University of Stuttgart
Germany

Visual World Paradigm

Acquisition and analysis of eye-tracking data (SoSe 23)

Team Pegasus

- Pritom Gogoi
- Manpa Barman
- Kapil Mulchandani

Research Question

> *Does the presence of similar-sounding words influence our tendency to focus more on pictures that match those words in a visual world paradigm study?*

Background

What is Visual World Paradigm?

The visual world paradigm is an experimental framework that investigates language processing by monitoring participants' eye movements while they interact with visual stimuli.

Relevance to our project?

Our project is to study the nature of spoken word recognition as the word unfolds. Here the key aspect of the visual world paradigm is that participants' eye movements serve as an index of their ongoing language processing and interpretation.

Reference Paper: Allopenna, P., Magnuson, J., & Tanenhaus, M. (1998). Tracking the time course of spoken word recognition using eye movements: evidence for continuous mapping models. Journal of Memory and Language, 38, 419-439.

Background (contd.)

What are the key conclusions investigated in our project regarding spoken word recognition and the underlying models?

- Spoken word recognition is dynamic in nature which suggests that listeners continuously update and refine their interpretations as more information becomes available.
- Spoken word recognition models make assumptions that multiple candidates compete for recognition during the unfolding of the spoken word.

Background (contd.)

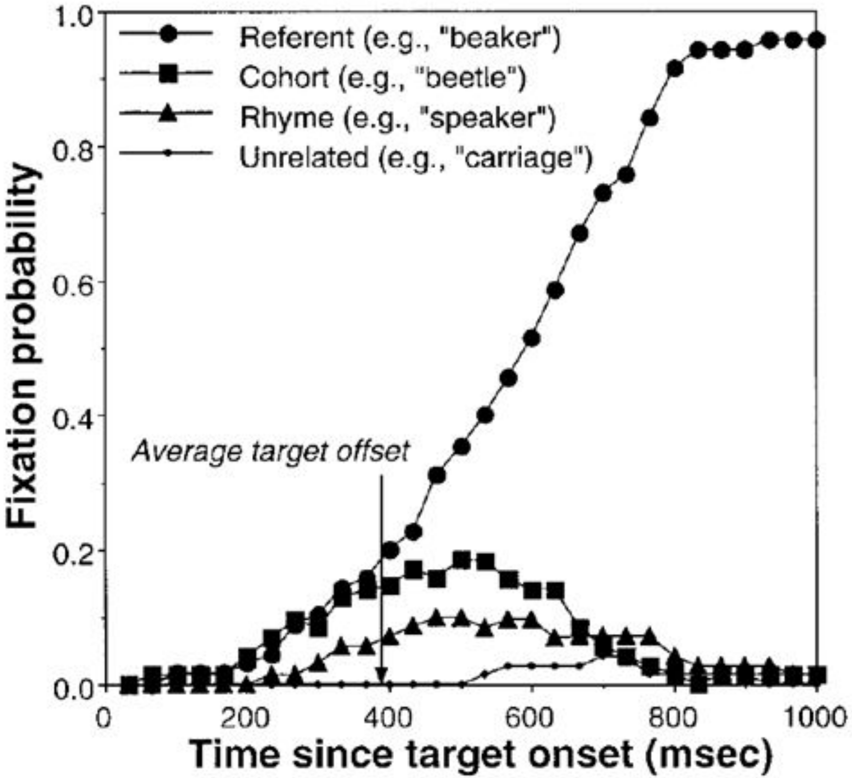
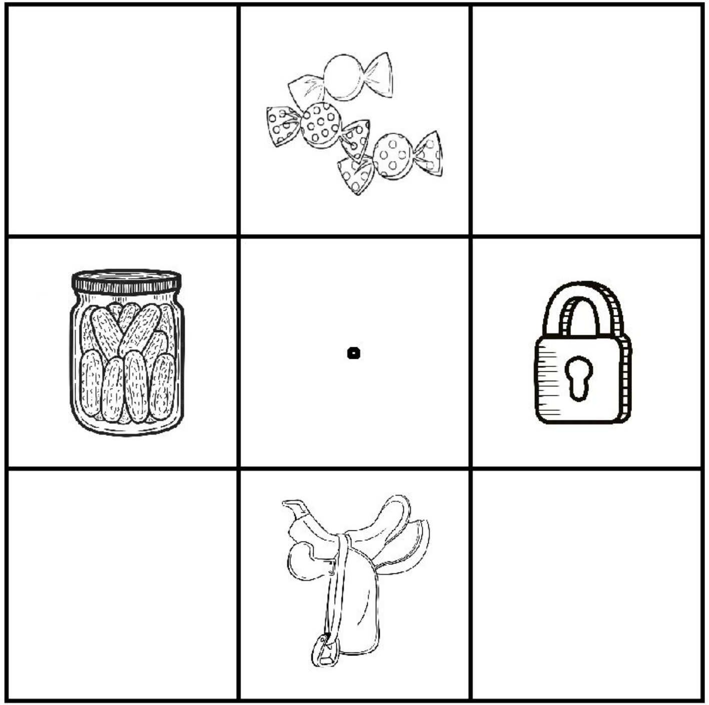


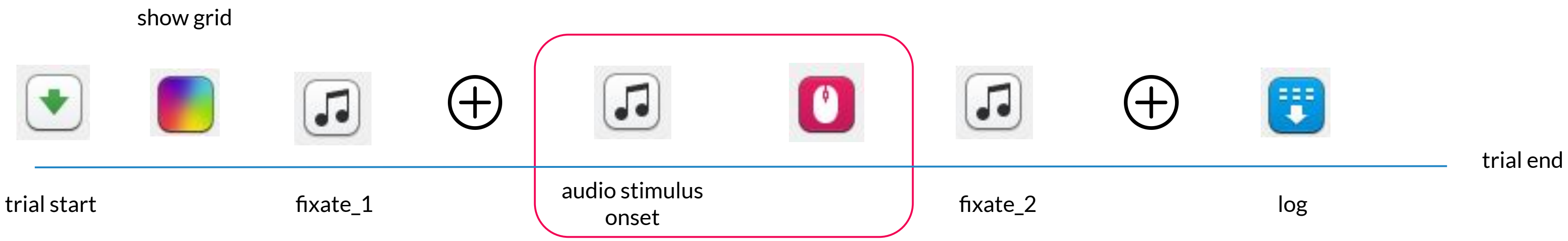
FIG: Probability of fixating on each item type over time in the full competitor condition

Design and Logic

Experiment UI



Timeline of a trial



show grid

trial start

fixate_1

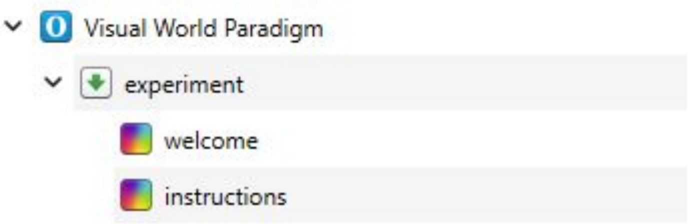
audio stimulus onset

fixate_2

log

trial end

Intro



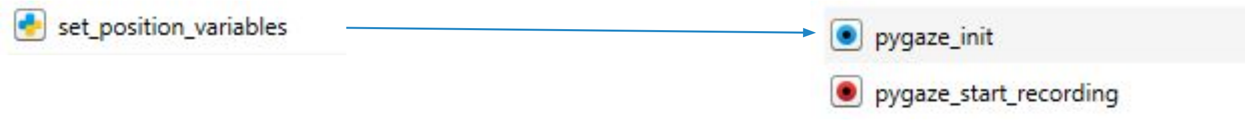
- Visual World Paradigm
  - experiment
    - welcome
    - instructions

- Contains introduction and preliminary instructions
- Progression requires mouse clicks

Backend: PsychoPy

Foreground: black
Background: white

Intro



set_position_variables

pygaze_init

pygaze_start_recording

- Initializes variables that are required in the trials

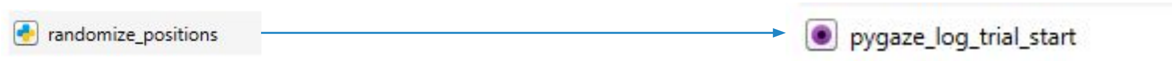
- Initializes the eye tracking hardware and starts the recording of the samples
- Logs: `START_EXP`

Trial loop

- The order of the trials is randomized
- Loop data is read from a file named `stimuli.csv` that contains the specific data required for each trial

The rest of the elements are children of this node.

