_legendres

    # Helper function to find specific file in a given diretory
    def find_file(filename, search_path):
      for root, dir, files in os.walk(search_path):
          if filename in files:
             result=os.path.join(root, filename)
             return result
```

```
[184]: #get_times_from_textgrid(r"C:\Users\kasia\asp2025\textgrids\SHELDON\1_10004_u.TextGrid", "words")
```

```
[185]: #get_times_from_textgrid(r"C:\Users\kasia\asp2025\textgrids\SHELDON\1_10004_u.TextGrid", "p")
```

```
[186]: #this finds the time range for only prom words
plus['cmudict'] = None
plus['prom_syl_count'] = None
plus["prom_duration"] = None
plus['prom_word'] = None
plus['prom_slypersec'] = None
for idx, key in plus['SCENE'].items():
    try:
        path = None
        for root, dirs, files in os.walk(base_dir):
            target_file = f"{key}.TextGrid"
            if target_file in files:
                path = os.path.join(root, target_file)
                break

        if path is None:
            raise FileNotFoundError(f"srry bb I can't find {key} idk...")

        ###find time interval of all words in textgrid
```

```
        word_intervals = get_times_from_textgrid(path, "words")
        prom_intervals = get_times_from_textgrid(path, "p")
        prom_timepoint = prom_intervals[0][1]
        #print(path)
        #print(prom_intervals)
        #print(prom_timepoint)
        #print(word_intervals)

        for word in word_intervals:
            start = word[1]
            end = word[2]
            if start < prom_timepoint < end:
                """
                prom_start = start
                prom_end = end
                """
                prom_word = word[0]
                break
        prom_duration = end-start
        plus.loc[idx, ["prom_duration", 'prom_word']] = [prom_duration, prom_word]
        #plus = plus.drop(columns=['cmudict'])
        """
        print(path)
        print(f"start time {prom_start}")
        print(f"end time {prom_end}")
        print(f"prom word {prom_word}")
        """
    except Exception as e:
        print(f"oh noses u have a problem in {path}: {e}")
```

```
87]: plus['cmudict'] = plus['prom_word'].astype(str).apply(cmudict_out)
plus['prom_syl_count'] = plus['cmudict'].astype(str).apply(lambda x: len(re.findall(r'\d+', x)))
plus['prom_slypersec'] = plus['prom_syl_count']/plus['prom_duration']
```

☑ ~~Send update email~~

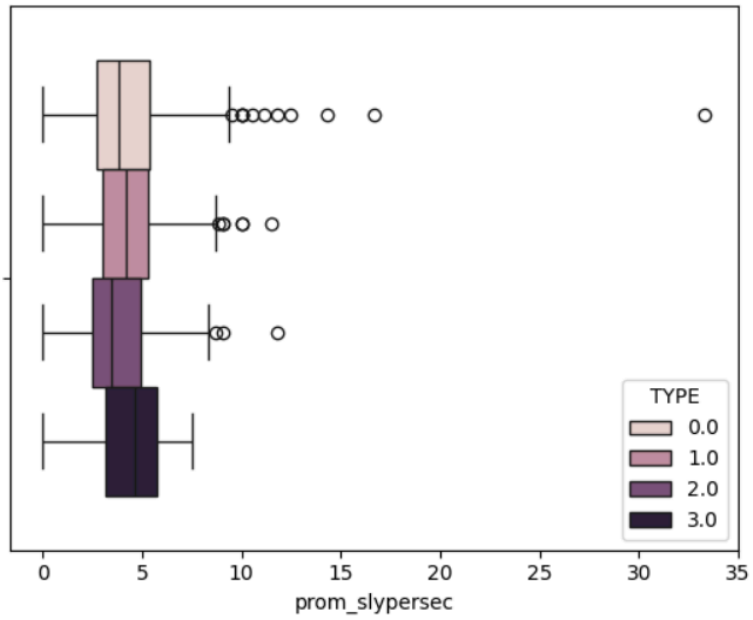
Email:

Hi Brooklyn and Dr Anthanasopoulou

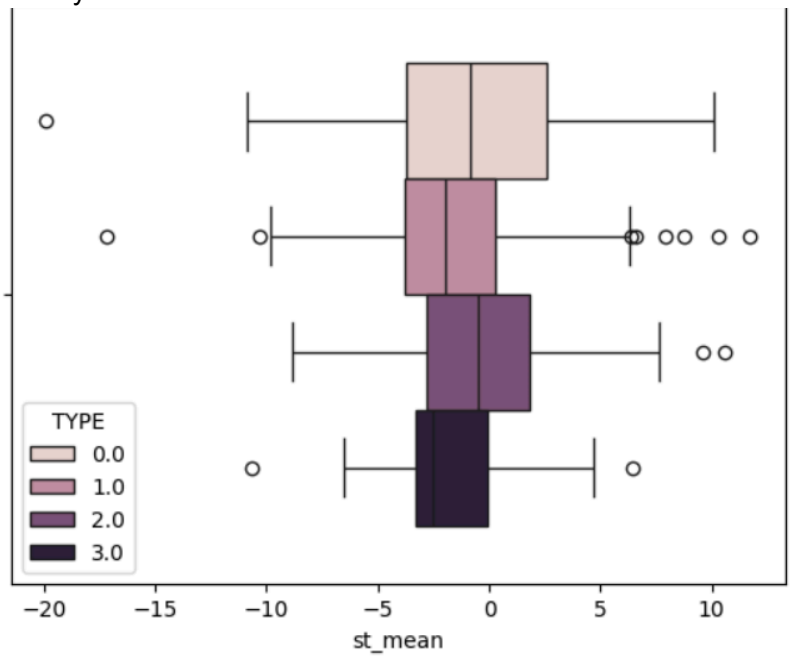
I converted Hz to semitones by calculating the mean pitch for each speaker as the reference pitch (I also added .interpolate() when finding the pitch contour so hopefully the numbers are more accurate). With the current function of converting db to z-scores, I feel like it's only helpful for measuring the db range of each utterance because it treats each utterance as its own distribution, so I rewrote part of the code to calculate the mean and SD of intensity for the speaker, and use those numbers to calculate the z score for the utterance.

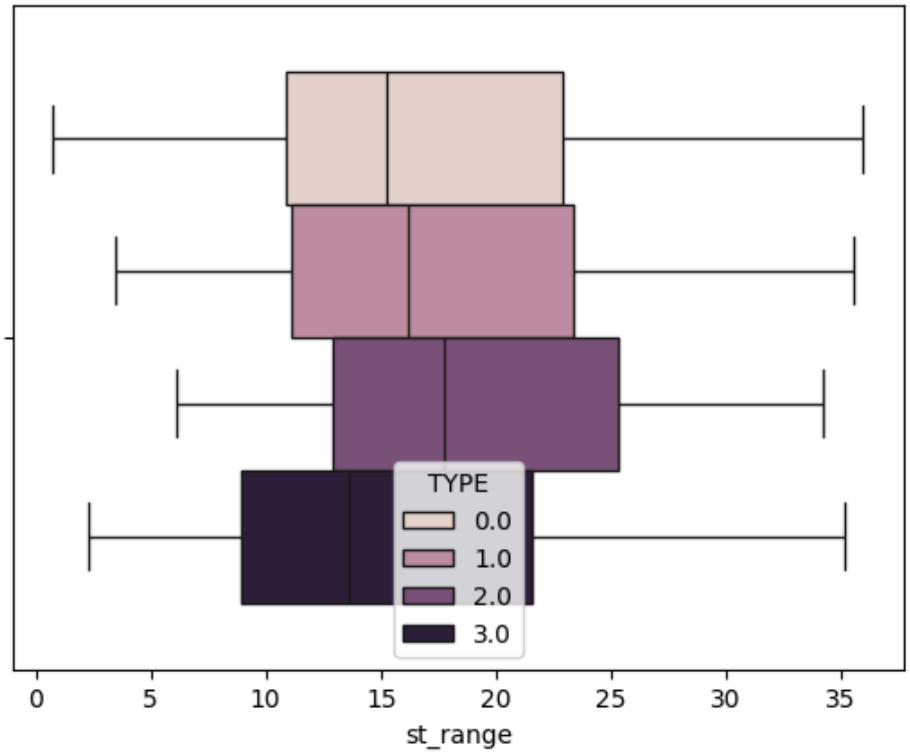
I found the duration of the prominent words, and calculated the speech rate. Overall, the difference in mean speech rate of the prominent words between sarcastic and sincere utterances is only a few tenths of a syllable/second. Generally, prominent words in sarcastic utterances are said slower which isn't surprising. When looking at sarcasm types, the slowest type is when all words have pitch accents and the fastest is monotone. Maybe because all pitch accents are more exaggerated than within phrase contrast, the speed at which the word is said in all pitch accents is slower as well?

The difference in speech rate of prominent words between the types is overall not that big (probably not statistically significant). There's lots of outliers from really fast one word utterances.



For mean pitch in semitones, there is also little difference between types of sarcasm and sincere utterances. There is less variance in

	mean pitch for monotone sarcastic utterances, which could be due to there being a set monotone tone of voice? Also, most monotone utterances came from one person, so most monotone utterances may sound really similar.  Distribution of pitch range in semitones sorta follows what the other paper observed. Monotone has the lowest pitch range, and all pitch accent has the highest. I think it would be more helpful to divide the within contrast utterances to before and after the prominent word.

	 Looking at what I've measured so far, none of the features seem too helpful in detecting sarcasm. Perhaps measuring Legendre coefficients will be more helpful. Technically, I now have all the baseline global features. Next, I will collect the Legendre coefficients for prominent words and cluster them. Andi
Winter Break ☃️	☑ ~~Extract features~~
Dec 23	☑ ~~Fix prom words with sly count of 0~~

7]:
```
dtale.show(plus)
```

Filter: prom_syl_count == 0

	mean_db_fix	sd_db_fix	range_db_fix	syl_count	slypersec	cmudict	prom_syl_count	prom_duration	prom_word	prom_slypersec
0	-0.28	0.63	2.72	15	5.38	[]	0	0.48	dennys	0.00
1	0.87	0.65	3.14	27	4.02	[]	0	0.52	pruny	0.00
2	1.15	0.55	2.55	6	1.70	[]	0	1.09	bff	0.00
3	1.02	0.41	2.04	17	4.66	[]	0	0.58	ladida	0.00
4	0.89	0.55	2.11	8	4.60	[]	0	0.29	thats	0.00
5	0.08	0.83	3.75	6	3.53	[]	0	0.43	didnt	0.00
6	-0.10	0.75	3.06	13	4.67	[]	0	0.57	sasslers	0.00
7	-0.31	1.12	4.53	7	3.39	[]	0	0.86	ballbusting	0.00
8	0.12	0.61	2.26	7	3.07	[]	0	0.73	batmobile	0.00
9	-0.42	0.83	3.57	14	5.16	[]	0	0.52	coachella	0.00
10	-0.28	0.62	2.56	7	4.76	[]	0	0.54	wackadoodle	0.00
11	0.33	0.70	3.09	3	2.16	[]	0	0.83	cappucino	0.00
12	-0.97	0.80	3.34	10	4.10	[]	0	0.55	gazara	0.00
13	0.74	0.54	2.28	18	6.04	[]	0	0.33	thats	0.00
14	0.22	0.93	4.64	8	3.06	[]	0	0.74	liberarche	0.00
15	0.02	0.70	3.56	9	4.60	[]	0	0.56	unagi	0.00

]:
```
plus.loc[[129, 553,627,678],"prom_syl_count"] = 1
plus.loc[[40, 44, 264,271,639,675, 703,760],"prom_syl_count"] = 2
plus.loc[[53, 115, 408,432,459,533,602,729,734, 754,], "prom_syl_count"] = 3
plus.loc[[514,526,563,797],"prom_syl_count"] = 4
```

- [ ] Normalize speech rate by speaker

- [x] ~~Use textgrid text as the transcript~~

5]:
```
#get transcript
base_dir = r'C:\Users\kasia\asp2025\textgrids'
plus['TRANSCRIPT'] = None
#this finds the time range for only prom words
for idx, key in plus['SCENE'].items():
    try:
        path = None
        for root, dirs, files in os.walk(base_dir):
            target_file = f"{key}.TextGrid"
            if target_file in files:
                path = os.path.join(root, target_file)
                break

        if path is None:
            raise FileNotFoundError(f"srry bb I can't find {key} idk...")

        ###find time interval of all words in textgrid
        word_intervals = get_times_from_textgrid(path, "words") #list of lists

        all_words = []
        num_words = len(word_intervals) - 1
        n = 0
        while n <= num_words:
            word = word_intervals[n][0]
            all_words.append(word)
            n += 1
        #print(all_words)
        transcript = ' '.join(all_words)
        #print(transcript)
        plus.loc[idx, 'TRANSCRIPT'] = transcript

    except Exception as e:
```

- [x] ~~Get legendres for prominent words of intensity and pitch~~

```
        raise FileNotFoundError(f"srry bb I can't find {key} idk...")
    #print(type(wav_path))
    sound = parselmouth.Sound(wav_path)
    prom_contour_pitch, prom_contour_intensity = get_contours (sound, prom_start , prom_end, keep_zeros = False)
    #print(type(prom_contour_pitch))
    prom_leg_pitch = get_legendres(prom_contour_pitch , n=4)
    pitch_1 = prom_leg_pitch[0]
    pitch_2 = prom_leg_pitch[1]
    pitch_3 = prom_leg_pitch[2]
    pitch_4 = prom_leg_pitch[3]
    prom_leg_intensity = get_legendres(prom_contour_intensity , n=4)
    intensity_1 = prom_leg_intensity[0]
    intensity_2 = prom_leg_intensity[1]
    intensity_3 = prom_leg_intensity[2]
    intensity_4 = prom_leg_intensity[3]
    plus.loc[idx, ['prom_1_pitch','prom_2_pitch','prom_3_pitch','prom_4_pitch',
                   'prom_1_intensity','prom_2_intensity','prom_3_intensity','prom_4_intensity']] = [[
        pitch_1, pitch_2, pitch_3, pitch_4, intensity_1, intensity_2, intensity_3, intensity_4]]
    #print(pitch_1)
    #print(pitch_2)
    print(prom_leg_intensity)
except Exception as e:
```

```
for idx, key in plus['SCENE'].items():
    try:
        path = None
        for root, dirs, files in os.walk(base_dir):
            target_file = f"{key}.TextGrid"
            if target_file in files:
                path = os.path.join(root, target_file)
                break

        if path is None:
            raise FileNotFoundError(f"srry bb I can't find {key} idk...")