Trial loop items



randomize_positions

pygaze_log_trial_start

- Randomizes the positions of the stimuli in the grid box
- Sets the variables:
  - img_top
  - img_left
  - img_bottom
  - img_right

- Log indicating the start of a trial

```
fixate_center_audio_onset,
cond: [condition]
target: [target]
```

Trial loop items

grid_stimulus

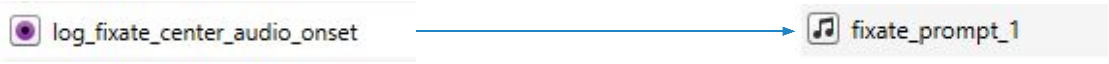
pygaze_log_trial_start

- Displays the 3x3 grid box along with the stimuli
- Duration: 3 s
  - To allow the participant to explore the elements on the screen

- Log indicating the start of a trial

```
start_trial: [count_trial_sequence]
t: [img_top]
r: [img_right]
b: [img_bottom]
l: [img_left]
```

Trial loop items



log_fixate_center_audio_onset

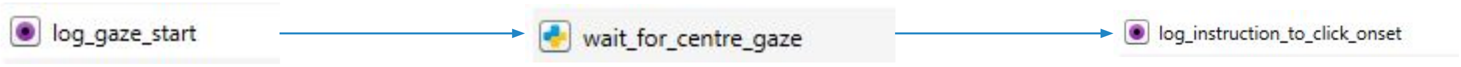
fixate_prompt_1

- Indicates the onset of the audio prompt for initial fixation at the center

```
fixate_center_audio_onset,
cond: [condition]
target: [target]
```

- Audio item for prompt “Fixate at the centre”

Trial loop items



- Logs:

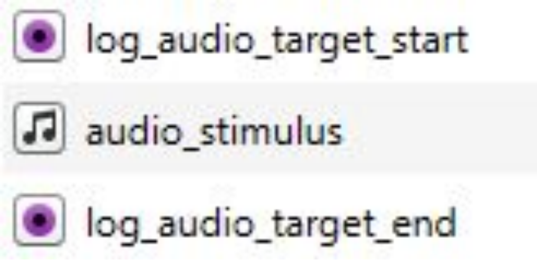
  `centre_gaze_start`

- Samples and checks if the participant's gaze is within a certain distance from the center
- Removed and later replaced by a delay of 1.1 s

- Log to indicate the onset of the instruction to click on a certain stimulus

  `instruction_to_click_onset`

Trial loop items



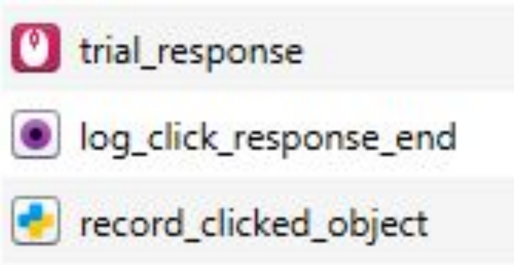
log_audio_target_start

audio_stimulus

log_audio_target_end

- Logs `log_audio_target_start`
- Plays the audio stimulus
- Logs `log_audio_target_end`

Trial loop items



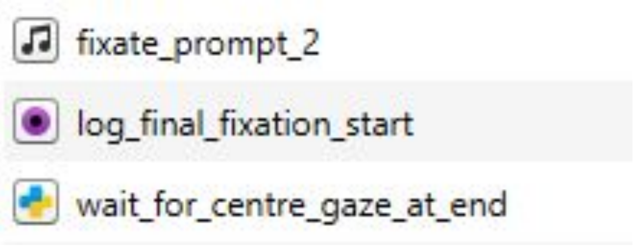
trial_response

log_click_response_end

record_clicked_object

- Records mouse click response
- Logs `click_response_end`
- Stores the id of the clicked object

Trial loop items



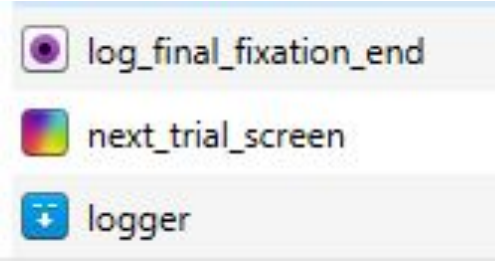
fixate_prompt_2

log_final_fixation_start

wait_for_centre_gaze_at_end

- Audio prompt to fixate at the center
- Logs `final_fixation_start, selected: [clicked]`
- Fixation check for the center (same as previous), also replaced by a delay

Trial loop items



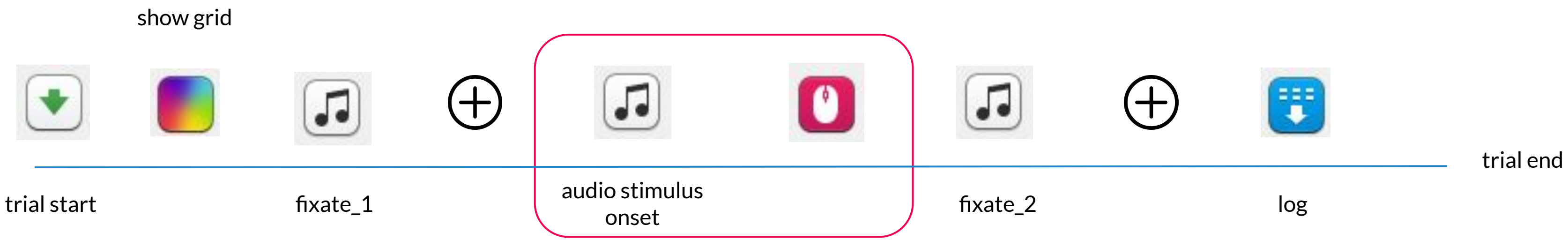
log_final_fixation_end

next_trial_screen

logger

- Logs `final_fixation_end`
- A screen indicating the end of a trial
- Logger (for all variables)

Timeline of a trial (recap)



show grid

trial end

trial start

fixate_1

audio stimulus
onset

fixate_2

log

Overall Logic

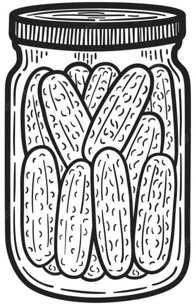
- Stimuli are chosen as per the different pairs of sets included in the original paper.
    - The authors chose the pairs based on frequency (per million words in the Kucera and Francis, 1967, corpus)
- A fixation at a point on the screen indicates that the participant is paying attention to it.
- Noting timestamps of the samples is essential. Our analysis depends on the chronology of events.
- Fixations at the centre of the screen marks the start and end of a trial.

Stimulus Design

Line drawings

Sources:

- Image search for assets licensed under Creative Commons BY-SA
  - Direct use
  - Edge detection in GIMP



Specifications:

- 256x256
- PNG Format

Line drawings (contd.)