        ###find time interval of all words in textgrid
        word_intervals = get_times_from_textgrid(path, "words")
        prom_intervals = get_times_from_textgrid(path, "p")
        prom_timepoint = prom_intervals[0][1]

        for word in word_intervals:
            start = word[1]
            end = word[2]
            if start < prom_timepoint < end:
                prom_start = start
                prom_end = end
                prom_word = word[0]
                break
        #prom_duration = prom_end-prom_start
        try:
            wav_path = None
            for root, dirs, files in os.walk(base_dir):
                target_file = f"{key}.wav"
                if target_file in files:
                    wav_path = os.path.join(root, target_file)
                    break

            if wav_path is None:
                raise FileNotFoundError(f"srry bb I can't find {key} idk...")
            #print(type(wav_path))
            sound = parselmouth.Sound(wav_path)
            prom_contour_pitch, prom_contour_intensity = get_contours (sound, prom_start , prom_end, keep_zeros = False)
            #print(type(prom_contour_pitch))
            prom_leg_pitch = get_legendres(prom_contour_pitch , n=4)
            prom_leg_intensity = get_legendres(prom_contour_intensity , n=4)
            #print(prom_leg_pitch)
        except Exception as e:
            print(f"oh noses u have a problem in {wav_path}: {e}")
        """
        print(path)
        print(f"start time {prom_start}")
        print(f"end time {prom_end}")
        print(f"prom word {prom_word}")
        """
    except Exception as e:
        print(f"oh noses u have a problem in {path}: {e}")
```

```
[np.float64(271.0350073832898), np.float64(13.8427118897952888), np.float64(17.85040990805886), np.float64(10.166815460382534)]
[np.float64(424.1736747164957), np.float64(-19.71920235077945), np.float64(26.611635537844798), np.float64(-3.2864125045142360)]
[np.float64(227.3745416518738), np.float64(-7.1317745907020078), np.float64(25.235334374180866), np.float64(-1.8756187794903951)]
[np.float64(402.6741134764438), np.float64(12.844299468953137), np.float64(-7.970409924138084), np.float64(-7.877249162742160)]
[np.float64(391.7682391852966), np.float64(-1.2382142549825128), np.float64(6.489062842120735), np.float64(-8.183069011333968)]
[np.float64(323.61724499263806), np.float64(-2.0450812997441554), np.float64(12.711385151126395), np.float64(-1.7146149516519036)]
[np.float64(352.00979833824084), np.float64(14.994628536945493), np.float64(24.75997439352782), np.float64(6.995089846697876)]
[np.float64(200.73941497873588), np.float64(4.7315826061027490), np.float64(11.580163128616803), np.float64(-1.0470165105894583)]
[np.float64(365.30933455983734), np.float64(0.0005184652821343461), np.float64(7.188678448267741), np.float64(0.11294840263355177)]
[np.float64(342.977472476599), np.float64(-3.702762768536889), np.float64(1.4608240859663004), np.float64(12.802155316363509)]
[np.float64(326.08470128269727), np.float64(-10.847315927812133), np.float64(56.944296483461834), np.float64(-3.999376661861407)]
```

Dec 22	☑ ~~Go through and double check all marked sarcasm type~~

		SCENE	SPEAKER	Sarcasm	TYPE	CHECK
36	37	1_11913_u	HOWARD	TRUE	1	
48	49	1_1549_u	HOWARD	TRUE	1	
69	72	1_2614_u	LEONARD	TRUE	1	
108	118	1_5156_u	HOWARD	TRUE	1	
117	128	1_5580_u	HOWARD	TRUE	1	
131	142	1_6188_u	PENNY	TRUE	1	
152	165	1_7402_u	LEONARD	TRUE	1	
193	211	1_S09E03_125_	LEONARD	TRUE	1	
221	240	1_S09E12_369_	PENNY	TRUE	1	
248	271	1_S10E02_101_	HOWARD	TRUE	1	
289	315	1_S10E09_258_	HOWARD	TRUE	1	
413	456	1_S12E01_332_	PENNY	TRUE	1	
481	535	1_S12E09_369_	PENNY	TRUE	1	
636	720	2_303_u	CHANDLER	TRUE	1	
673	762	2_376_u	CHANDLER	TRUE	1	
690	780	2_408_u	CHANDLER	TRUE	2	
709	804	2_438_u	CHANDLER	TRUE	1	
714	810	2_443_u	CHANDLER	TRUE	2	
733	833	2_478_u	CHANDLER	TRUE	1	
762	864	2_518_u	CHANDLER	TRUE	1	
840	952	2_92_u	CHANDLER	TRUE	1	
854	976	3_S02E01_050_	GILFOYLE	TRUE	1	
855	979	3_S02E02_081_	GILFOYLE	TRUE	3	
858	983	3_S02E03_561_	GILFOYLE	TRUE	1	
884						

Dec 23

- [x] ~~Cluster leg coef with k means clustering~~
  - [x] ~~By coef number~~

[332]:
```
one_coefs_pitch = plus[['prom_1_pitch']]
#all_coefs.head()
from sklearn.cluster import KMeans
k = KMeans(init='random', n_clusters=3, random_state=None)
k.fit(one_coefs_pitch)
print(k.cluster_centers_)
```

```
[[457.59959401]
 [284.30157812]
 [725.22608762]]
```

[333]:
```
two_coefs_pitch = plus[['prom_2_pitch']]
#all_coefs.head()
from sklearn.cluster import KMeans
k = KMeans(init='random', n_clusters=3, random_state=None)
k.fit(two_coefs_pitch)
print(k.cluster_centers_)
```

```
[[-65.74364768]
 [ -1.09436813]
 [ 61.08610545]]
```

[334]:
```
three_coefs_pitch = plus[['prom_3_pitch']]
#all_coefs.head()
from sklearn.cluster import KMeans
k = KMeans(init='random', n_clusters=3, random_state=None)
k.fit(three_coefs_pitch)
print(k.cluster_centers_)
```

```
[[311.85378127]
 [  7.42628528]
 [ 55.75154605]]
```

[335]:
```
four_coefs_pitch = plus[['prom_4_pitch']]
#all_coefs.head()
```

335]:
```
four_coefs_pitch = plus[['prom_4_pitch']]
#all_coefs.head()
from sklearn.cluster import KMeans
k = KMeans(init='random', n_clusters=3, random_state=None)
k.fit(four_coefs_pitch)
print(k.cluster_centers_)
```

```
[[ 21.95864104]
 [ -0.44252527]
 [-35.23252606]]
```

LEGENDRES INTENSITY

327]:
```
all_coefs_intensity = plus[['prom_1_intensity','prom_2_intensity','prom_3_intensity','prom_4_intensity']]
#print(all_coefs_intensity)
#print(all_coefs_intensity == None)
from sklearn.cluster import KMeans
k = KMeans(init='random', n_clusters=3, random_state=None)
k.fit(all_coefs_intensity)
print(k.cluster_centers_)
```

```
[[ 1.52330829e+02  2.68893129e-02  3.53414065e+00  2.56592378e-01]
 [ 1.31727629e+02 -3.28786448e-01  2.92713544e+00  2.10956119e-01]
 [ 1.15375060e+02  3.65091397e-01  1.86132599e+00  7.37939438e-02]]
```

336]:
```
one_coefs_intensity = plus[['prom_1_intensity']]
#print(all_coefs_intensity)
#print(all_coefs_intensity == None)
from sklearn.cluster import KMeans
k = KMeans(init='random', n_clusters=3, random_state=None)
k.fit(one_coefs_intensity)
print(k.cluster_centers_)
```

```
[[152.25627766]
 [131.74337916]
```

[337]:
```
two_coefs_intensity = plus[['prom_2_intensity']]
#print(all_coefs_intensity)
#print(all_coefs_intensity == None)
from sklearn.cluster import KMeans
k = KMeans(init='random', n_clusters=3, random_state=None)
k.fit(two_coefs_intensity)
print(k.cluster_centers_)
```

```
[[ 0.17950496]
 [-5.08764512]
 [ 5.68768408]]
```

[338]:
```
three_coefs_intensity = plus[['prom_3_intensity']]
#print(all_coefs_intensity)
#print(all_coefs_intensity == None)
from sklearn.cluster import KMeans
k = KMeans(init='random', n_clusters=3, random_state=None)
k.fit(three_coefs_intensity)
print(k.cluster_centers_)
```

```
[[ 0.60917093]
 [31.71410611]
 [ 6.81425315]]
```

[339]:
```
four_coefs_intensity = plus[['prom_4_intensity']]
#print(all_coefs_intensity)
#print(all_coefs_intensity == None)
from sklearn.cluster import KMeans
k = KMeans(init='random', n_clusters=3, random_state=None)
k.fit(four_coefs_intensity)
print(k.cluster_centers_)
```

```
[[ 0.15040942]
```

☑ ~~All four coef~~

1]:
```
all_coefs_pitch = plus[['prom_1_pitch','prom_2_pitch','prom_3_pitch','prom_4_pitch']]
#all_coefs.head()
from sklearn.cluster import KMeans
k = KMeans(init='random', n_clusters=3, random_state=None)
k.fit(all_coefs_pitch)
```

1]:
```
▸ KMeans
```

7]:
```
all_coefs_intensity = plus[['prom_1_intensity','prom_2_intensity','prom_3_intensity','prom_4_intensity']]
#print(all_coefs_intensity)
#print(all_coefs_intensity == None)
from sklearn.cluster import KMeans
k = KMeans(init='random', n_clusters=3, random_state=None)
k.fit(all_coefs_intensity)
print(k.cluster_centers_)
```

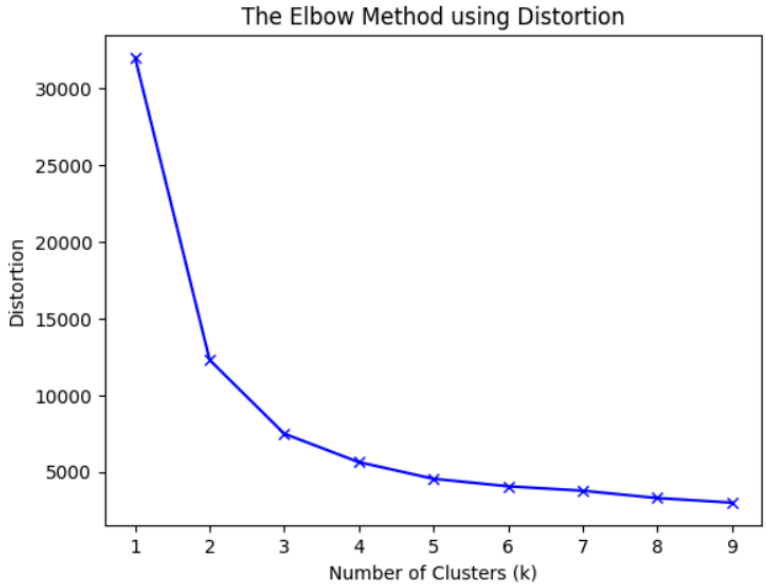
Dec 25

- [x] ~~Document removed utterances because prom word too short~~

	index	level_0	SCENE	SPEAKER	SHOW
0	268	188	1_S09E02_424_u	SHELDON	BBT
1	495	379	1_S11E11_077_u	HOWARD	BBT
2	685	540	2_13_u	RACHEL	FRIENDS
3	706	555	2_163_u	RACHEL	FRIENDS
4	731	570	2_198_u	ROSS	FRIENDS

Prominent word too short to extract contours	1	4

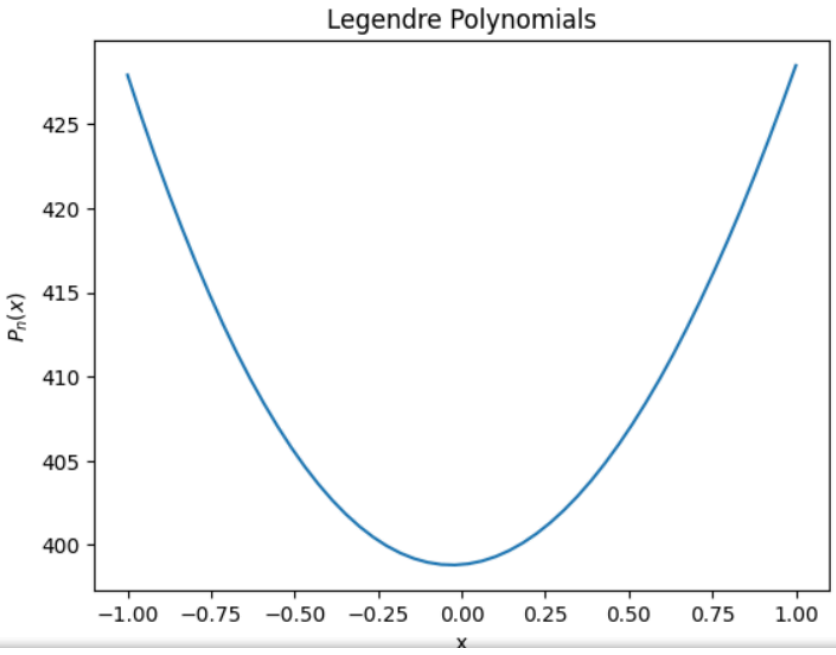
- [x] ~~Find optimal number of groups by finding minimizing sum of square distrances (distortion)~~

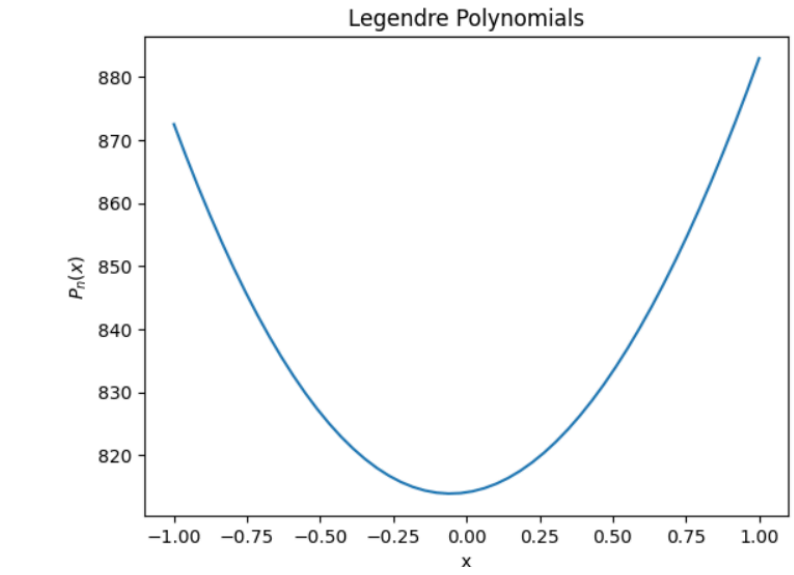


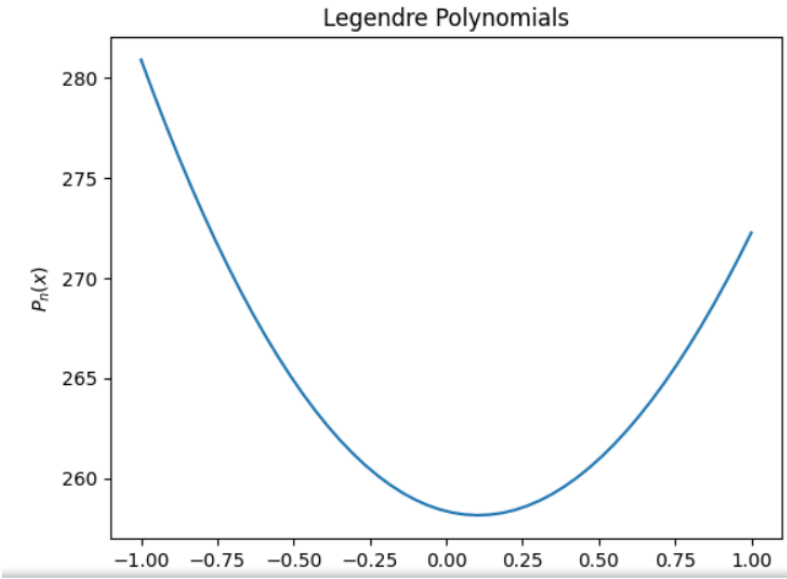
- [x] ~~Graph cluster centroids~~

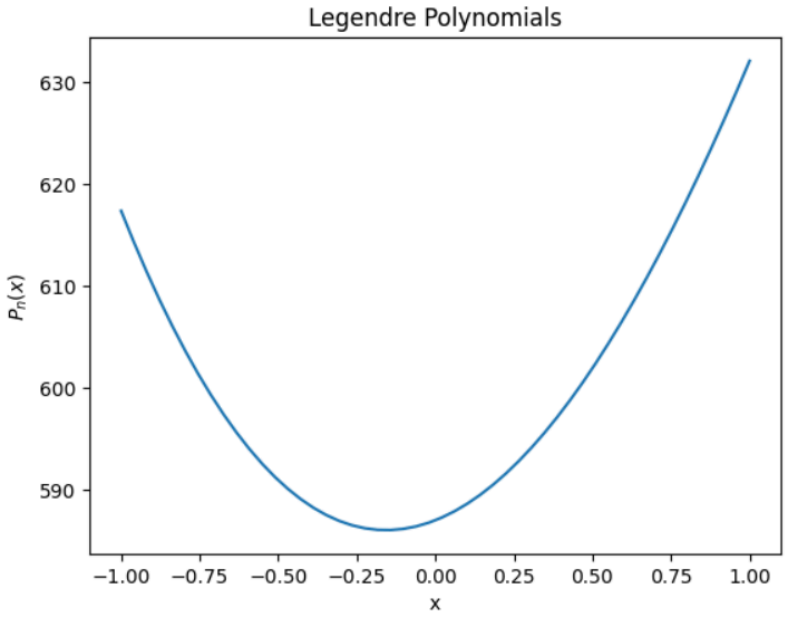
```
from sklearn.cluster import KMeans
k = KMeans(init='random', n_clusters=4, random_state=1)
k.fit(all_coefs_pitch)
#print(center)
center = k.cluster_centers_
#print(center)
print(k.cluster_centers_)
#all_coefs_pitch[0]
import numpy as np
from scipy.special import eval_legendre
X = np.linspace(-1, 1)
for n in range(0,4):
    fff = dict(enumerate(center[n]))
    print(fff)
    y=0
    for m in range(0, 4):
        print(fff[m])
        poly = eval_legendre(m, X) * fff[m]
        y += poly
    plt.plot(X, y)
    plt.title("Legendre Polynomials")
    plt.xlabel("x")
    plt.ylabel(r'$P_n(x)$')
    plt.show()
    n+=1
```