Choice of Stimuli:

Based on the eight ‘referent – cohort – rhyme – unrelated’ sets:

- beaker, beetle, speaker, dolphin
- carrot, carriage, parrot, nickel
- candle, candy, handle, dollar
- pickle, picture, nickel, speaker
- casket, castle, basket,nickel
- paddle, padlock, saddle, dollar
- dollar, dolphin, collar, beaker
- sandal, sandwich, candle, parrot

Auditory stimuli

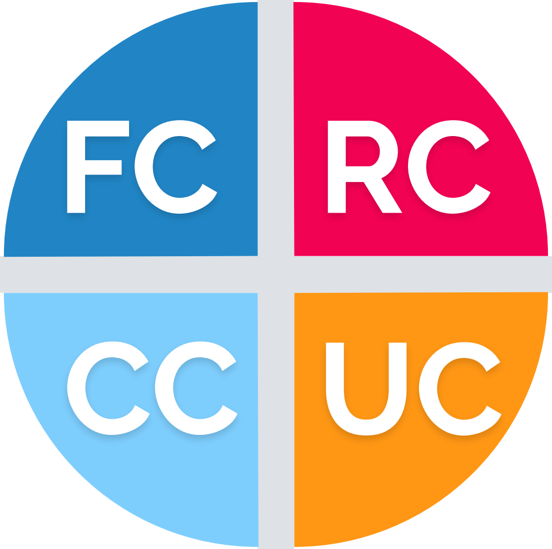
Sources:

- Generated using app.acoust.io
  - Voice profile: Davis
  - Playback speed: 0.8x

Specifications:

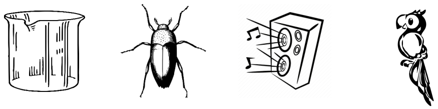
- Sampling rate: 48Hz (relevant for use with psychoPy backend in OpenSesame)

Conditions



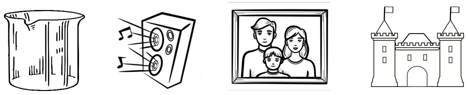
Full Competitor Set (FC)

a referent (Re), a cohort (C), a rhyme (Rh) and an unrelated (U)



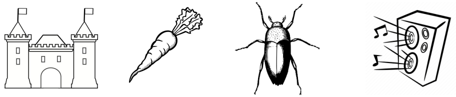
Rhyme Competitor Set (RC)

a referent (Re), a rhyme (Rh), two unrelated (U)



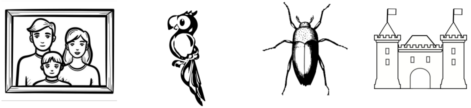
Cohort Competitor Set (CC)

a referent (Re), a cohort (C) and two unrelated (U).



Unrelated Set (UC)

a referent (Re), three unrelated (U)



Condition Chart

Condition No	**Trials (how many examples of such)**	**Conditions**		
		Competitor Set	Target	Distractors
1	3	FC	Re	C, Rh, U
2	3	FC	C	Re, Rh, U
3	3	FC	Rh	Re, C, U
4	3	FC	U	Re, C, Rh
5	3	CC	Re	C, U, U
6	3	CC	C	Re, U, U
7	3	CC	U	Re, C, U
8	3	RC	Re	Rh, U, U
9	3	RC	Rh	Re, U, U
10	3	RC	U	Re, Rh, U
11	3	UC	Re	U, U, U
12	3	UC	U	Re, U, U

Randomization of stimuli and trials

Order Randomization

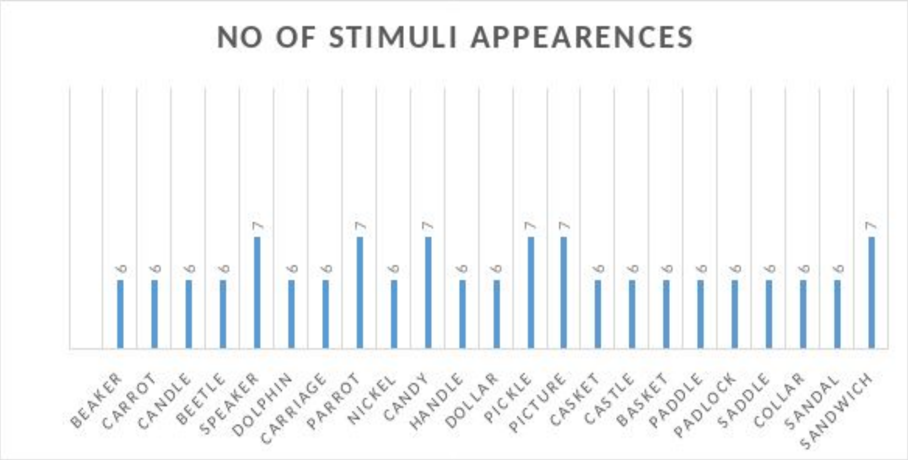
The order of the trials are randomized across participants.

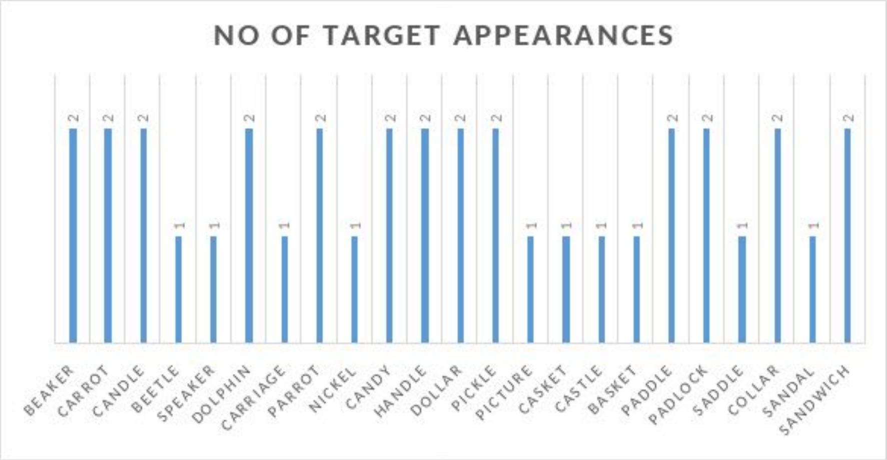
Stimuli Distribution

Each stimuli is shown approx. equal number of times.

Target Distribution

Each stimuli is chosen as the target an approx. equal number of times.





[Link to the stimuli distribution sheet]

How we organized the trials?

Total Number of Participants: 15

1. The GazePoint GP3 eye tracker was used for the experiment. Details on the GazePoint API can be found here.
2. We provided the participants with preliminary information regarding the experiment as per the protocols of data collection. You can find the information sheet and protocols that we used, here. We also asked the participants to sign the consent forms.
3. Prior to the experiment, the participants were made familiar with the stimuli by showing them the pictures of the stimuli on a sheet. They were asked to name each of the objects aloud, until named correctly.

How we organized the trials? (contd.)

4. The experiment is then started after calibrating the participants' eyes with the calibration software.
5. The experiment duration is around 18 mins per participant including the pre-experiment information and post-experiment discussion sessions.
6. After each experiment we briefed the participants on the main aim of our experiment and noted their feedbacks.

Quality control

Trials were not included in our analyses if:

- During a trial, the calibration deteriorated to such an extent that it was not possible to label fixations.
- the participant did not maintain fixation on the cross until the appropriate instruction began
- the participant never fixated on or selected the correct target

(Allopenna et al. 1998)

Quality control

Trials were not included in our analyses if:

- During a trial, the calibration deteriorated to such an extent that it was not possible to label fixations.
- the participant did not maintain fixation on the cross until the appropriate instruction began
- the participant never fixated on or selected the correct target

(Allopenna et al. 1998)

We discarded the readings of 3 participants due to quality issues (see later)

Quality control (contd.)

- Gaze samples relevant for our task?
  - From the onset of auditory stimuli: `LOG_AUDIO_TARGET_START`
  - until the mouse click response: `CLICK_RESPONSE_END`

- Verify all logs checkpoints exist for each trial
  - If missing, check and adjust sampling rate

Coordinates!

- OpenSesame to cartesian (scaled):

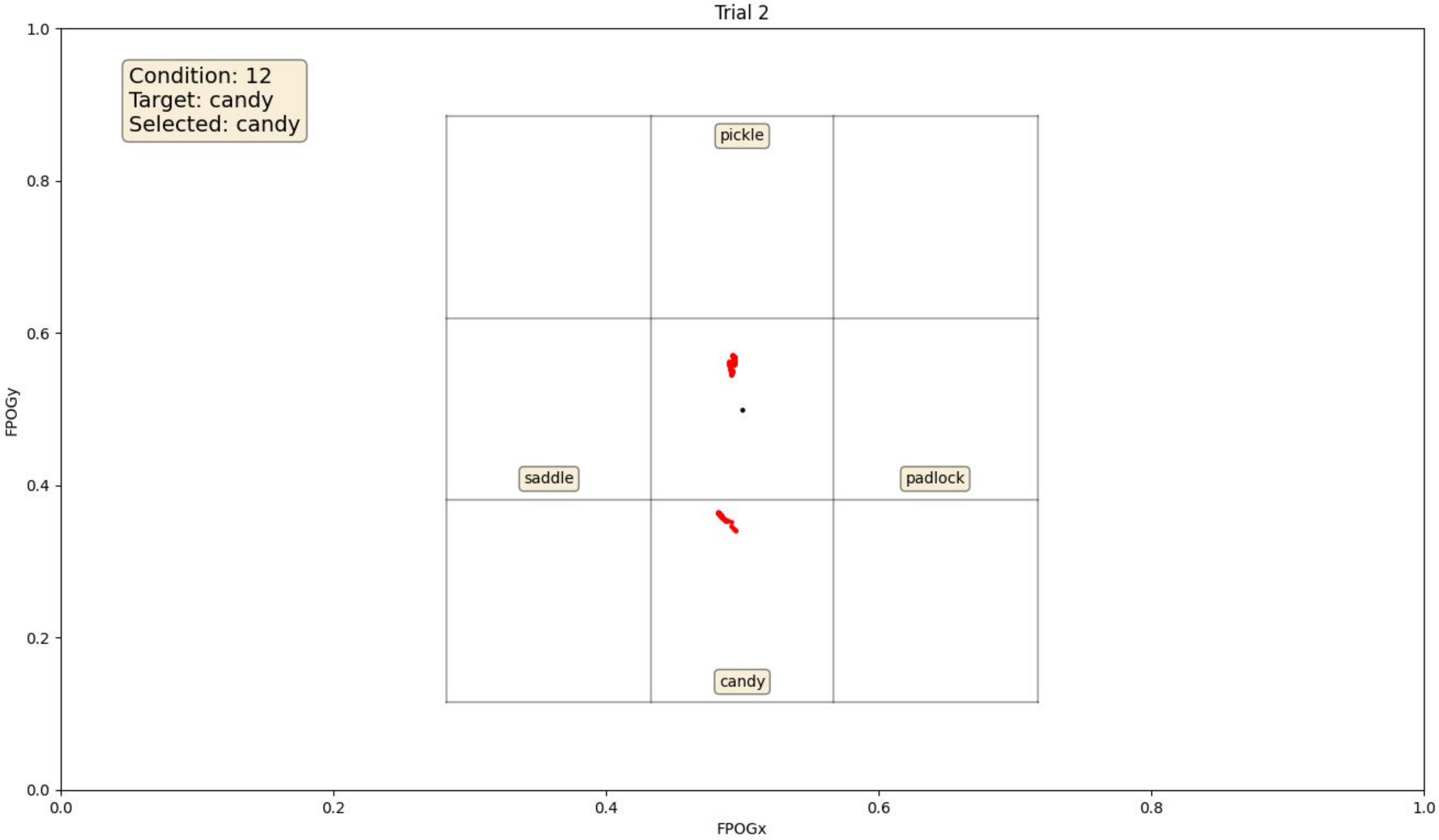
```
# shift the origin from (0, 0) to (-960, 540)
# perform the same on outer_points and inner_points
def shift_coordinate_system(coord_dict):
       for key, value in coord_dict.items():
              coord_dict[key] = (value[0] + 960, -1 * value[1] + 540)
              # scale to [0, 1]
              coord_dict[key] = (coord_dict[key][0] / 1920, coord_dict[key][1] / 1080)
return coord_dict
```

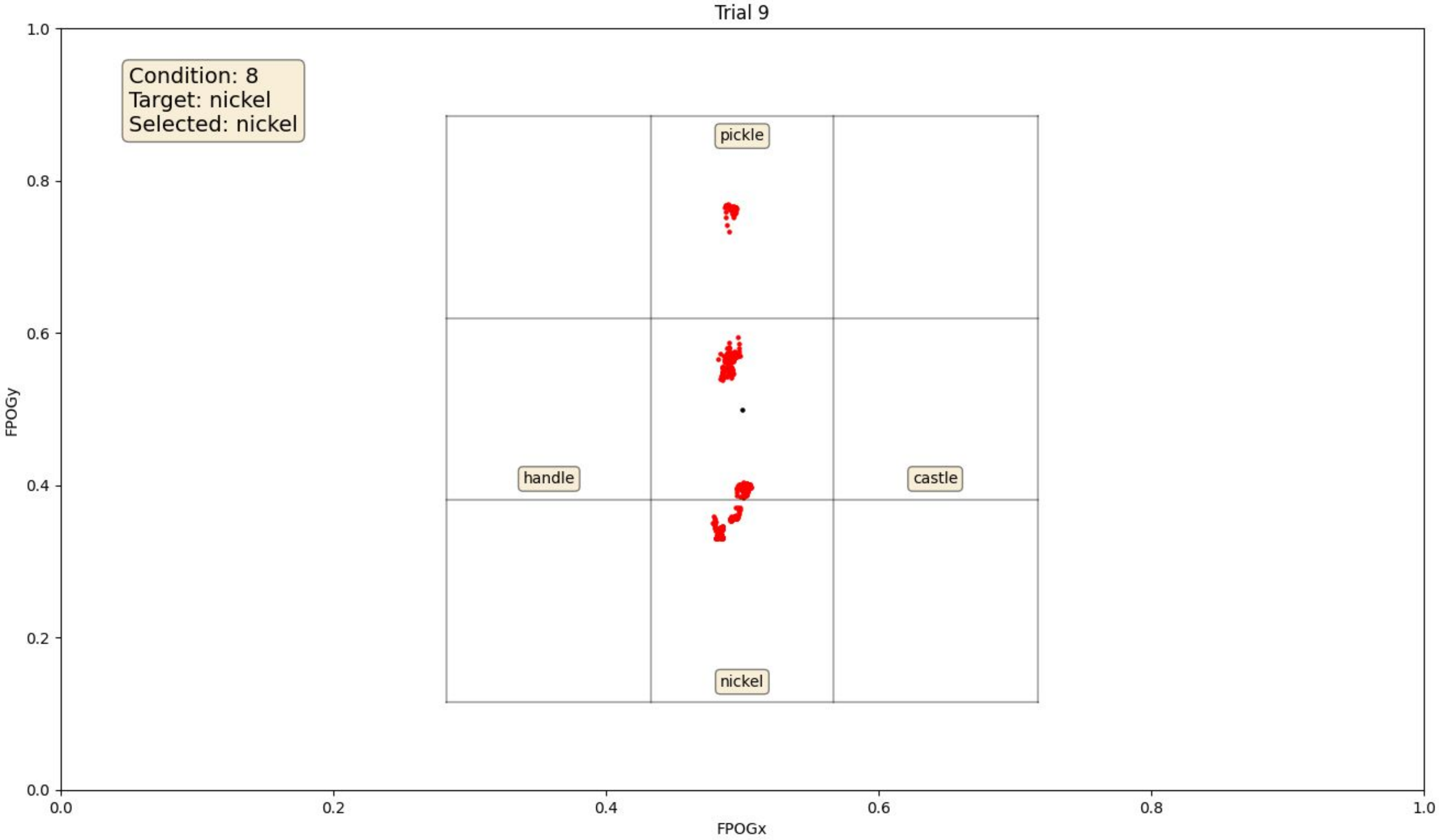
- OpenGaze to cartesian:

```
def shift_coordinate_system_bottom_left_to_top_left(x, y):
       return (x, -1 * y + 1)
```

Plots of fixations (spatial view)

- Plot FPOGX vs. FPOGY (red dots)
    - Plot only those sample with FPOGV == 1 (valid samples)
- Draw the stimuli grid box
- Add label on each of the stimulus boxes
- Add a legend to the plot
    - Condition number
    - Target
    - Selected stimulus


Trial 2
Condition: 12
Target: candy
Selected: candy
pickle
saddle
padlock
candy
1.0
0.8
0.6
0.4
0.2
0.0
FPOGy
0.0
0.2
0.4
0.6
0.8
1.0
FPOGx