```
-0.5686525431981557
```




Legendre Polynomials
880
870
860
850
840
830
820
$P_n(x)$
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x


Legendre Polynomials
280
275
270
265
260
$P_n(x)$
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00



5]:

```
center = k.cluster_centers_
#print(center)
print(k.cluster_centers_)
#all_coefs_pitch[0]
import numpy as np
from scipy.special import eval_legendre
X = np.linspace(-1, 1)
for n in range(0,3):
    fff = dict(enumerate(center[n]))
    print(fff)
    y=0
    for m in range(0, 4):
        print(fff[m])
        poly = eval_legendre(m, X) * fff[m]
        y += poly
    plt.plot(X, y)
    plt.title("Legendre Polynomials")
    plt.xlabel("x")
    plt.ylabel(r'$P_n(x)$')
    plt.show()
    n+=1
```

Dec 28

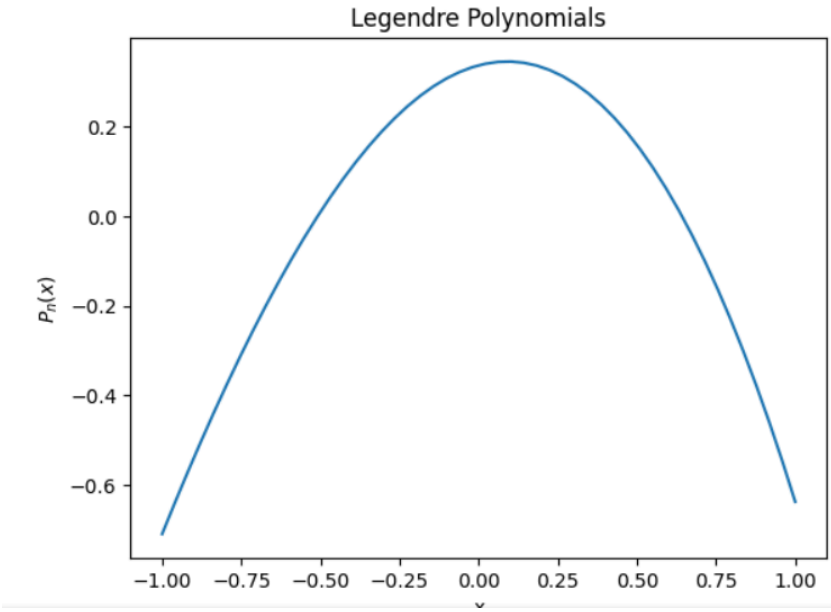
- ☑ ~~Normalize pitch contour before generating contour graph~~
  - ☑ ~~Just try not using the mean and slope when making clusters~~

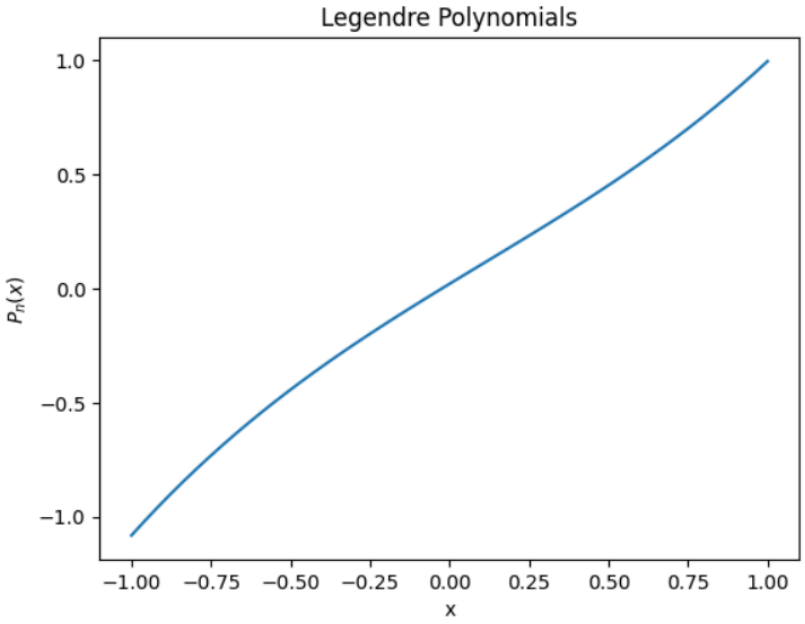
```
]: from sklearn.cluster import KMeans
k = KMeans(init='random', n_clusters=4, random_state=1)
k.fit(all_coefs_pitch)
#print(center)
center = k.cluster_centers_
#print(center)
print(k.cluster_centers_)
center = [[1, 1] + row.tolist() for row in center ]

print(k.cluster_centers_)
#all_coefs_pitch[0]
import numpy as np
from scipy.special import eval_legendre
X = np.linspace(-1, 1)
for n in range(0,4):
    fff = dict(enumerate(center[n]))
    print(fff)
    y=0
    for m in range(0, 4):
        print(fff[m])
        poly = eval_legendre(m, X) * fff[m]
        #if m > 1:
        y += poly
    plt.plot(X, y)
    plt.title("Legendre Polynomials")
    plt.xlabel("x")
    plt.ylabel(r'$P_n(x)$')
    plt.show()
    n+=1
```

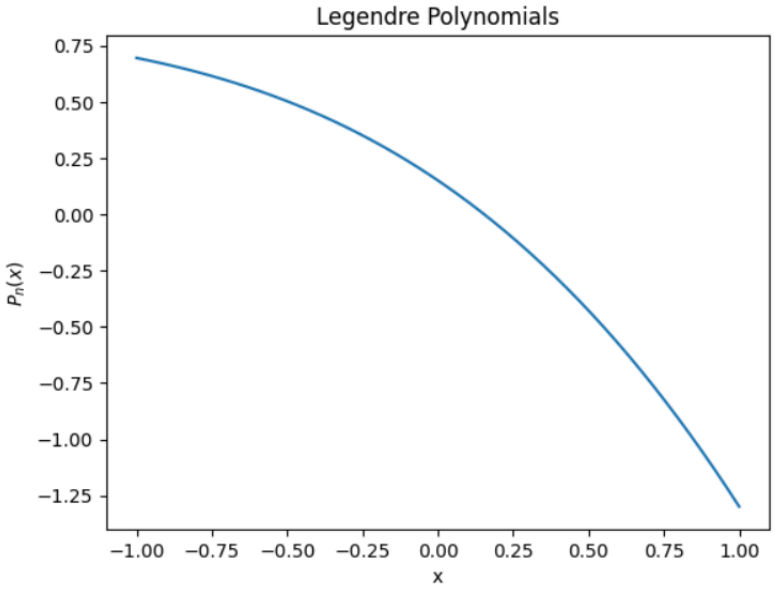
☑ ~~Normalize full contour by z score~~

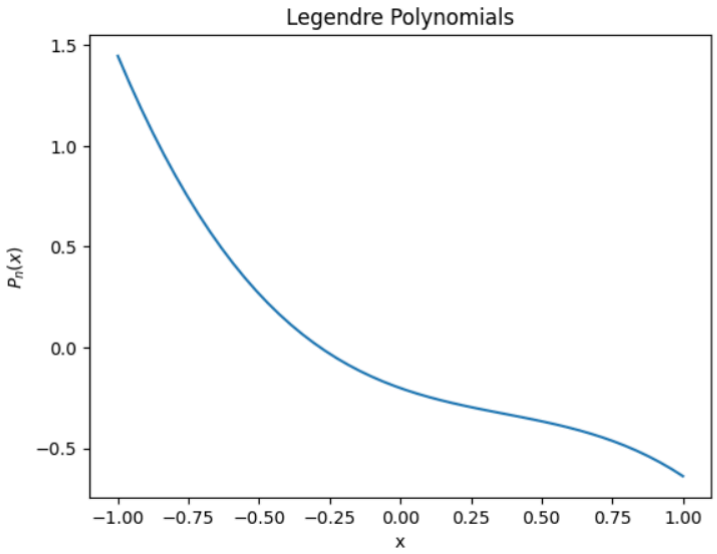
```
prom_contour_pitch, prom_contour_intensity = get_contours (sound,
#print(prom_contour_pitch.dtype())
prom_contour_pitch = scipy.stats.zscore(prom_contour_pitch)
print(prom_contour_pitch)
```

```
        prom_contour_pitch = scipy.stats.zscore(prom_contour_pitch)
        prom_contour_intensity = scipy.stats.zscore(prom_contour_intensity)
        #print(prom_contour_pitch)
        prom_leg_pitch = get_legendres(prom_contour_pitch, n=4)
```

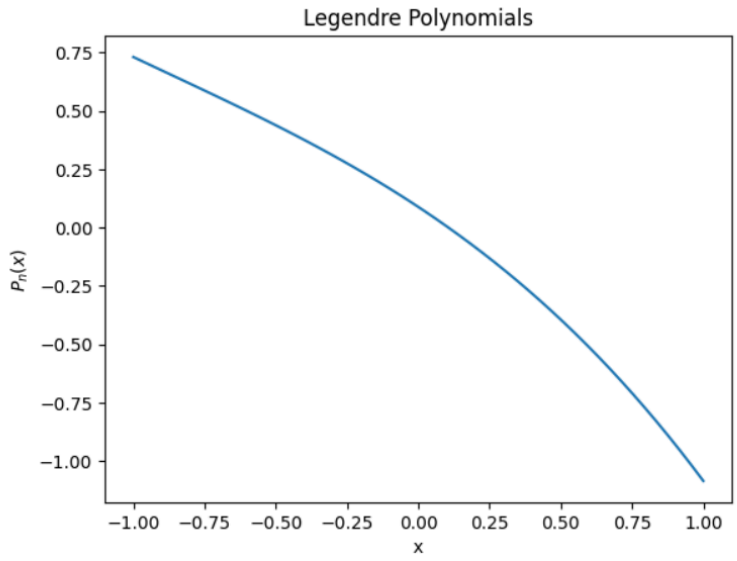

Legendre Polynomials
0.2
0.0
−0.2
−0.4
−0.6
$P_n(x)$
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

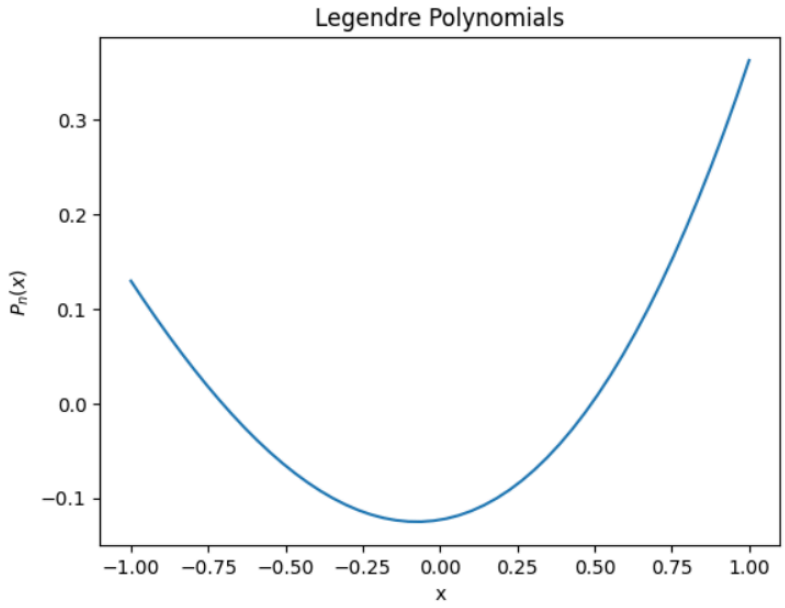

Legendre Polynomials
1.0
0.5
0.0
−0.5
−1.0
$P_n(x)$
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x

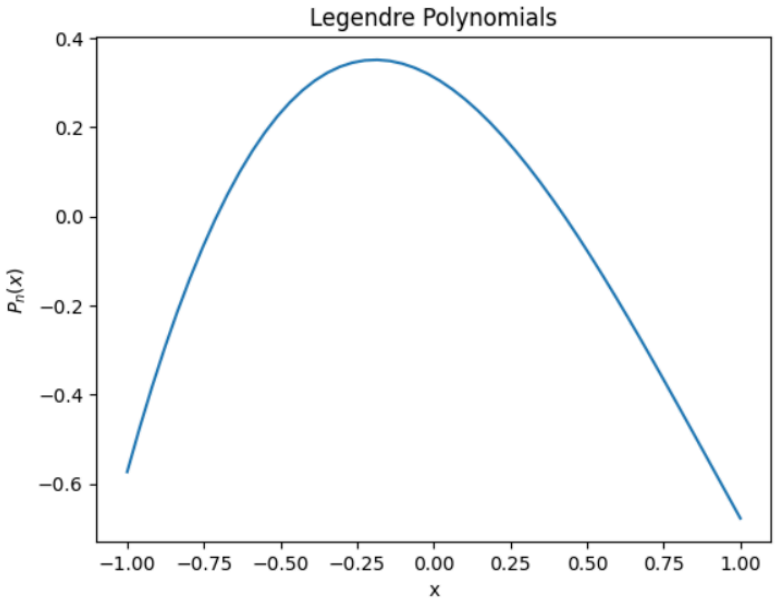
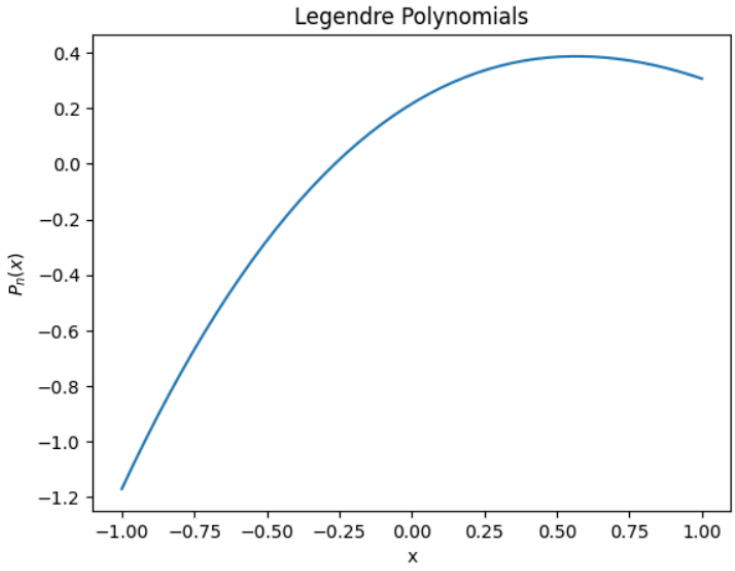

Legendre Polynomials
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
$P_n(x)$
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
x


Legendre Polynomials
1.5
1.0
0.5
0.0
−0.5
$P_n(x)$
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
x

Intensity:


Legendre Polynomials
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
$P_n(x)$
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
x


Legendre Polynomials
0.3
0.2
0.1
0.0
−0.1
$P_n(x)$
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
x

Dec 29	☑ ~~Label utterances with current 4 cluster pitch and intensity classes~~
	`k = KMeans(init='random', n_clusters=4, random_state=1)`
	`plus['cluster_allpitch'] = k.fit_predict(all_coefs_pitch)`
	`#print(center)`

	cluster_allpitch 1 3 3 0 ☑ ~~Make list of all baseline features and confirm presence~~

Sentence level features:

- Mean pitch
- Pitch range
- Pitch SD
- Mean intensity
- Intensity range
- Intensity SD
- Speaker rate in syl/sec

☑ ~~Make list of all new local features and confirm presence~~

Word level features

- Speech rate of prominent word
- Prominent word pitch contour type
- Prominent word intensity contour type
- Difference between mean syl/sec of non prominent words in utterance and rate of prominent word
- Pitch mean, range, SD before and after prom word
- Intensity mean, range, SD before and after prom word

FIND HERE: list of features

☑ ~~Run basic log regression model on baseline features~~

```
import itertools
import numpy as np
counts = plus["SPEAKER"].value_counts()
target = 174.4
speakers = counts.index.tolist()
best_combo = None
best_diff = float('inf')

for r in range(1, len(speakers) + 1):
    for combo in itertools.combinations(speakers, r):
        total = counts[list(combo)].sum()
        diff = abs(target - total)
        if diff < best_diff:
            best_diff = diff
            best_combo = combo

best_combo, counts[list(best_combo)].sum()
```

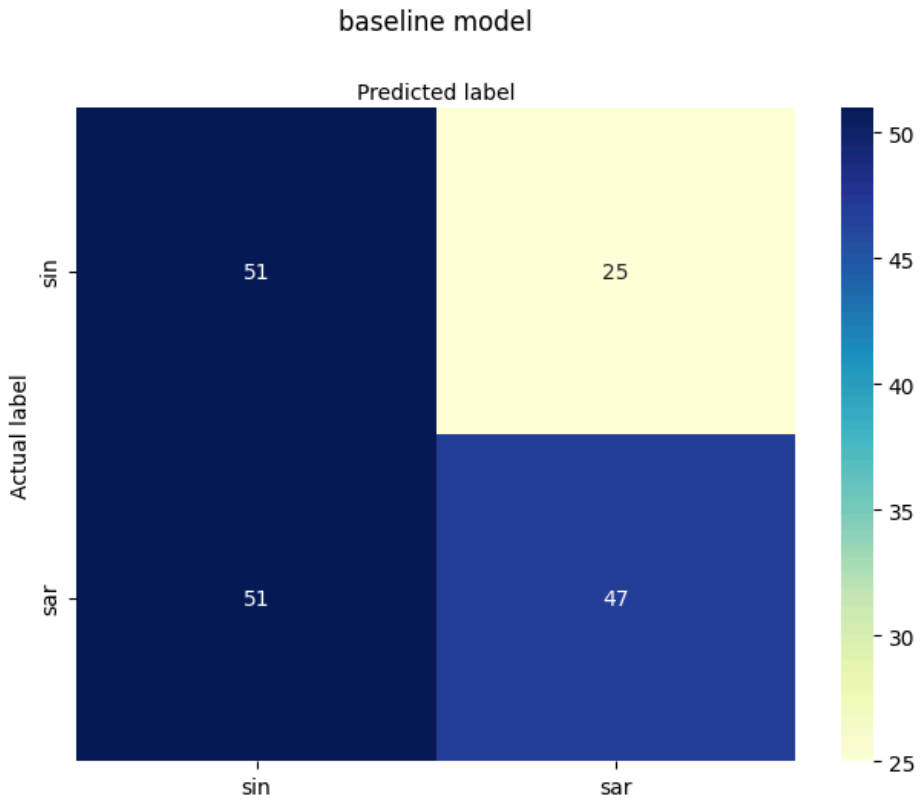
[300]:
```
len(plus[plus["SPEAKER"].isin(['RACHEL', 'CHANDLER','MONICA'])])
```

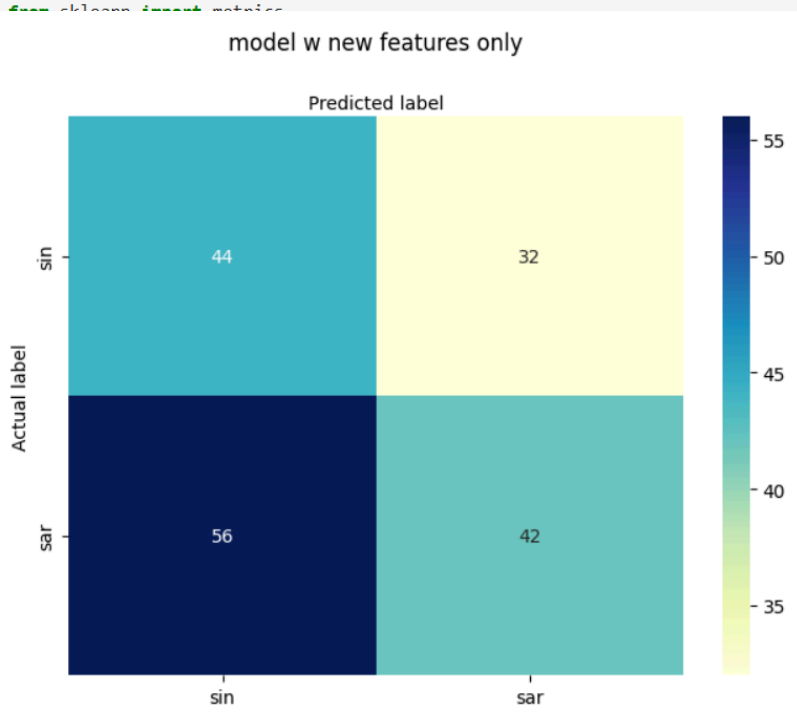
[300]: 174

```
12]: traiin = plus[~plus["SPEAKER"].isin(['RACHEL', 'CHANDLER','MONICA'])]
X_train = traiin[['st_mean', 'st_range', 'st_sd', 'mean_db_fix','range_db_fix','sd_db_fix','slypersec']]
y_train = traiin.Sarcasm

testy = plus[plus["SPEAKER"].isin(['RACHEL', 'CHANDLER','MONICA'])]
X_test = testy[['st_mean', 'st_range', 'st_sd', 'mean_db_fix','range_db_fix','sd_db_fix','slypersec']]
y_test = testy.Sarcasm
```

```
316]: from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import StratifiedKFold, cross_val_predict
from sklearn.metrics import classification_report

logreg = LogisticRegression(max_iter=1000, random_state=16)
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)

names = ['sin', 'sar']
print(classification_report(y_test, y_pred, target_names=names))
```

```
              precision    recall  f1-score   support

         sin       0.50      0.67      0.57        76
         sar       0.65      0.48      0.55        98

    accuracy                           0.56       174
   macro avg       0.58      0.58      0.56       174
weighted avg       0.59      0.56      0.56       174
```

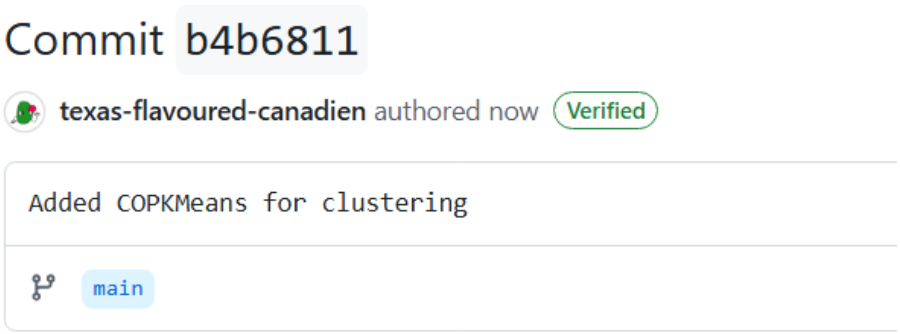


Dec 30	☑ ~~Create model with new features only~~ ``` traain = plus[~plus["SPEAKER"].isin(['RACHEL', 'CHANDLER','MONICA'])] X_train = traain[['prom_slypersec','cluster_allpitch','cluster_allintensity','prom_2_pitch','prom_3_pitch','prom_4_pitch','prom_2_intensity', 'prom_3_intensity','prom_4_intensity']] y_train = traain.Sarcasm testy = plus[plus["SPEAKER"].isin(['RACHEL', 'CHANDLER','MONICA'])] X_test = testy[['prom_slypersec','cluster_allpitch','cluster_allintensity','prom_2_pitch','prom_3_pitch','prom_4_pitch','prom_2_intensity', 'prom_3_intensity','prom_4_intensity']] y_test = testy.Sarcasm ``` ``` from sklearn.linear_model import LogisticRegression from sklearn.model_selection import StratifiedKFold, cross_val_predict from sklearn.metrics import classification_report logreg = LogisticRegression(max_iter=1000, random_state=16) logreg.fit(X_train, y_train) y_pred = logreg.predict(X_test) names = ['sin', 'sar'] print(classification_report(y_test, y_pred, target_names=names)) ``` |	precision	recall	f1-score	support
---	---	---	---	---
sin	0.44	0.58	0.50	76
sar	0.57	0.43	0.49	98
accuracy			0.49	174
macro avg	0.50	0.50	0.49	174
weighted avg	0.51	0.49	0.49	174	 
Jan 1	☐ Add remaining local features ☐ Pitch and intensity features before and after the prom word

	☑ ~~Speech rate of non prominent words and difference between speech rates~~
	10]:
	plus['nonprom_slypersec'] = None
	for idx, key in plus['SCENE'].items():
	try:
	path = None
	for root, dirs, files in os.walk(base_dir):
	target_file = f"{key}.TextGrid"
	if target_file in files:
	path = os.path.join(root, target_file)
	break
	if path is None:
	raise FileNotFoundError(f"srry bb I can't find {key} idk...")
	###find time interval of all words in textgrid
	word_intervals = get_times_from_textgrid(path, "words")
	prom_intervals = get_times_from_textgrid(path, "p")
	prom_timepoint = prom_intervals[0][1]
	print(word_intervals)
	#total_dur = plus.loc[idx, 'duration']
	#total_syl = plus.loc[idx, "syl_count"]
	#print(total_syl)
	nonprom_duration = plus.loc[idx, 'duration'] - plus.loc[idx, 'prom_duration']
	nonprom_syl_count = plus.loc[idx, "syl_count"] - plus.loc[idx, "prom_syl_count"]
	print(nonprom_syl_count)
	print(nonprom_duration)
	print(nonprom_syl_count/nonprom_duration)
	plus.loc[idx, 'nonprom_slypersec'] = nonprom_syl_count/nonprom_duration
	except Exception as e:
	print(f"oh noses u have a problem in {path}: {e}")
	]:
	#negative = prom word is faster than rest
	# positive = prom word is slower than rest
	plus['speech_rate_diff'] = plus['nonprom_slypersec']-plus['prom_slypersec']

Jan 2	☑ ~~Run COP-Kmeans for clustering~~
	Microsoft Windows [Version 10.0.26100.7462]
	(c) Microsoft Corporation. All rights reserved.
	
	C:\Users\kasia>cd
	
	C:\Users\kasia>cd C:\Users\kasia\Downloads\COP-Kmeans-master\COP-Kmeans-master
	
	C:\Users\kasia\Downloads\COP-Kmeans-master\COP-Kmeans-master>py -m pip install .
	Processing c:\users\kasia\downloads\cop-kmeans-master\cop-kmeans-master
	Installing build dependencies ... done
	Getting requirements to build wheel ... done
	Preparing metadata (pyproject.toml) ... done
	Building wheels for collected packages: copkmeans
	Building wheel for copkmeans (pyproject.toml) ... done
	Created wheel for copkmeans: filename=copkmeans-1.5-py3-none-any.whl size=5088 sha256=c738ad4a4c4681244fc0226944b89f96cb9fd76bfee028aedd3be0838106dd45
	Stored in directory: c:\users\kasia\appdata\local\pip\cache\wheels\7c\65\e5\4e1c38ac80be5d50b8bb1359e124f06137ee9967154d371f6f
	Successfully built copkmeans
	Installing collected packages: copkmeans
	Attempting uninstall: copkmeans
	Found existing installation: copkmeans 1.5
	Uninstalling copkmeans-1.5:
	Successfully uninstalled copkmeans-1.5
	Successfully installed copkmeans-1.5
	
	C:\Users\kasia\Downloads\COP-Kmeans-master\COP-Kmeans-master>
	[400]:
	import numpy as np
	from itertools import combinations
	# Extract feature matrix and labels
	data = plus[['prom_1_pitch', 'prom_2_pitch','prom_3_pitch','prom_4_pitch']].to_numpy()
	y = plus["Sarcasm"].to_numpy()
	# Build cannot-link constraints: all pairs where labels differ
	cannot_link = [(i, j) for i, j in combinations(range(len(y)), 2) if y[i] != y[j]]
	[481]:
	from copkmeans.cop_kmeans import cop_kmeans
	clusters, centers = cop_kmeans(dataset = data, k = 8, cl=cannot_link)
	[482]:
	print(centers)
	[[np.float64(-1.3486644403183013e-16), np.float64(0.5063934020223847), np.float64(-0.6349107246939554), np.float64(-0.023121391495546453)], [np.float64(3.5531028370722196-16), np.float64(0.9319794458677341), np.float64(-0.23014054630763703), np.float64(0.05353271053339028)], [np.float64(-2.82590463430 28324e-16), np.float64(0.9765511333656032), np.float64(0.08230180858183152), np.float64(0.05150339367455194)], [np.float64(-3.073928956755694e-16), np.float64(-0.5526540853394917), np.float64(-0.5590308758896814), np.float64(0.026812672496961426)], [np.float64(-1.0885382946766502e-16), np.float64(-1.0353396095774203), np.float64(0.10418211047585442), np.float64(-0.190215291396512)], [np.float64(3.6003759747722629e-16), np.float64(-0.49144850696372 42), np.float64(-0.6110401549707288), np.float64(0.011267814305702731)], [np.float64(9.606831417853756-17), np.float64(-1.0064761473302084), np.float64(0.12651452735587151), np.float64(-0.16482962599394757)], [np.float64(-8.245060031984808e-18), np.float64(0.17481574907967276), np.float64(0.4953352716 874816), np.float64(0.025228143378059203)]]
	[483]:
	#cluster 0 = sin
	#cluster 1 = sar
	#cluster 2 = sar
	#cluster 3 = sin
	print(len(clusters))
	print(clusters)
	plus['COP_clusters_pitch'] = clusters

	[496]:
```
import numpy as np
from itertools import combinations

# Extract feature matrix and labels
data = plus[['prom_1_intensity','prom_2_intensity', 'prom_3_intensity','prom_4_intensity']].to_numpy()
y = plus["Sarcasm"].to_numpy()

# Build cannot-link constraints: all pairs where labels differ
cannot_link = [(i, j) for i, j in combinations(range(len(y)), 2) if y[i] != y[j]]
```
[497]:
```
from copkmeans.cop_kmeans import cop_kmeans
clusters, centers = cop_kmeans(dataset = data, k = 7, cl=cannot_link)
```
[498]:
```
print(centers)
print(len(clusters))
print(clusters)
plus['COP_clusters_intensity'] = clusters
```
☑ ~~Commit code to github~~


 |
| January 🎉 | ☑ ~~Report final model performance~~ |
| Jan 5 | ☑ ~~Add if the utterance is a question as training data~~
```
traiin = plus[~plus["SPEAKER"].isin(['RACHEL', 'CHANDLER','MONICA'])]
X_train = traiin[['prom_slypersec','cluster_allpitch','cluster_allintensity','prom_2_pitch','prom_3_pitch','prom_4_pitch','prom_2_intensity',
                  'prom_3_intensity','prom_4_intensity', 'speech_rate_diff', 'COP_clusters_pitch','QUESTION?']]