Trial 9
Condition: 8
Target: nickel
Selected: nickel
pickle
handle
castle
nickel
FPOGy
FPOGx
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

Preprocessing & Analysis

Logs (recap)

```
START_EXP
START_TRIAL: 0 T: SADDLE.PNG R: PICKLE.PNG B: PADLOCK.PNG L: CANDY.PNG
FIXATE_CENTER_AUDIO_ONSET, COND: 12 TARGET: CANDY
CENTRE_GAZE_START
INSTRUCTION_TO_CLICK_ONSET
LOG_AUDIO_TARGET_START
LOG_AUDIO_TARGET_END
CLICK_RESPONSE_END
FINAL_FIXATION_START, SELECTED: CANDY.PNG
FINAL_FIXATION_END
….
….
….
….
STOP_EXP
```

Logs (recap)

```
START_EXP
START_TRIAL: 0 T: SADDLE.PNG R: PICKLE.PNG B: PADLOCK.PNG L: CANDY.PNG
FIXATE_CENTER_AUDIO_ONSET, COND: 12 TARGET: CANDY
CENTRE_GAZE_START
INSTRUCTION_TO_CLICK_ONSET
LOG_AUDIO_TARGET_START
LOG_AUDIO_TARGET_END
CLICK_RESPONSE_END
FINAL_FIXATION_START, SELECTED: CANDY.PNG
FINAL_FIXATION_END
….
….
….
….
STOP_EXP
```

Extract data from logs

```
# use regex to extract the number afer 'COND:'
cond_numbers = [re.findall(r'COND: (\d+)', row)[0] for row in target_rows]
# use regex to extract the word after 'TARGET:'
target_words = [re.findall(r'TARGET: (\w+)', row)[0] for row in target_rows]
# use regex to extract the word after 'SELECTED: '
selected_words = [re.findall(r'SELECTED: (\w+)', row)[0] for row in fixation_rows]
```

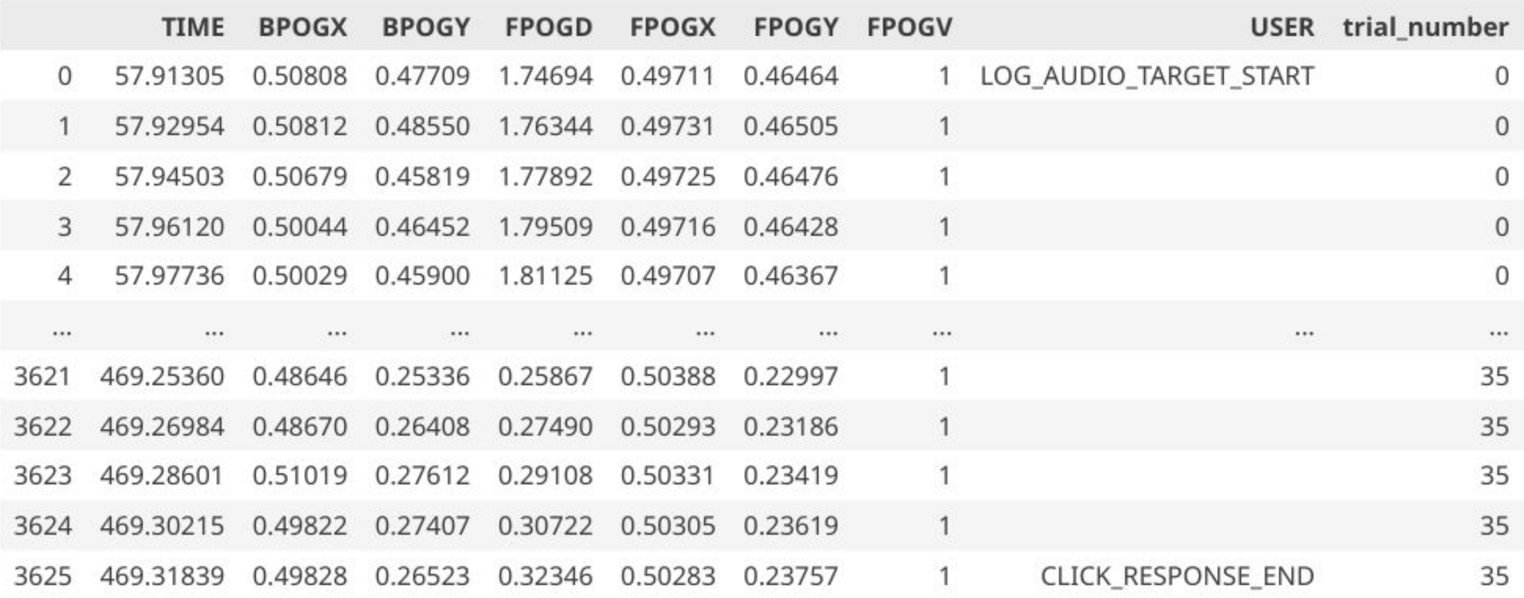
Extract data from logs

```
{0: ('COLLAR', 'HANDLE', 'DOLLAR', 'BASKET', '8', 'DOLLAR', 'DOLLAR'),
 1: ('PICKLE', 'PADLOCK', 'CANDY', 'SADDLE', '12', 'CANDY', 'CANDY'),
 2: ('CASTLE', 'DOLPHIN', 'COLLAR', 'DOLLAR', '3', 'COLLAR', 'COLLAR'),
 3: ('PADLOCK', 'SADDLE', 'PADDLE', 'CANDY', '4', 'CANDY', 'CANDY'),
 4: ('DOLPHIN', 'SPEAKER', 'CANDY', 'PADDLE', '12', 'PADDLE', 'PADDLE'),
 5: ('CASTLE', 'BEAKER', 'BEETLE', 'SPEAKER', '1', 'BEAKER', 'BEAKER'),
 6: ('PARROT', 'COLLAR', 'SADDLE', 'PADLOCK', '7', 'SADDLE', 'SADDLE'),
 7: ('BASKET', 'PARROT', 'CASKET', 'SANDAL', '8', 'BASKET', 'BASKET'),
 8: ('DOLPHIN', 'HANDLE', 'CASKET', 'CASTLE', '6', 'CASKET', 'NONE'),
 9: ('CANDLE', 'PICKLE', 'BASKET', 'PICTURE', '6', 'PICKLE', 'NONE'),
  ...
  ...
  ...
35: ('DOLPHIN', 'SANDAL', 'CANDY', 'CANDLE', '4', 'DOLPHIN', 'DOLPHIN')}
```

Logs (recap)

```
START_EXP
START_TRIAL: 0 T: SADDLE.PNG R: PICKLE.PNG B: PADLOCK.PNG L: CANDY.PNG
FIXATE_CENTER_AUDIO_ONSET, COND: 12 TARGET: CANDY
CENTRE_GAZE_START
INSTRUCTION_TO_CLICK_ONSET
LOG_AUDIO_TARGET_START
LOG_AUDIO_TARGET_END
CLICK_RESPONSE_END
FINAL_FIXATION_START, SELECTED: CANDY.PNG
FINAL_FIXATION_END
….
….
….
….
STOP_EXP
```

Relevant data from each trial



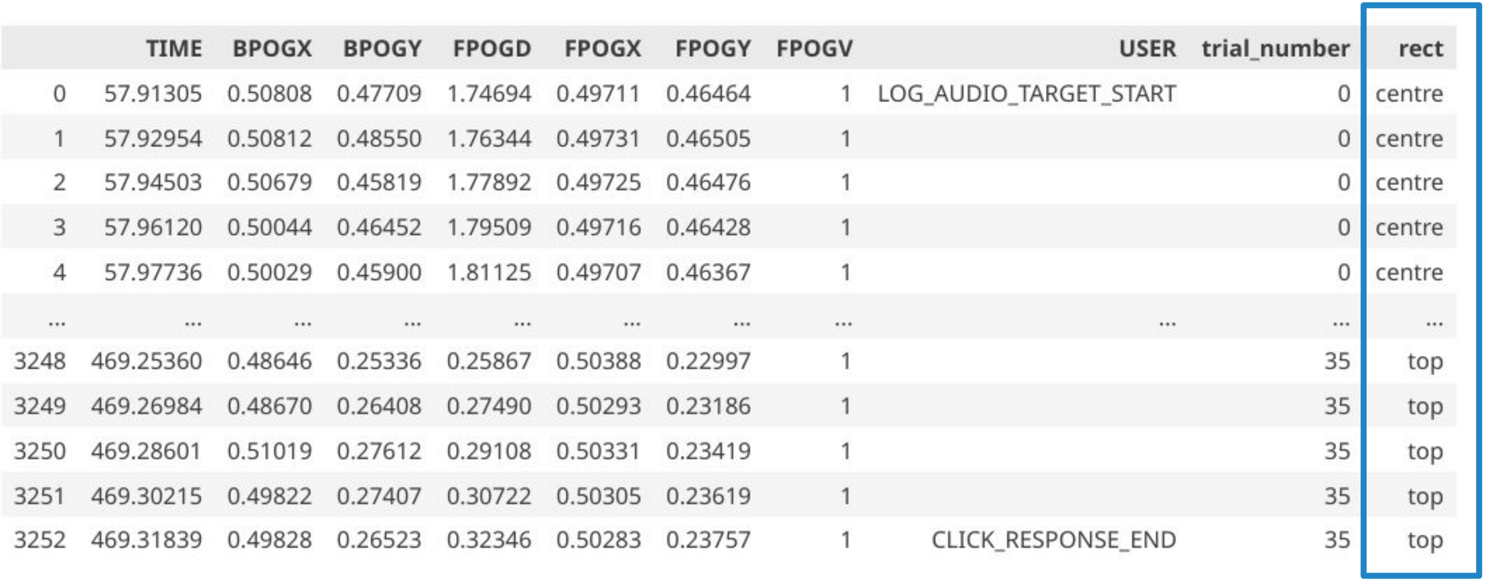
	TIME	BPOGX	BPOGY	FPOGD	FPOGX	FPOGY	FPOGV	USER	trial_number
0	57.91305	0.50808	0.47709	1.74694	0.49711	0.46464	1	LOG_AUDIO_TARGET_START	0
1	57.92954	0.50812	0.48550	1.76344	0.49731	0.46505	1		0
2	57.94503	0.50679	0.45819	1.77892	0.49725	0.46476	1		0
3	57.96120	0.50044	0.46452	1.79509	0.49716	0.46428	1		0
4	57.97736	0.50029	0.45900	1.81125	0.49707	0.46367	1		0
...	...	...	...	...	...	...	...	...	...
3621	469.25360	0.48646	0.25336	0.25867	0.50388	0.22997	1		35
3622	469.26984	0.48670	0.26408	0.27490	0.50293	0.23186	1		35
3623	469.28601	0.51019	0.27612	0.29108	0.50331	0.23419	1		35
3624	469.30215	0.49822	0.27407	0.30722	0.50305	0.23619	1		35
3625	469.31839	0.49828	0.26523	0.32346	0.50283	0.23757	1	CLICK_RESPONSE_END	35

Where? → Which box?

```
# define the bounds of the rectangles
top_rect = [(-128, -416), (128, -128)]
right_rect = [(128, -128), (416, 128)]
bottom_rect = [(-128, 128), (128, 416)]
left_rect = [(-416, -128), (-128, 128)]
centre_rect = [(-128, -128), (128, 128)]
```

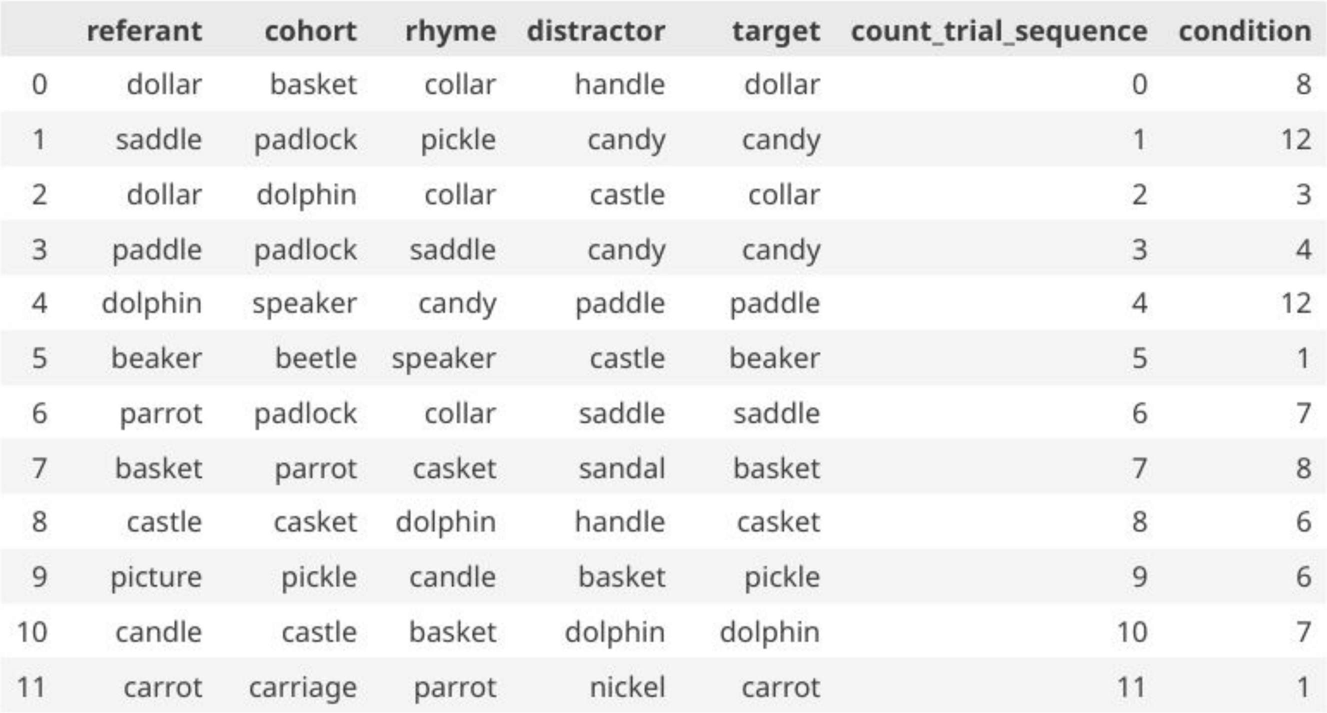
```
def get_rect(x, y):
      if check_if_within_rect(x, y, top_rect):
            return 'top'
      elif check_if_within_rect(x, y, right_rect):
            return 'right'
      elif check_if_within_rect(x, y, bottom_rect):
            return 'bottom'
      elif check_if_within_rect(x, y, left_rect):
            return 'left'
      elif check_if_within_rect(x, y, centre_rect):
            return 'centre'
      Else:
            return 'outside'
```