y_train = traiin.Sarcasm

testy = plus[plus["SPEAKER"].isin(['RACHEL', 'CHANDLER','MONICA'])]
X_test = testy[['prom_slypersec','cluster_allpitch','cluster_allintensity','prom_2_pitch','prom_3_pitch','prom_4_pitch','prom_2_intensity',
                'prom_3_intensity','prom_4_intensity', 'speech_rate_diff','COP_clusters_pitch','QUESTION?']]
y_test = testy.Sarcasm
```
☑ ~~Send email~~

LARGE UPDATE EMAIL:

Hi Brooklyn and Dr Anthanasopoulou,

Happy new year! I have lots of updates:

<u>Features:</u>

I extracted the first 4 legendre coefficients for pitch and intensity for prominent words. With Kmeans, I clustered both into an arbitrary 4 groups each (cluster the pitch contours together and the intensity |

contours together). The pitch clusters were at first mostly based on pitch (clustered mostly based on the first coefficient) so then I normalized the pitch contours for each utterance to make the first coefficient equal to zero. I made sure to normalize the pitch contour after getting the mean pitch in semitones so that it's still useful.

I also tried COP-Kmeans to only generate clusters that don't mix the coefs of sarcastic with sincere utterances (again with an arbitrary 4 clusters). Using Kmeans but clustering sarcastic and sincere utterances separately would probably do the same thing, but I used COP-Kmeans just in case the optimal number of clusters is odd.

For both methods, I labeled utterances with the cluster number the contour of the prominent word it is closest to and used that as training data.

I extracted the 7 global features that all the other papers used, pitch mean, SD and range measured in semitones normalized by speaker mean pitch and intensity mean, SD, and range in db normalized by speaker mean intensity. I also extracted utterance speech rate (not normalized yet).

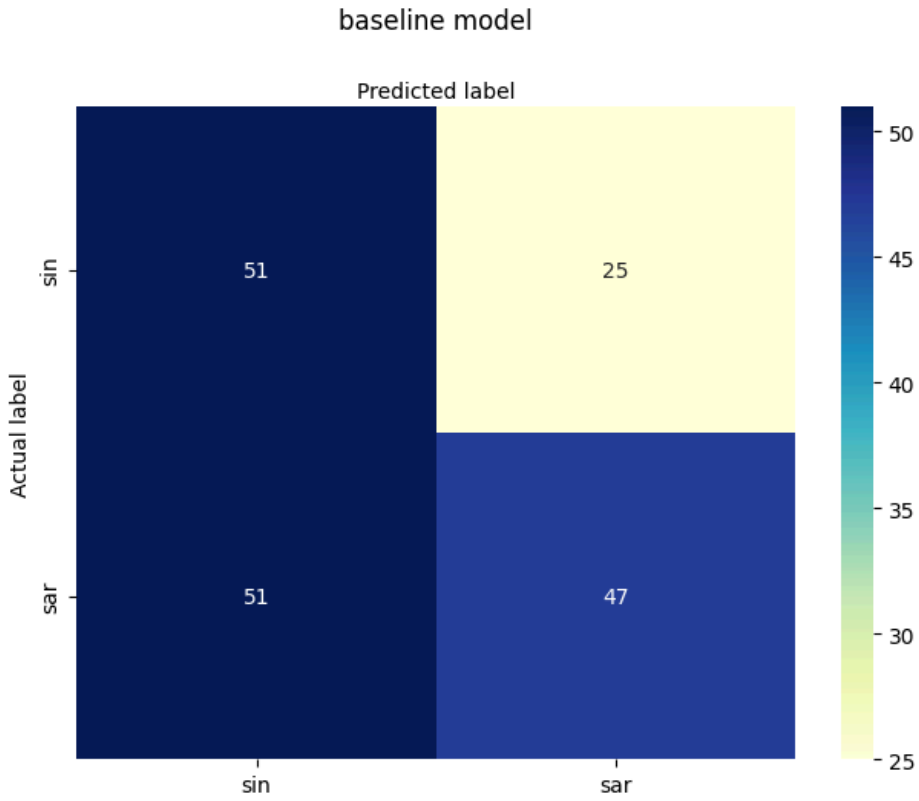
I removed 5 speakers because the prominent word was too short to get pitch and intensity contours, bringing the dataset to 872 utterances (439 sarcastic and 433 sincere).

Model

I trained a logistic regression model on just global features with default settings. I used a 20/80 split. To keep testing speaker independent, I found the exact size the testing set should be, (174.4 utterances) and then found the combination of speakers to exclude from the training set that would produce the utterance amount closest to 174.4. These were Rachel's, Chandler's, and Monica's utterances (174 total). There would be a slight data imbalance (76 sarcastic vs 98 sincere), but I don't think its big enough to be concerning.

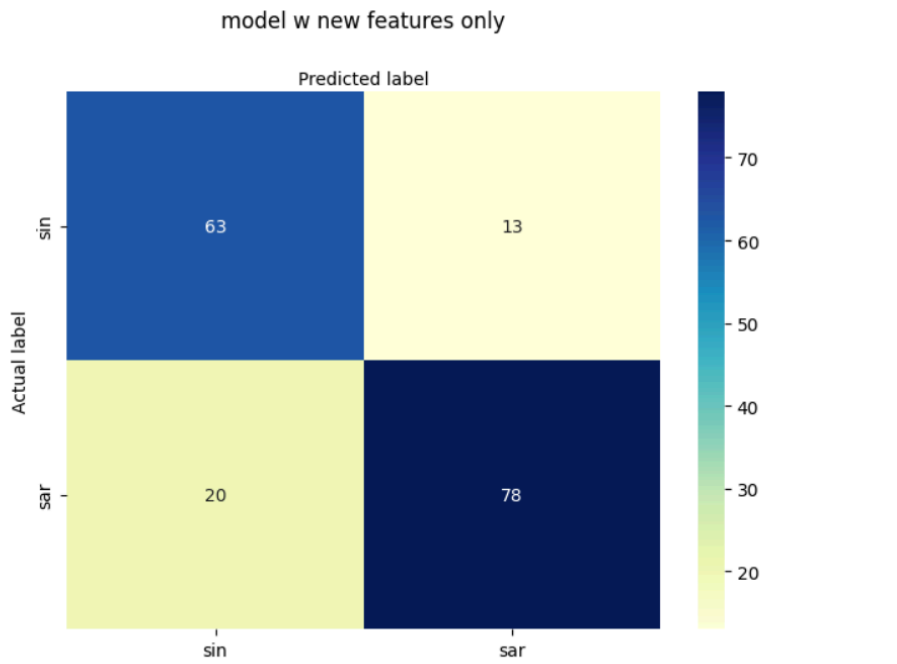
The performance of the baseline model is very similar to Rakov as expected.

	precision	recall	f1-score	support
sin	0.50	0.67	0.57	76
sar	0.65	0.48	0.55	98
accuracy			0.56	174
macro avg	0.58	0.58	0.56	174
weighted avg	0.59	0.56	0.56	174



For the model with new features, I included the legendre coefs for both pitch and intensity, the cluster number generated with Kmeans for pitch and intensity, the speech rate of the prominent word and the difference between the speech rate of the prominent word and non prominent words in the utterance. When I used the cluster number generated with COP-Kmeans (for pitch only) instead, I got a significantly improved performance.

	precision	recall	f1-score	support
sin	0.76	0.83	0.79	76
sar	0.86	0.80	0.83	98
accuracy			0.81	174
macro avg	0.81	0.81	0.81	174
weighted avg	0.81	0.81	0.81	174

	 The model got worse when I added the cluster number generated with COP-Kmeans for intensity, so I suppose next steps for this is to find the optimal combination of the number of pitch and intensity clusters. Rakov used bigrams and trigrams of clusters, but I'm not sure how that would fit into my project. Instead, maybe I should focus on getting the pitch and intensity contours before and after the utterance and comparing the two (like a semi global feature?) I'll also create a document to start depositing my graphs and analysis. Are we meeting online this Thursday? Andi
Jan 6	Email Hi all, I just realized that I have a meeting on Thursday 😅 Would rescheduling to Monday Jan 12 work? Andi
Jan 8	☐ Find optimal combo of # of pitch and intensity clusters
Jan 12	<u>Notes:</u> • Print what features are significant • Statsmodels library • Watch vid on how log reg works

	• Analysis of the 4 diff cats • Write preprocessing section for the methodology • Next meetings at 10-11am • Leg coefs based on normalized pitch contour
Jan 19	☑ ~~Install statsmodels~~

```
C:\Users\kasia>py -m pip install statsmodels
Requirement already satisfied: statsmodels in c:\users\kasia\appdata\local\programs\python\python313\lib\site-packages (
0.14.6)
Requirement already satisfied: numpy<3,>=1.22.3 in c:\users\kasia\appdata\local\programs\python\python313\lib\site-packa
ges (from statsmodels) (2.3.3)
Requirement already satisfied: scipy!=1.9.2,>=1.8 in c:\users\kasia\appdata\local\programs\python\python313\lib\site-pac
kages (from statsmodels) (1.15.3)
Requirement already satisfied: pandas!=2.1.0,>=1.4 in c:\users\kasia\appdata\local\programs\python\python313\lib\site-pa
ckages (from statsmodels) (2.3.2)
Requirement already satisfied: patsy>=0.5.6 in c:\users\kasia\appdata\local\programs\python\python313\lib\site-packages
(from statsmodels) (1.0.2)
Requirement already satisfied: packaging>=21.3 in c:\users\kasia\appdata\local\programs\python\python313\lib\site-packag
es (from statsmodels) (25.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\kasia\appdata\local\programs\python\python313\lib\site
-packages (from pandas!=2.1.0,>=1.4->statsmodels) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\kasia\appdata\local\programs\python\python313\lib\site-packages
(from pandas!=2.1.0,>=1.4->statsmodels) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\kasia\appdata\local\programs\python\python313\lib\site-package
s (from pandas!=2.1.0,>=1.4->statsmodels) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\kasia\appdata\local\programs\python\python313\lib\site-packages (fro
m python-dateutil>=2.8.2->pandas!=2.1.0,>=1.4->statsmodels) (1.17.0)

C:\Users\kasia>
```

☑ ~~Print results of log reg for both models~~

BASELINE

```
[123]: plus['st_mean'] = plus['st_mean'].astype(float)
       plus['st_range'] = plus['st_range'].astype(float)
       plus['st_sd'] = plus['st_sd'].astype(float)
       plus['mean_db_fix'] = plus['mean_db_fix'].astype(float)
       plus['range_db_fix'] = plus['range_db_fix'].astype(float)
       plus['sd_db_fix'] = plus['sd_db_fix'].astype(float)
       plus['slypersec'] = plus['slypersec'].astype(float)

*[124]: import statsmodels.api as sm
        X_train = plus[['st_mean', 'st_range', 'st_sd', 'mean_db_fix','range_db_fix','sd_db_fix','slypersec']]
        y_train = plus.Sarcasm

[125]: log_reg = sm.Logit(y_train, X_train).fit()

[126]: print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                Sarcasm   No. Observations:                  872
Model:                          Logit   Df Residuals:                      865
Method:                           MLE   Df Model:                            6
Date:                Tue, 20 Jan 2026   Pseudo R-squ.:                0.02712
Time:                        08:40:18   Log-Likelihood:                -588.01
converged:                       True   LL-Null:                       -604.40
Covariance Type:            nonrobust   LLR p-value:                 1.156e-05
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
st_mean        -0.0600      0.019     -3.242      0.001      -0.096      -0.024
st_range        0.0198      0.017      1.191      0.234      -0.013       0.052
st_sd          -0.0968      0.069     -1.397      0.162      -0.233       0.039
mean_db_fix     0.3776      0.144      2.630      0.009       0.096       0.659
range_db_fix    0.5411      0.168      3.213      0.001       0.211       0.871
sd_db_fix      -1.7082      0.635     -2.688      0.007      -2.954      -0.463
slypersec      -0.0822      0.040     -2.073      0.038      -0.160      -0.004
```

NEW FEATURES

Logit Regression Results

Dep. Variable:	Sarcasm	No. Observations:	872
Model:	Logit	Df Residuals:	852
Method:	MLE	Df Model:	19
Date:	Tue, 20 Jan 2026	Pseudo R-squ.:	0.04453
Time:	22:34:11	Log-Likelihood:	-577.49
converged:	True	LL-Null:	-604.40
Covariance Type:	nonrobust	LLR p-value:	3.506e-05

	coef	std err	z	P>|z|	[0.025	0.975]
prom_slypersec	-0.1298	0.074	-1.766	0.077	-0.274	0.014
prom_1_pitch	0.0263	0.011	2.308	0.021	0.004	0.049
prom_2_pitch	-0.0048	0.026	-0.189	0.850	-0.055	0.045
prom_3_pitch	0.0188	0.050	0.379	0.705	-0.079	0.116
prom_4_pitch	0.0851	0.090	0.941	0.347	-0.092	0.262
prom_1_intensity	-0.3643	0.180	-2.021	0.043	-0.718	-0.011
prom_2_intensity	-0.0579	0.179	-0.324	0.746	-0.408	0.292
prom_3_intensity	0.3318	0.325	1.021	0.307	-0.305	0.969
prom_4_intensity	-0.0454	0.460	-0.099	0.921	-0.948	0.857
speech_rate_diff	-0.0765	0.057	-1.338	0.181	-0.189	0.036
QUESTION?	0.0494	0.165	0.300	0.764	-0.273	0.372
st_mean	-0.1064	0.028	-3.811	0.000	-0.161	-0.052
st_range	0.0122	0.017	0.715	0.475	-0.021	0.046
st_sd	-0.0232	0.073	-0.318	0.750	-0.166	0.120
mean_db_fix	0.9122	0.254	3.593	0.000	0.415	1.410
range_db_fix	0.5885	0.177	3.332	0.001	0.242	0.935
sd_db_fix	-0.9687	0.687	-1.409	0.159	-2.316	0.378
slypersec	0.1967	0.124	1.592	0.111	-0.046	0.439
cluster_allpitch	-0.1760	0.070	-2.515	0.012	-0.313	-0.039
cluster_allintensity	-0.1622	0.184	-0.883	0.377	-0.522	0.198

Jan 20

- [x] ~~Normalize speech rate by speaker~~

```
speaker_meanspeechrate = plus.groupby("SPEAKER")["slypersec"].mean().to_dict()
print(speaker_meanspeechrate)
```

```
{'AMY': 5.2337268938067405, 'BERNADETTE': 4.923037996287565, 'CHANDLER': 4.859418179747528, 'DOROTHY': 4.072422154506618, 'GILFOYLE': 4.9875
371, 'HOWARD': 4.872607721606251, 'JOEY': 4.20745144668958, 'LEONARD': 5.074342008098685, 'MONICA': 4.914897998433404, 'PENNY': 5.189271728
'PHOEBE': 4.457394059596489, 'RACHEL': 4.558430036197937, 'ROSS': 4.462241614891628, 'SHELDON': 5.023658632499979}
```

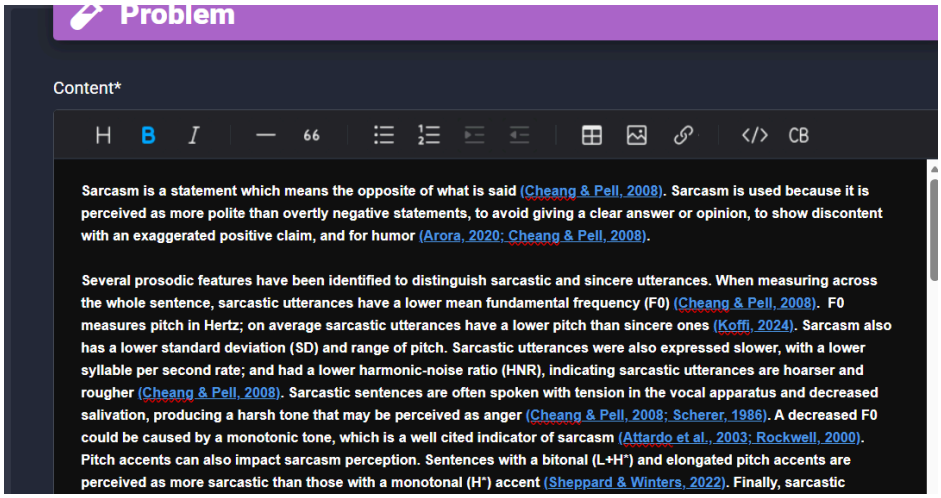
```
speaker_sdspeechrate = plus.groupby("SPEAKER")["slypersec"].std().to_dict()
print(speaker_sdspeechrate)
```

```
{'AMY': 1.4345071690986677, 'BERNADETTE': 1.2218360409615798, 'CHANDLER': 1.3563880726399393, 'DOROTHY': 0.8247510092217228, 'GILFOYLE': 1.6
9958275, 'HOWARD': 1.2171968513472586, 'JOEY': 1.422810155529511, 'LEONARD': 1.276292219763137, 'MONICA': 2.1067689383520607, 'PENNY': 1.42
26833, 'PHOEBE': 1.4313507512315582, 'RACHEL': 1.8573575434927285, 'ROSS': 1.693260499365582, 'SHELDON': 1.2301026021056325}
```

```
for idx, key in plus['SCENE'].items():
    try:
        speaker = plus.loc[idx, 'SPEAKER']
        unnorm_speechrate = plus.loc[idx, 'slypersec']
        norm_speechrate = (unnorm_speechrate - speaker_meanspeechrate[speaker])/speaker_sdspeechrate[speaker]
        plus.loc[idx, 'slypersec'] = norm_speechrate

    except Exception as e:
        print(f"oh noses u have a problem in {path}: {e}")
```

WORD-LEVEL FEATURES

- [x] ~~Add intro section to CYSF portal~~

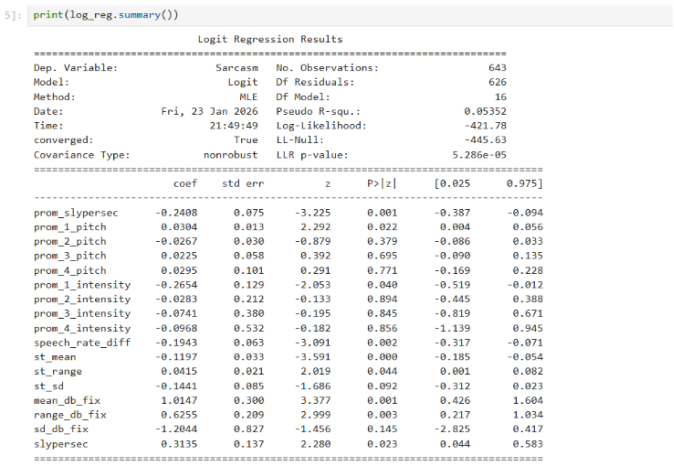
<table>
<tr>
<td></td>
<td>


Check-in notes:

Get portal done
Trifold: get tomorrow after school on 3rd floor
Look at intro

</td>
</tr>
<tr>
<td>Jan 22</td>
<td>
<u>Meeting notes:</u>

Prob going to print out a poster
Try both only clustering or only leg coefs to see which improves performance
Keep the train and test set the same from the summary to the predict

</td>
</tr>
<tr>
<td>Jan 23</td>
<td>
☑ ~~Try both only clustering or only leg coefs to see which improves performance~~
<h2>Word level feature (binary) w leg coefs</h2>

5]: print(log_reg.summary())
Logit Regression Results
Dep. Variable: Sarcasm No. Observations: 643
Model: Logit Df Residuals: 626
Method: MLE Df Model: 16
Date: Fri, 23 Jan 2026 Pseudo R-squ.: 0.05352
Time: 21:49:49 Log-Likelihood: -421.78
converged: True LL-Null: -445.63
Covariance Type: nonrobust LLR p-value: 5.286e-05
coef std err z P>|z| [0.025 0.975]
prom_slypersec -0.2408 0.075 -3.225 0.001 -0.387 -0.094
prom_1_pitch 0.0304 0.013 2.292 0.022 0.004 0.056
prom_2_pitch -0.0267 0.030 -0.879 0.379 -0.086 0.033
prom_3_pitch 0.0225 0.058 0.392 0.695 -0.090 0.135
prom_4_pitch 0.0295 0.101 0.291 0.771 -0.169 0.228
prom_1_intensity -0.2654 0.129 -2.053 0.040 -0.519 -0.012
prom_2_intensity -0.0283 0.212 -0.133 0.894 -0.445 0.388
prom_3_intensity -0.0741 0.380 -0.195 0.845 -0.819 0.671
prom_4_intensity -0.0968 0.532 -0.182 0.856 -1.139 0.945
speech_rate_diff -0.1943 0.063 -3.091 0.002 -0.317 -0.071
st_mean -0.1197 0.033 -3.591 0.000 -0.185 -0.054
st_range 0.0415 0.021 2.019 0.044 0.001 0.082
st_sd -0.1441 0.085 -1.686 0.092 -0.312 0.023
mean_db_fix 1.0147 0.300 3.377 0.001 0.426 1.604
range_db_fix 0.6255 0.209 2.999 0.003 0.217 1.034
sd_db_fix -1.2044 0.827 -1.456 0.145 -2.825 0.417
slypersec 0.3135 0.137 2.280 0.023 0.044 0.583
</td>
</tr>
</table>

Word level feature (binary) w clustering (both pitch and iten)

Logit Regression Results

Dep. Variable:	Sarcasm	No. Observations:	643
Model:	Logit	Df Residuals:	632
Method:	MLE	Df Model:	10
Date:	Fri, 23 Jan 2026	Pseudo R-squ.:	0.04624
Time:	21:50:53	Log-Likelihood:	-425.02
converged:	True	LL-Null:	-445.63
Covariance Type:	nonrobust	LLR p-value:	1.035e-05

	coef	std err	z	P>|z|	[0.025	0.975]
prom_slypersec	-0.2396	0.082	-2.928	0.003	-0.400	-0.079
cluster_allpitch	-0.1098	0.078	-1.408	0.159	-0.263	0.043
cluster_allintensity	0.1474	0.129	1.141	0.254	-0.106	0.401
speech_rate_diff	-0.1862	0.067	-2.762	0.006	-0.318	-0.054
st_mean	-0.0548	0.022	-2.472	0.013	-0.098	-0.011
st_range	0.0413	0.020	2.033	0.042	0.001	0.081
st_sd	-0.1497	0.084	-1.785	0.074	-0.314	0.015
mean_db_fix	0.7507	0.242	3.101	0.002	0.276	1.225
range_db_fix	0.6252	0.205	3.049	0.002	0.223	1.027
sd_db_fix	-1.4405	0.798	-1.806	0.071	-3.004	0.123
slypersec	0.3238	0.141	2.291	0.022	0.047	0.601

- [x] ~~Keep the train and test set the same from the summary to the predict~~

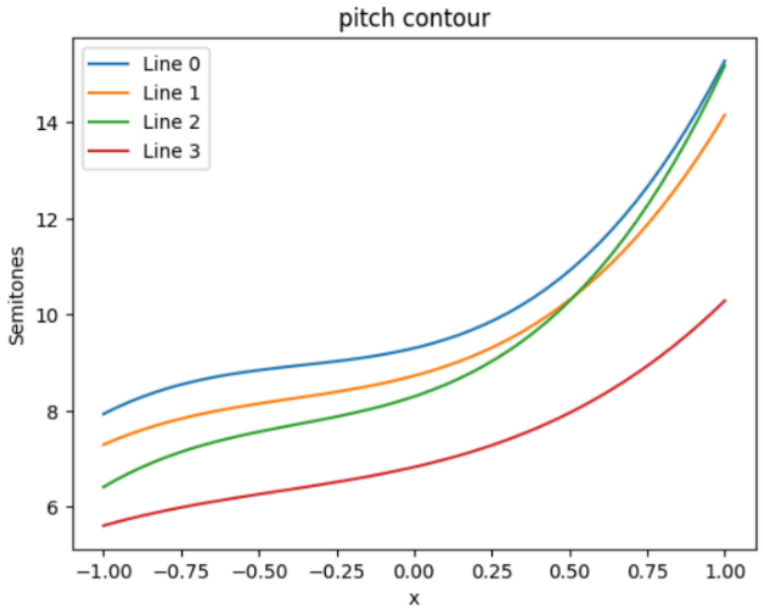
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8,
                                                    random_state=42, stratify = y)
X_train = X_train.apply(pd.to_numeric, errors='coerce')
```

```
log_reg = sm.Logit(y_train, X_train).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.672294
         Iterations 5
```

```
print(log_reg.summary())
```

- [x] ~~Notes on results slides:~~
  - [x] ~~Find mean of leg coefs for classes and graph it~~



Otc I'n
differe
the pr
cluste
sarcas

<table>
<tr><td></td><td>
☑ ~~Send A and B graphs to clustering group freq~~
SLIDES 48-52
☑ ~~Finish preprocessing section of methodology~~
Preprocessing

We retrieved audio clips and sarcasm labels from the MUStARD++ GitHub repository. Audio clips were downloaded as .mp4 files and sarcasm labels as a .csv file. We converted the audio files to .wav files and removed two speakers with non-American accents. There were also three “general” speaker labels that encompassed a large number of speakers. Because we were not able to find mean speaker pitch or intensity, we also removed utterances from speakers with general labels. Utterances were also removed due to background music or being too short to extract pitch and intensity contours. There were two speakers that were unnamed, “Person1” and “Person3.” These speakers were also included.

<table>
<tr><td>Reason</td><td>Sarcastic</td><td>Sincere</td></tr>
<tr><td>Not an American accent</td><td>42</td><td>32</td></tr>
<tr><td>Loud</td><td>1</td><td>1</td></tr>
<tr><td>Background music/sound</td><td>25</td><td>9</td></tr>
<tr><td>Too much cut off/too short</td><td>5</td><td>9</td></tr>
<tr><td>Unclear who speaker is</td><td>0</td><td>1</td></tr>
<tr><td>Generic speaker</td><td>24</td><td>48</td></tr>
<tr><td>Too small for normalization (<10)</td><td>9</td><td>8</td></tr>
<tr><td>Prominent word too short to extract contours</td><td>1</td><td>5</td></tr>
</table>

In total, 220 utterances were removed. The final dataset has 872 utterances, 439 sarcastic and 433 sincere. Most of the utterances included laughter at the start and end. We manually opened all utterances in Praat and trimmed the utterance to only include the dialogue. The dataset included a transcript of the utterance in the dataframe. To align the transcript with the audio, we converted the transcript into .txt files. Montreal Forced Aligner (MFA) was used to align the text to the utterance. We ran MFA g2p to find words that were not in the MFA pronunciation dictionary and add the ARPABET pronunciation.
</td></tr>
<tr><td>Jan 24</td><td>
☑ ~~Finalize results~~
results/analysis rough draft
</td></tr>
<tr><td>Jan 25</td><td>
☑ ~~Send all graphs to A and B~~

Email:

Hi Brooklyn and Dr Anthanasopoulou,

For the statsmodel report, I split the data into testing and training sets before only training the model with the training set so there is less leakage. I then used the same training and testing sets on a sklearn logistic regression model for the “official results”

Overall, all the models tested have poorer performance than Thursday. The data leakage must have made a big impact on performance. Currently, I’m selecting features by only including the significant
</td></tr>
</table>

	features the statsmodel model found in the actual results. How effective would this be, though? Even if a feature is statistically significant the difference could be very small and still not helpful. I also tried selecting the features with the biggest regression coefficient, and found better performance Also for the multiclass model, there is a problem with selecting by significance because some features are significant for one class and not for another. Perhaps I should focus on comparing each sarcasm type individually with sincere utterances? In the doc I also added normal reporting of features, which I find more interesting. results/analysis rough draft Andi
Jan 28	☑ ~~Submit EDA results on slides~~ (SLIDES 25-46) Notes: ● Intro handed back soon
Jan 30	☑ ~~Send methodology to B and A for review~~

Sarcasm Type

Sarcastic utterances can be divided into three groups with distinctive prosodic utterances. (Attardo et al. 2003) These three categories are within-statement contrast; monotone, or compressed pitch pattern; and exaggerated, or pronounced pitch accents. Utterances with strong within-statement contrast contain sections with high mean pitch and pitch range and sections with low mean pitch and pitch range. Monotone type is characterized by highly compressed pitch range. Exaggerated sarcastic utterances have pronounced pitch accents on many words in the utterance, elongation on syllables, and rhythmic structures.
As sarcasm type can heavily impact the prosody of the utterance, we manually annotated sarcastic utterances in the dataset with the sarcasm type. We confirmed our annotations with four native English speakers.

Global Features

Following Rakov's (2013) methodology, we measured seven global features. (Rakov and Rosenberg 2013) We used the Paselmouth library to extract speech features. (Jadoul et al. 2018)

A. Mean pitch, pitch range and Standard Deviation (SD) (in semitones)
B. Speech rate (syllables per second)
C. Mean intensity, intensity range and SD (in z-scores)

The pitch and intensity contours were initially represented as an array of pitch values in Hz and intensity values in db. The measurement interval for pitch was 60 milliseconds and 64 milliseconds for intensity. (Time step settings) To avoid an inflated pitch range value, we dropped all zeros in the pitch contour and interpolated the pitch.

We measured pitch in semitones instead of Hertz b

Intensity features were reported as z-scores. We calculated the mean and SD intensity of the speaker in db, then converted the intensity contour of each utterance from db to z-scores using speaker mean as the mean and speaker SD as SD. The mean, range, and SD of intensity were computed as the mean, range and SD of the z-score array respectively for each utterance.

The Python wrapper for CMU Pronouncing Dictionary (CMUdict) was used to determine the number of syllables in an utterance. Each vowel was counted as a syllable.

Word-level Features

We measured the following word-level features. All features were measured only from the prominent word of the sentence.

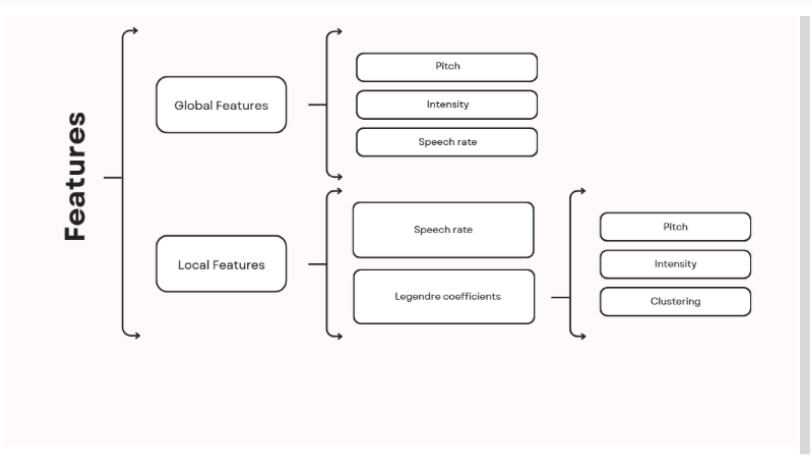
A. Values of the first four Legendre coefficients for both intensity and pitch
B. Speech rate of prominent words (syllables/second)
C. Difference between speech rate of prominent word and non-prominent words in the utterance
D. Cluster type of Legendre coefficients

To identify the prominent word, we added a new point tier to the Textgrid files of all utterances and added a boundary that was within the timeframe of the prominent word. Our selection was confirmed by one of four native English speakers.