`rect` column



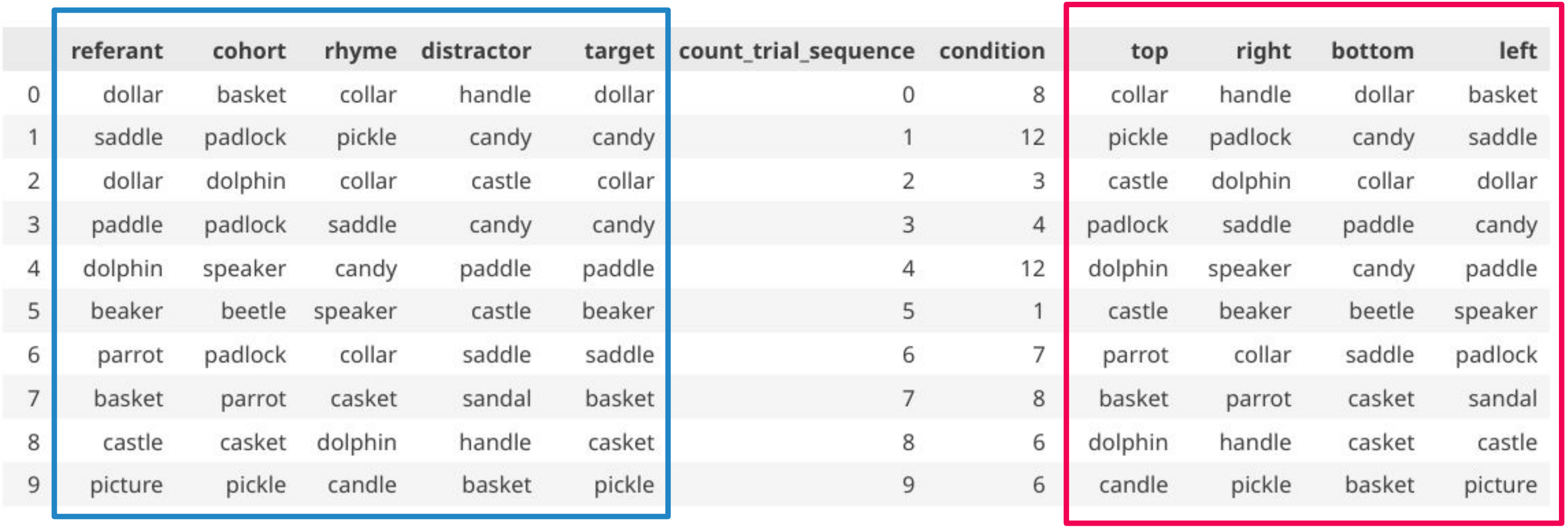
	TIME	BPOGX	BPOGY	FPOGD	FPOGX	FPOGY	FPOGV	USER	trial_number	rect
0	57.91305	0.50808	0.47709	1.74694	0.49711	0.46464	1	LOG_AUDIO_TARGET_START	0	centre
1	57.92954	0.50812	0.48550	1.76344	0.49731	0.46505	1		0	centre
2	57.94503	0.50679	0.45819	1.77892	0.49725	0.46476	1		0	centre
3	57.96120	0.50044	0.46452	1.79509	0.49716	0.46428	1		0	centre
4	57.97736	0.50029	0.45900	1.81125	0.49707	0.46367	1		0	centre
...	...	...	...	...	...	...	...	...	...	...
3248	469.25360	0.48646	0.25336	0.25867	0.50388	0.22997	1		35	top
3249	469.26984	0.48670	0.26408	0.27490	0.50293	0.23186	1		35	top
3250	469.28601	0.51019	0.27612	0.29108	0.50331	0.23419	1		35	top
3251	469.30215	0.49822	0.27407	0.30722	0.50305	0.23619	1		35	top
3252	469.31839	0.49828	0.26523	0.32346	0.50283	0.23757	1	CLICK_RESPONSE_END	35	top

subject-X.csv



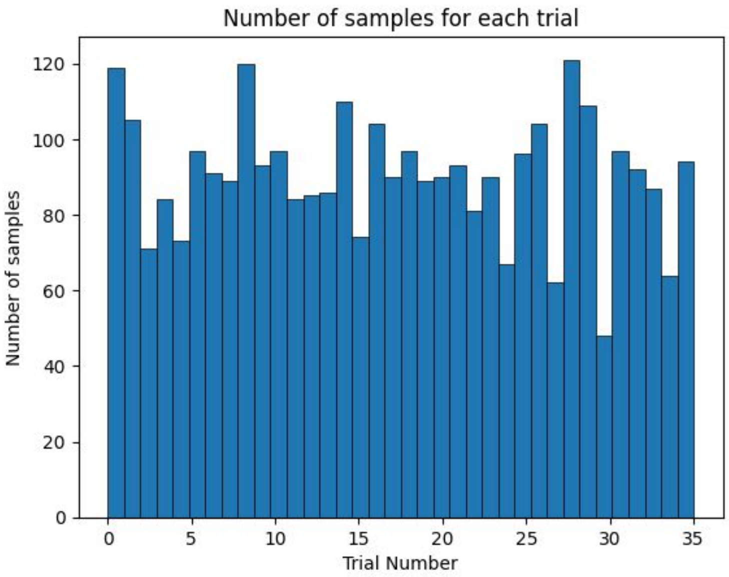
	referant	cohort	rhyme	distractor	target	count_trial_sequence	condition
0	dollar	basket	collar	handle	dollar	0	8
1	saddle	padlock	pickle	candy	candy	1	12
2	dollar	dolphin	collar	castle	collar	2	3
3	paddle	padlock	saddle	candy	candy	3	4
4	dolphin	speaker	candy	paddle	paddle	4	12
5	beaker	beetle	speaker	castle	beaker	5	1
6	parrot	padlock	collar	saddle	saddle	6	7
7	basket	parrot	casket	sandal	basket	7	8
8	castle	casket	dolphin	handle	casket	8	6
9	picture	pickle	candle	basket	pickle	9	6
10	candle	castle	basket	dolphin	dolphin	10	7
11	carrot	carriage	parrot	nickel	carrot	11	1

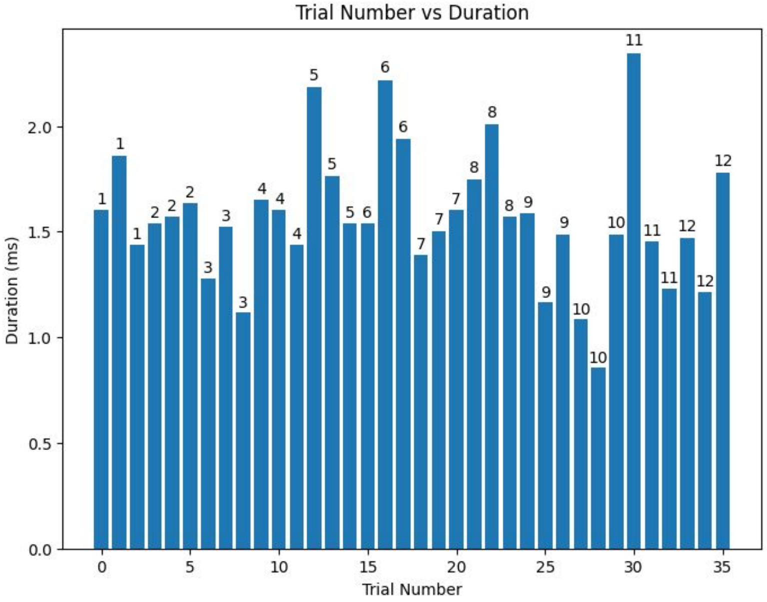
Primary Key! Foreign Key!



	referant	cohort	rhyme	distractor	target	count_trial_sequence	condition	top	right	bottom	left
0	dollar	basket	collar	handle	dollar	0	8	collar	handle	dollar	basket
1	saddle	padlock	pickle	candy	candy	1	12	pickle	padlock	candy	saddle
2	dollar	dolphin	collar	castle	collar	2	3	castle	dolphin	collar	dollar
3	paddle	padlock	saddle	candy	candy	3	4	padlock	saddle	paddle	candy
4	dolphin	speaker	candy	paddle	paddle	4	12	dolphin	speaker	candy	paddle
5	beaker	beetle	speaker	castle	beaker	5	1	castle	beaker	beetle	speaker
6	parrot	padlock	collar	saddle	saddle	6	7	parrot	collar	saddle	padlock
7	basket	parrot	casket	sandal	basket	7	8	basket	parrot	casket	sandal
8	castle	casket	dolphin	handle	casket	8	6	dolphin	handle	casket	castle
9	picture	pickle	candle	basket	pickle	9	6	candle	pickle	basket	picture