Speech rate was calculated by dividing the duration of the prominent word by the number of syllables of the word. CMUdict was used to determine syllable count. Syllable count was manually added for 25 words because they were not in the pronunciation dictionary. The speech rate of the non-prominent words in an utterance was calculated by subtracting the syllable count and duration of the prominent word from the syllable count and duration of the utterance respectively and calculating the quotient. The difference in speech rates between prominent and non-prominent words is (non-prominent speech rate) - (prominent speech rate). A negative value can be interpreted as the prominent word being said faster than the rest of the sentence, while a positive value is the opposite.

Legendre polynomials are a low dimensional method for representing pitch and intensity contours. Pitch and intensity values can be represented as curves which are then simplified with Legendre polynomials. (Rakov 2019) The first four Legendre polynomials are adequate to represent distinctive prosodic pitch contours. (Svatošová and Volín 2023) We used Scipy to extract the coefficients for the first four Legendre polynomials for the prominent word of the utterance. (Virtanen et al. 2020). Coefficient values were based on the original pitch (in semitones) and intensity contours (z-score) with the same settings as when extracting global features. As the coefficient of the first polynomial also represents the average of the contour, we used the first coefficient of pitch and intensity as the proxy for mean pitch and intensity of the prominent word.

K-means clustering groups data into k distinct groups. (Hearon 2025) We clustered the first four Legendre coefficients together with scikit-learn, where each label is determined by the cluster centroid the coefficients are closest to. (Pedregosa et al. 2011) We used either k=3, 4, or 5 while training models.



Data Split and Feature Selection

We used a 80/20 training testing split for utterances with the scikit-learn library. (Pedregosa et al. 2011) We first ran a simple logistic regression model with the training set with the Statsmodel library. (Seabold and Perktold 2010) To select features for the final models, we only included features with statistically significant p-values ($a<0.05$) and used a simple logistic regression model from scikit-learn.

Models Used

We developed several different types of logistic regression models. The ‘control model’ was a binary logistic regression model trained only on global features that differentiated between all sarcastic and sincere utterances. We also trained a ‘general’ model on both global and word-level features. We trained the statsmodel results on either raw Legendre coefficients, or pitch and intensity labels of k=3, 4, or 5.

We did not use a multinomial logistic regression model to differentiate the three sarcasm types because features would be significant for some sarcasm types and not others. We trained an additional three binary classification models that differentiated sarcasm types separately. For example, one model would classify sincere and exaggerated sarcastic utterances. The prerequisite statsmodels results were similarity split. As the number of sincere utterances outnumbered the number of sarcastic ones for each sarcasm type separately, we balanced the utterances before splitting into training and testing sets by randomly removing sincere utterances until the class sizes of the two groups were equal. The general model trained on k=3 clustering had the best performance out of the other variations; we used k=3 clustering labels as the training feature for the type specific models.

Model name	Global	Word -level	Clustering	Raw Coefficients	Features use
Control	X				
General with coefficients	X	X		X	
General with k=3	X	X	X		
General with k=4	X	X	X		
General with k=5	X	X	X		
Type specific (exaggerated)	X	X	X		
Type specific (phrase contrast)	X	X	X		
Type specific (monotone)	X	X	X		

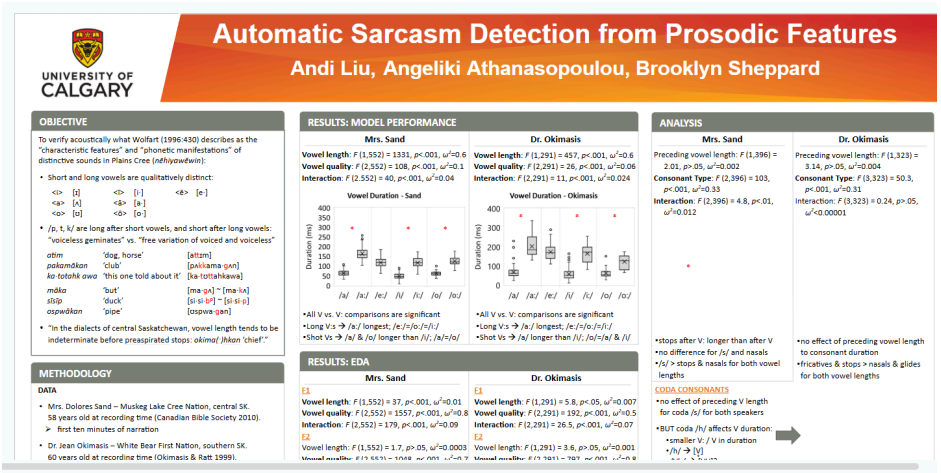
- ☑ ~~List all features used at the bottom of model performance (written in **pink**)~~
- ☑ ~~Watch and take notes on following videos:~~

[StatQuest: Logistic Regression]
[Logistic Regression Details Pt1: Coefficients]
[Logistic Regression Details Pt 3: R-squared and p-value]

Notes:

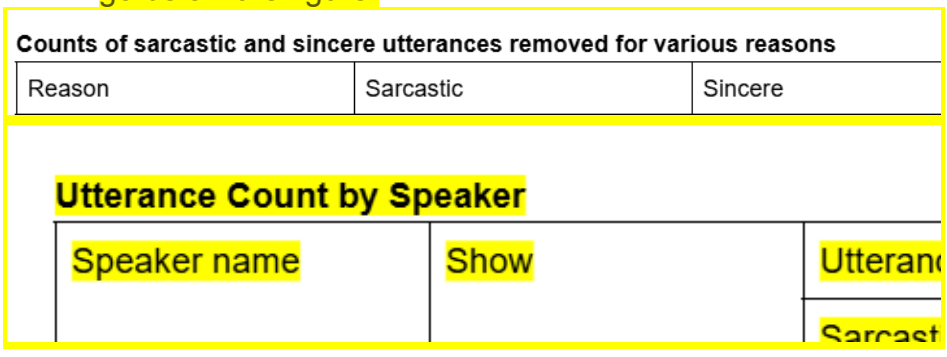
- Pacing
  - BG 1-2 min
  - Methods 2-3 min
  - Results 4-5 min
  - Analysis/Conc 2-3 min
- Log Reg
  - Predicts discrete values
  - Fits logistic function to points
    - 0-1, probability of True/False given input
  - Fits multiple features
    - Line fitted with maximum likelyhood
    - See weight of features
    - Sees if features impact on prediction is not 0

	Email: Hi Brooklyn and Dr Anthanasopoulou, Thanks for sending the poster examples! I tried finding a template online, but nothing looks as nice as the ones you guys showed me. Is there a set template you use? I'm done writing the first draft of the methodology. I was wondering if you could give some feedback on the tone and especially the feature extraction and model training section. I'm not sure I was clear enough or gave enough detail. https://docs.google.com/document/d/1Oy7XPdRYj3Lh-c74d7n4nEYYmOpjIaHciZsft3-vUYM/edit?usp=sharing See you on Saturday Andi
Jan 31	<u>Meeting Notes:</u> ● Discuss analysis and conclusion ● Should I email Rakov to ask for her P, R, F1 to compare? ○ Yes! ● Poster digital slides ● Send methods ● Write down results ● First draft of poster Graphs of the coefs…
February ❤️	☐ Present project ☑ ~~Finish poster~~
Feb 2	☑ ~~Send email to Rakok results request~~ <u>Email (rrakov@gmail.com/rrakov@sorenson.com):</u> Dear Dr Rakov, My name is Andi, I am a high school student from Canada and I'm writing a research paper on sarcasm detection. I recently read your 2013 Interspeech conference paper, which I found incredibly insightful. Your work inspired me to develop my own machine learning model focusing on prosodic features. I have a quick follow-up question:

	• Did you measure the precision, recall, and F1 score of the models you trained? I'm looking to compare metrics in my own study and would appreciate any additional data you might be able to share Your paper was very interesting and showed me that ML models do not need to be complicated to be effective. Thank you for your time and research, Andi
Feb 3-5	☑ ~~Start working on poster~~  **Automatic Sarcasm Detection from Prosodic Features** Andi Liu, Angeliki Athanasopoulou, Brooklyn Sheppard UNIVERSITY OF CALGARY OBJECTIVE To verify acoustically what Wolfart (1996:430) describes as the "characteristic features" and "phonetic manifestations" of distinctive sounds in Plains Cree (*nêhiyawêwin*): • Short and long vowels are qualitatively distinct: <i> [ɪ] <ī> [iː] <ê> [eː] <a> [ʌ] <â> [aː] <o> [o] <ô> [oː] • /p, t, k/ are long after short vowels, and short after long vowels: "voiceless geminates" vs. "free variation of voiced and voiceless" *atim* 'dog, horse' [ʌttɪm] *pakamâkan* 'club' [pʌkkamaːgʌn] *ka-totahk awa* 'this one told about it' [ka-tottahkawa] *mâka* 'but' [maːgʌ] ~ [maːkʌ] *sîsîp* 'duck' [si-siːpᵊ] ~ [si-siːp] *aspwâkan* 'pipe' [aspwaːgan] • "In the dialects of central Saskatchewan, vowel length tends to be indeterminate before preaspirated stops: *okimaʻ/ ihkan* 'chief'." METHODOLOGY DATA • Mrs. Dolores Sand – Muskeg Lake Cree Nation, central SK. 58 years old at recording time (Canadian Bible Society 2010). ➢ first ten minutes of narration • Dr. Jean Okimasis – White Bear First Nation, southern SK. 60 years old at recording time (Okimasis & Ratt 1999). RESULTS: MODEL PERFORMANCE Mrs. Sand **Vowel length**: F (1,552) = 1331, p<.001, ω^2=0.6 **Vowel quality**: F (2,552) = 108, p<.001, ω^2=0.1 **Interaction**: F (2,552) = 40, p<.001, ω^2=0.04 Vowel Duration - Sand Duration (ms) /a/ /aː/ /e/ /i/ /iː/ /o/ /oː/ •All V vs. V: comparisons are significant •Long V:s → /aː/ longest; /eː/=/oː/>/iː/ •Shot Vs → /a/ & /o/ longer than /i/; /a/=/o/ Dr. Okimasis **Vowel length**: F (1,291) = 457, p<.001, ω^2=0.6 **Vowel quality**: F (2,291) = 26, p<.001, ω^2=0.06 **Interaction**: F (2,291) = 11, p<.001, ω^2=0.024 Vowel Duration - Okimasis Duration (ms) /a/ /aː/ /e/ /i/ /iː/ /o/ /oː/ •All V vs. V: comparisons are significant •Long V:s → /aː/ longest; /eː/=/oː/=/iː/ •Shot Vs → /a/ longer than /i/; /o/=/a/ & /i/ RESULTS: EDA Mrs. Sand F1 **Vowel length**: F (1,552) = 37, p<.001, ω^2=0.01 **Vowel quality**: F (2,552) = 1557, p<.001, ω^2=0.8 **Interaction**: F (2,552) = 179, p<.001, ω^2=0.09 F2 Vowel length: F (1,552) = 1.7, p>.05, ω^2=0.0003 Dr. Okimasis F1 **Vowel length**: F (1,291) = 5.8, p<.05, ω^2=0.007 **Vowel quality**: F (2,291) = 192, p<.001, ω^2=0.5 **Interaction**: F (2,291) = 26.5, p<.001, ω^2=0.07 F2 Vowel length: F (1,291) = 3.6, p>.05, ω^2=0.001 ANALYSIS Mrs. Sand Preceding vowel length: F (1,396) = 2.01, p>.05, ω^2=0.002 **Consonant Type**: F (2,396) = 103, p<.001, ω^2=0.33 **Interaction**: F (2,396) = 4.8, p<.01, ω^2=0.012 •stops after V: longer than after V •no difference for /s/ and nasals •/s/ > stops & nasals for both vowel lengths Dr. Okimasis Preceding vowel length: F (1,323) = 3.14, p>.05, ω^2=0.004 **Consonant Type**: F (3,323) = 50.3, p<.001, ω^2=0.31 Interaction: F (3,323) = 0.24, p>.05, ω^2<0.00001 •no effect of preceding vowel length to consonant duration •fricatives & stops > nasals & glides for both vowel lengths CODA CONSONANTS •no effect of preceding V length for coda /s/ for both speakers •BUT coda /h/ affects V duration: •smaller V: / V in duration •/h/ → [ɣ]

	☑ ~~Finish writing results summary~~
	
	Results
	• Different speakers have different styles of sarcasm
	  ○ Ie Dorothy uses exaggerated sarcasm, Gilfoyle uses monotone sarcasm
	• Global features
	  ○ Sarcastic utterances are .91 st lower than sincere (-.56 v -1.37)
	  ○ On average, within phrase contrast utterances are 1.1 semitone lower than sincere utterances (-.56 v -1.65) and monotone are 1.3 st lower (-.56 v -1.91)
	  ○ Sarcastic utterances are louder than sincere ones (diff by 0.09 SD)
	  ■ Biggest contributor to this is exaggerated sarcasm (diff by 0.31 SD)
	  ○ Sarcastic utterances have a larger range (diff by 0.15 SD)
	  ■ Difference of 0.14 SD in within phrase contrast and 0.2 SD in exaggerated sarcasm
	• Local features
	  ○ In exaggerated sarcasm, the prominent word is 0.34SD louder
	• Some features aren't significant in the t-test but are for logistic regression
	  ○ Aren't expressed differently in sarcastic and sincere utterances but has predictive power?
	• Significance in the mean intensity of the prominent word and the whole utterance in exaggerated sarcasm could be due to the fact that exaggerated sarcasm is louder because there is no difference between sincere and sarcastic utterance
	
	☑ ~~Book times for Dr Garcia presentation~~
	☐ Get papers from library

	Papers to get: Underpinning / nailon /: Automatic Estimation of Pitch Range and Speaker Relative Pitch | Springer Nature Link Posters = 4ftx3ft • Work on video for platform • Send photo of trifold before school sf
Fed 7 💜	MEETING CANCELED
Feb 9	☑ ~~Look at methodology feedback~~ ☑ ~~"How to insert GitHub repo": typically you would include the link in a footnote.~~ Ask Dr Garcia about this.. ☑ ~~"Non-american accent"~~ ○ I would say something more like "remove two speakers that spoke with dialects other than Standard American English" or something like that? Essentially, I would avoid using the word "accent", and use "dialect" instead. ○ It would also be good to justify this choice: Perhaps you could find papers that show that prosody and sarcasm differ across dialects. ☑ ~~You might already be planning on this, but all tables/figures will need descriptive captions and be numbered. For captions, it's usually good for it to be detailed enough for the reader to understand what's going on without reading the text. For example, your first table might have something like "Counts sarcastic and sincere of datapoints removed from the dataset for various reasons".~~ ~~Table captions usually go above the table, figure captions usually go below the figure.~~ **Counts of sarcastic and sincere utterances removed for various reasons** Reason | Sarcastic | Sincere **Utterance Count by Speaker** Speaker name | Show | Utteran Sarcast



Summary of global and local features

Data Split and Feature Selection

Training features of various model

Model name	Glob	Wor	Clusterin	Rav

- [x] ~~You might want to very briefly mention what "MFA g2p" means - which is basically a pretrained grapheme-to-phoneme model that converts orthography to phonetic transcription.~~
- [x] ~~"Sarcastic utterances can be divided into…"~~
    - Very minor, but I would maybe soften this claim to something like "Previous work on the prosodic cues of sarcasm has proposed that sarcastic utterances can be divided into…", just because I don't think enough work has been done to confirm if this is true (but it's a useful assumption for us to make for this work)

Previous work on sarcasm prosody has divided utterances into three groups with distinctive prosodic features. (Attardo et al. 2003) These three categories are within-statement contrast; monotone, or compressed pitch pattern; and exaggerated, or pronounced pitch accents.

- [x] ~~It might be good to explicitly state your train/test size for each of the models as well, since those varied quite a bit.~~

Control	X				697/175	Mean, range, SD pitch of utterance; mean, range, SD intensity of utterance; speech rate of utterance
General with coefficients	X	X		X	697/175	Speech rate of prominent word and of utterance, speech rate difference, mean pitch and intensity of utterance, range of intensity of utterance
General with k=3	X	X	X		697/175	Mean pitch and intensity of utterance, range of intensity of utterance
General with k=4	X	X	X		697/175	Mean pitch and intensity of utterance, range of intensity of utterance
General with k=5	X	X	X		697/175	Speech rate of prominent word and of utterance,

General with k=4	X	X	X		697/175	Mean pitch and intensity of utterance, range of intensity of utterance
General with k=5	X	X	X		697/175	Speech rate of prominent word and of utterance, speech rate difference, mean pitch and intensity of utterance, range of intensity of utterance
Type specific (exaggerated)	X	X	X		164/42	Speech rate of prominent word, mean pitch of utterance, range and SD of intensity of utterance
Type specific (phrase contrast)	X	X	X		481/121	Mean pitch of utterance
Type specific (monotone)	X	X	X		51/13	No significant features

- [x] ~~Make model performance tables~~

<table>
<tr><td></td><td>
<table>
<tr><th>Model</th><th>Type specific, exaggerated</th><th>Control baseline</th><th colspan="2">(Wang et al.)</th><th>(Shi 2024)</th><th>(Xue et al. 2024)</th><th>(Gao et al. 2024)</th></tr>
<tr><td></td><td></td><td></td><td>Logistic regression</td><td>LSTM</td><td></td><td></td><td></td></tr>
<tr><td>Precision</td><td>0.71</td><td>0.64</td><td>0.55</td><td>0.80</td><td>0.62</td><td>0.73</td><td>0.81</td></tr>
<tr><td>Recall</td><td>0.71</td><td>0.64</td><td>0.67</td><td>0.54</td><td>0.61</td><td>0.72</td><td>0.81</td></tr>
<tr><td>F1</td><td>0.71</td><td>0.64</td><td>0.61</td><td>0.64</td><td>0.61</td><td>0.71</td><td>0.81</td></tr>
</table>
</td></tr>
<tr><td>Feb 10-11</td><td>☑ ~~Send draft to Dr Garcia for review~~
☑ ~~Send results to A and B~~

Email:
Hi Dr Garcia and Ms Kale,

I was wondering if you could give some feedback on my methodology. I will fix the formatting and citations later, but right now I’m looking for feedback on just the content.

https://docs.google.com/document/d/1Oy7XPdRYj3Lh-c74d7n4nEYYmOpjIaHciZsft3-vUYM/edit?usp=sharing

Also, how should I cite a github repository? Like a website?

Have a great long weekend!

Andi</td></tr>
<tr><td>Feb 14</td><td>Meeting notes:
● Some features were predictavily sig but not on the t-test → implications??
○ Sig based off log reg , not t-test
● British sarcasm paper → compare across dialects
○ Prosody differs across dialects
○ Cultural differences in sarcasm
●
● Rakov didn’t respond
● Found another good british sarcasm classification paper
● Add binary all to model comparison table
● Finish poster and slides
● Analysis:
○ Legendre coefficients
○ Dialect differences
○ Sarcasm type
○ On “acting” voice, speech mode
○ Speech rate differences
○ Comparable performances → audio only vs multimodal</td></tr>
</table>

	○ Performance jump in adding type ○ “Signicifant predictor” ● Citation : (number) in space ● Write test used ● Switch model performance and EDA section ● Logo → replace u of c with webber logo? ● No full sentences ● Talk to IT guy about unblocking Dr A’s email ● Methodology table → cut some out
Feb 15	Summary of meeting: ☑ ~~Analysis:~~ ☑ ~~Did legendre coefficients work?~~ ☑ ~~Comparisons in british vs american english~~ ☑ ~~Speech rate difference~~ ☑ ~~Impact of sarcasm type~~ ☑ ~~Compare model performance~~

ANALYSIS

Legendre coefficients

- No significance across sarcasm type → pitch + intensity contours not distinct enough/difference not on prominent word
- (X) found contribution of contour shape on sarcasm perception, not expression
- (X) found FPC of contours to be significant in detection of British English → possible impact of dialect + representation method

Speech rate

- Difference in speech rate predictor in noisier models → not consistent in utterance types
- Speech rate of prominent word slower in exaggerated sarcasm → possibly adds additional emphasis

Sarcasm type

- Performance jump in different types → influence of type
  - No significant predictors for monotone → perhaps monotone is prosodically similar to sincerity
  - Low performance on within-phrase contrast → type characterized by "sincere" part + "sarcastic" part ∴ interference from "sincere" part

Comparison to other models

- Performance of best model comparable to other models trained on MUStARD++

Orange = multimodal (audio + text), Red = audio only

Model name	**Type specific, exaggerated**	Control	(X)		(X)	(X)	(X)
			LR	LSTM			
Precision	**0.71**	0.64	0.55	0.80	0.62	0.73	0.81
Recall	**0.71**	0.64	0.67	0.54	0.61	0.72	0.81
F1	**0.71**	0.64	0.61	0.64	0.61	0.71	0.81

- Performance of best model comparable to other models trained on MUStARD++

Orange = multimodal (audio text), red = audio

Model	**Type specific, exaggerated**	Control	(X)		(X)	(X)	(X)
			LR	LSTM			
Precision	**0.71**	0.64	0.55	0.80	0.62	0.73	0.81
Recall	**0.71**	0.64	0.67	0.54	0.61	0.72	0.81
F1	**0.71**	0.64	0.61	0.64	0.61	0.71	0.81

- ☑ ~~In project~~
  - ☑ ~~Performance jump w adding type~~
- ☑ ~~With other projects~~

	☑ ~~Confounding factor → "acting" sarcasm~~ **CONCLUSION** • Sarcasm type influences system performance → need of type sensitive systems (eg. multimodal with text in monotone type) • Potential of LR to be effective in sarcasm detection → prosody based features useful in sarcasm detection **Future direction:** FPC, multimodal LR model, comparison to conversational sarcasm **Sources of error:** speakers "acting sarcastic" → prosodic differences bigger; noise in audio ☑ ~~Switch EDA and model performance section~~ ☑ ~~No full sentences, symbols~~ ☑ ~~Cut stuff out of methodology table~~

METHODOLOGY

Model name	Global	Local	Legendre Clustering	Legendre Coefficients
Control	X			
General (coefficients)	X	X		X
General (clustering)	X	X	X	
Type specific	X	X	X	

Feb 17-18

- ☑ ~~Send poster to sylvan, ms kale, A and B~~
- ☑ ~~Move poster to slide show, run through timing~~
- ☑ ~~Add references to poster~~

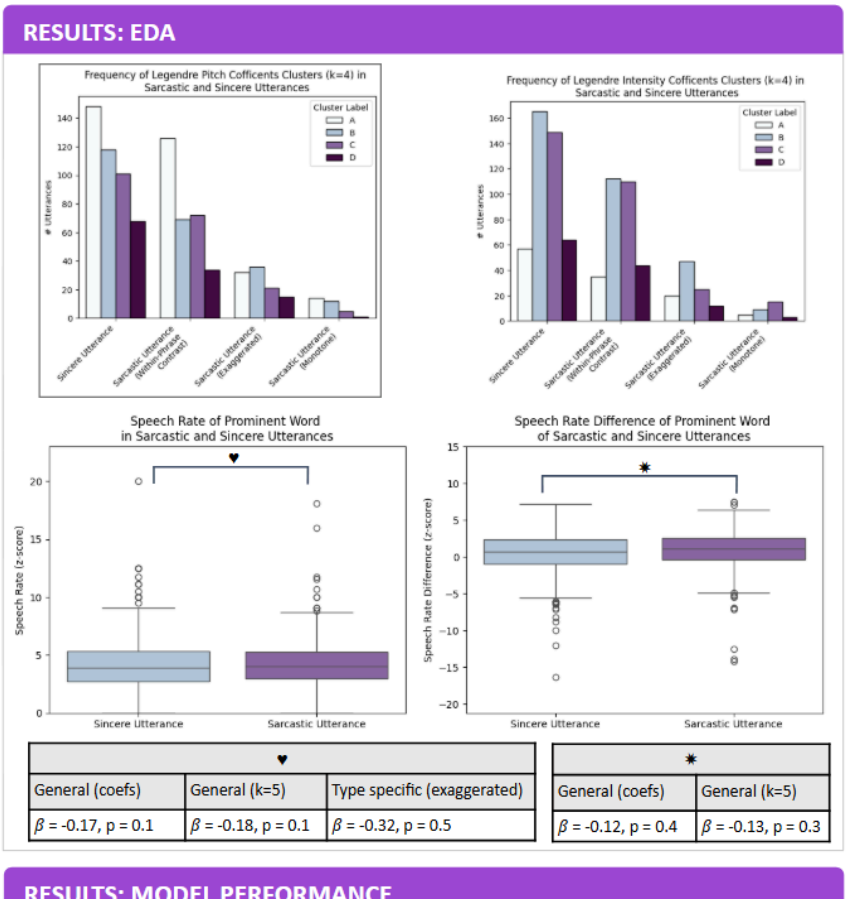
REFERENCES

MUStARD++ dataset:

Ray, A., Mishra, S., Nunna, A., & Bhattacharyya, Pushpak . (2022). A Multimodal Corpus for Emotion Recognition in Sarcasm. *Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022)*, 6992–7003. https://aclanthology.org/2022.lrec-1.756.pdf

(1) Attardo, S., Eisterhold, J., Hay, J., & Poggi, I. (2003). Multimodal markers of irony and sarcasm. *Humor: International Journal of Humor Research*, *16*(2), 243–260. https://doi.org/10.1515/humr.2003.012
(2) Sheppard, B., & Winters, S. (2022). *Effects of speaking rate and pitch accent types on the perception of verbal irony* [Undergraduate]. University of Calgary.
(3)Chen, A., & Boves, L. (2018). What's in a word: Sounding sarcastic in British English. *Journal of the International Phonetic Association*, *48*(1), 57–76. https://doi.org/10.1017/S0025100318000038
(4)Wang, Y., Huang, Z., Gandhi, M., & Chen, H. C. (2024). *Multimodal Sarcasm Detection Based on MUStARD Dataset* [Masters, University of Pennsylvania]. https://monagandhi09.github.io/asset/pdf/SarcasmDetector.pdf
(5)Shi, E. (2024). *Multimodal Sarcasm Detection Using BERT, TimesFormer, and Wav2Vec 2.0 with MUStARD++* [Masters]. University of Groningen.
(6) Xue, H., Xu, L., Tong, Y., Li, R., Lin, J., & Jiang, D. (2024). Breakthrough from nuance and inconsistency: Enhancing multimodal sarcasm detection with context aware self attention fusion and word weight calculation. In N. Calzolari, M. Y. Kan, V. Hoste, A. Lenci, S. Sakti, & N. Xue (Eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC COLING 2024)* (pp. 2493–2503). ELRA and ICCL. https://aclanthology.org/2024.lrec-main.224/
(7) Gao, X., Bansal, S., Gowda, K., Li, Z., Nayak, S., Kumar, N., & Coker, M. (2024). *AMuSeD: An attentive deep neural network for multimodal sarcasm detection incorporating bi-modal data augmentation*. arXiv. https://doi.org/10.48550/ARXIV.2412.10103

- [ ] Review feedback on methodology
- [x] ~~Make good copy of graphs~~
  - [x] ~~Legendre~~
    - [x] ~~Mean by type~~
    - [ ] For clusters, display all clusters at once (only if room)
  - [x] ~~Speech rate diff, and prom speech rate~~



Email:

Hi Brooklyn and Dr Anthanasopoulou,

I finished the rough draft of the poster! Here it is:
Andi-ASP-2026.pptx

I measured intensity and speech rate in z-scores, should I make a disclaimer at the beginning about this and just refer to them without referring units?

Andi

	_____ Hi Ms Kale, I finished the poster I was talking about today, I was wondering if you could take a look at it. Andi-ASP-2026.pptx Thanks! Andi Meeting notes: ● Ask K to look @ presentation ● How to show sig in graph? ● Add equation for speech rate difference ●
Feb 19-20	☑ ~~Present practice presentation~~ ● Pacing ○ BG 1-2 min ○ Methods 2-3 min ○ Results 4-5 min ○ Analysis/Conc 2-3 min
Feb 21	☐ Send A and B rubrics b4 meeting (CANCELLED: SICK) Meeting notes: ● Go over feedback ● What to say when asked about contribution of mentors? ● Reduce speaking time ● Enough emphasis on impact/analysis? ● Send cysf rubric
Feb 23	☑ ~~Presenting today~~
Feb 25	☐ Send poster to printing ☐ Talk about problem first, or audio intro first

	Notes: ● Structure of Al SiC ○ How chose the different percentage steps of silicon carbide? ○ Why didn't do centering stage? ● Obesity ○ High fat intake effect astrocytes, too ○ How much work did she do with the mice? ○ Problems with low sample size? Representative in humans? ● Kidney structure + diabetes ○ Low sample size?? ● POTs ○ Use of LBNP? ○ Applications of results ○ # of patients??? ● CRISPR Cas 9 ○ What speech disorders are caused? ○ Stats tests? Also, ● Global/Local feature define better ● "Not significant" ● Explain coefficients ● Highlight cross Email: Hi Brooklyn and Dr Anthanasopoulou, I gave my class presentation on Monday and the school fair is next Monday. The class presentation went okay, but I have a lot of stuff to cut out of my speech. I'm going to print the poster soon, but wanted to get it checked one last time. I could just wait for our Saturday meeting and print it after if there's time. What is the name of the printing company? I can also use Staples. I've also attached the rubric used for judging for interest. I hope you're feeling better, Dr Anthanasopoulou! Andi
Feb 26	☑ ~~To get from dollar store:~~ ☑ ~~Large and small plastic eggs~~ ☑ ~~Sharpie~~

	☑ ~~Large and small rectangle box~~ ☑ ~~2 medium square box~~ Notes: Intro → talk lots about sarcasm types → humans vs machines detection ability Methodology → training testing → type specific has local features in it Results → focus on only accuracy Discussion → different orange colours Conclusion → future direction of type → skip future directions + "Noise" → variability Subtypes is source of noise Shop name is bound and copied
Feb 27	☑ ~~PRINT TODAY!!~~
March 🍀	☐ Present at school fair
March 1	☐ Rehearse speech ☐ Finish CYSF platform ☐ Review methodology
	☑ ~~Format transcripts for MFA and align~~ ☐ Get syl per second ☐ Separate function and content words ☐ Add syl speed and changes in speed over time ☐ Can you graph syl speech over time? ☐ Can you measure relative speaking time of a word? ☐ Pre: can you make a histogram of sly/sec speech of each word for a speaker? ☐ Record leg coef of prominent words and all words by modifying b's code

	☐ Display distribution of runtime and remove utterances under 2 seconds ☐ Split utterances into 3 and record pitch and intensity features for each third
For reference	☐ Take notes on the following: ☐ 1. Peterson, C., Wellman, H.M., & Slaughter, V. (2012). The mind behind the message: advancing theory of mind scales for typically developing children and those with deafness, autism, or Asperger syndrome. *Child Development, 83*, 469–485 ☐ 2. O'Reilly, K., Peterson, C., & Wellman, H. (2014). Sarcasm and advanced theory of mind understanding in children and adults with prelingual deafness. *Developmental Psychology, 50*(7), 1862-1877. ☐ 3. Gibbs, R. (2000). Irony in talk among friends. *Metaphor & Symbol, 15*, 5-27. ☐ 4. Hancock, J. (2004). Verbal irony use in computer-mediated and face-to-face conversations. *Journal of Language and Social Psychology, 23*, 447-463.