The Need for Binning

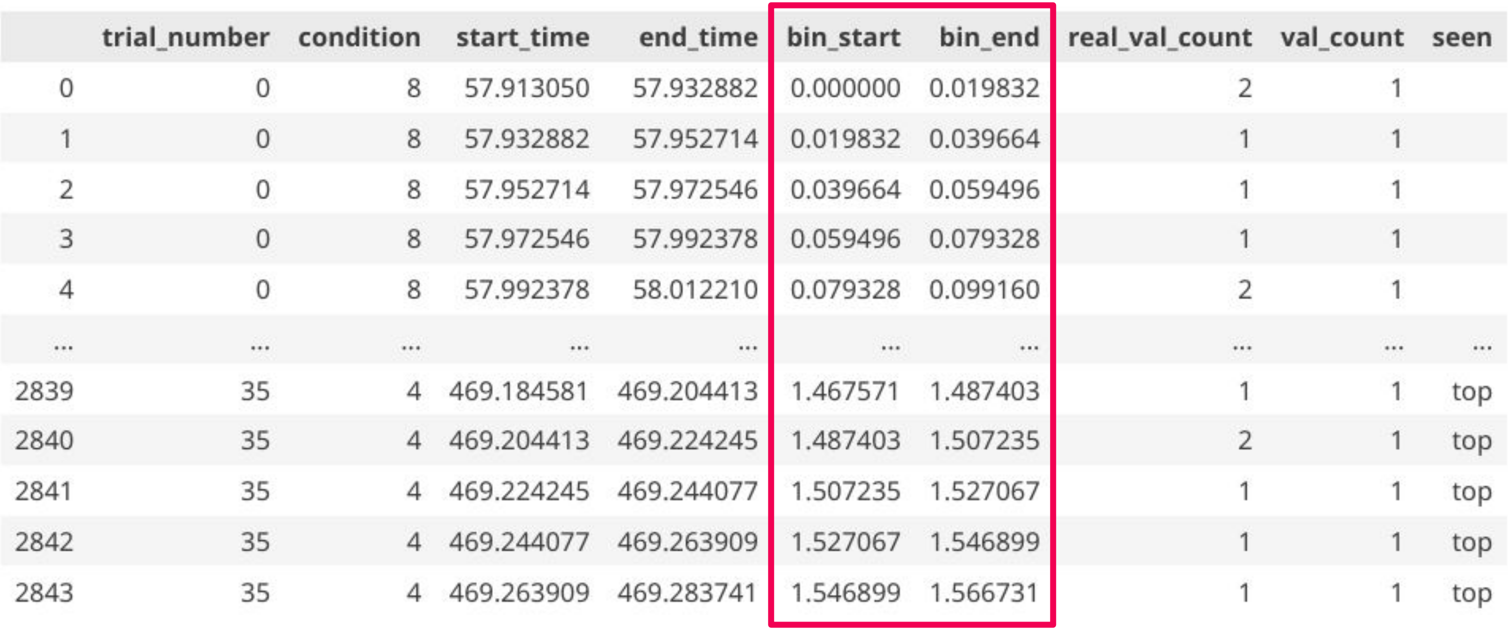




The Need for Binning (contd.)

```
# divide the avg duration into N equal parts
N = 80 # parameter
duration_thresholds = np.linspace(0, avg_duration, N, endpoint=True)
```

The Need for Binning (contd.)



	trial_number	condition	start_time	end_time	bin_start	bin_end	real_val_count	val_count	seen
0	0	8	57.913050	57.932882	0.000000	0.019832	2	1	
1	0	8	57.932882	57.952714	0.019832	0.039664	1	1	
2	0	8	57.952714	57.972546	0.039664	0.059496	1	1	
3	0	8	57.972546	57.992378	0.059496	0.079328	1	1	
4	0	8	57.992378	58.012210	0.079328	0.099160	2	1	
...	...	...	...	...	...	...	...	...	...
2839	35	4	469.184581	469.204413	1.467571	1.487403	1	1	top
2840	35	4	469.204413	469.224245	1.487403	1.507235	2	1	top
2841	35	4	469.224245	469.244077	1.507235	1.527067	1	1	top
2842	35	4	469.244077	469.263909	1.527067	1.546899	1	1	top
2843	35	4	469.263909	469.283741	1.546899	1.566731	1	1	top

Implementing special conditions

Condition No	**Trials (how many examples of such)**	**Conditions**		
		Competitor Set	Target	Distractors
1	3	FC	Re	C, Rh, U
2	3	FC	C	Re, Rh, U
3	3	FC	Rh	Re, C, U
4	3	FC	U	Re, C, Rh
5	3	CC	Re	C, U, U
6	3	CC	C	Re, U, U
7	3	CC	U	Re, C, U
8	3	RC	Re	Rh, U, U
9	3	RC	Rh	Re, U, U
10	3	RC	U	Re, Rh, U
11	3	UC	Re	U, U, U
12	3	UC	U	Re, U, U

Implementing special conditions (contd.)

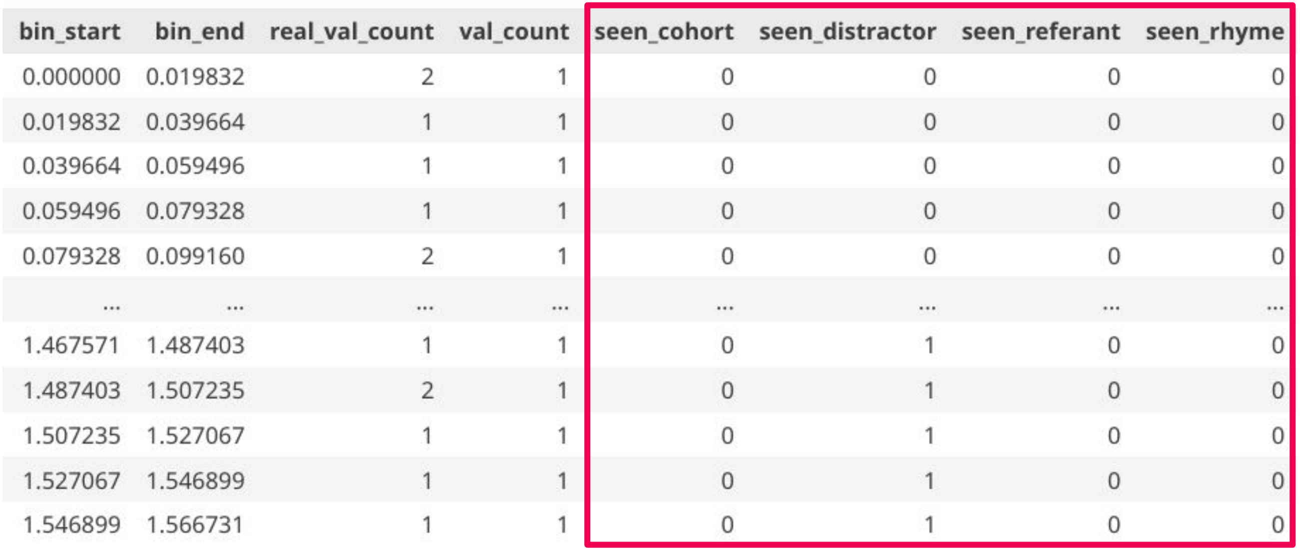
```
full_competitor_sets_cond = [1, 2, 3, 4]
cohort_competitor_sets_cond = [5, 6, 7]
rhyme_competitor_sets_cond = [8, 9, 10]
distractor_competitor_sets_cond = [11, 12]
```

```
# for trials with condition numbers in cohort_competitor_sets_cond, replace 'rhyme' with 'distractor'
count_df.loc[count_df['condition'].isin(cohort_competitor_sets_cond), 'seen'] = count_df['seen'].apply(lambda
x: 'distractor' if x == 'rhyme' else x)
```

- Similarly, for rhyme competitors sets replace cohort with distractor
- Replace both cohort and rhyme with distractor for the distractor competitor sets

Counting fixations (on each stimuli type)

```
# one hot encode the 'seen' column
one_hot_count_df = pd.get_dummies(count_df, columns=['seen'])
```



bin_start	bin_end	real_val_count	val_count	seen_cohort	seen_distractor	seen_referant	seen_rhyme
0.000000	0.019832	2	1	0	0	0	0
0.019832	0.039664	1	1	0	0	0	0
0.039664	0.059496	1	1	0	0	0	0
0.059496	0.079328	1	1	0	0	0	0
0.079328	0.099160	2	1	0	0	0	0
...	...	...	...	...	...	...	...
1.467571	1.487403	1	1	0	1	0	0
1.487403	1.507235	2	1	0	1	0	0
1.507235	1.527067	1	1	0	1	0	0
1.527067	1.546899	1	1	0	1	0	0
1.546899	1.566731	1	1	0	1	0	0

A Few Final Steps

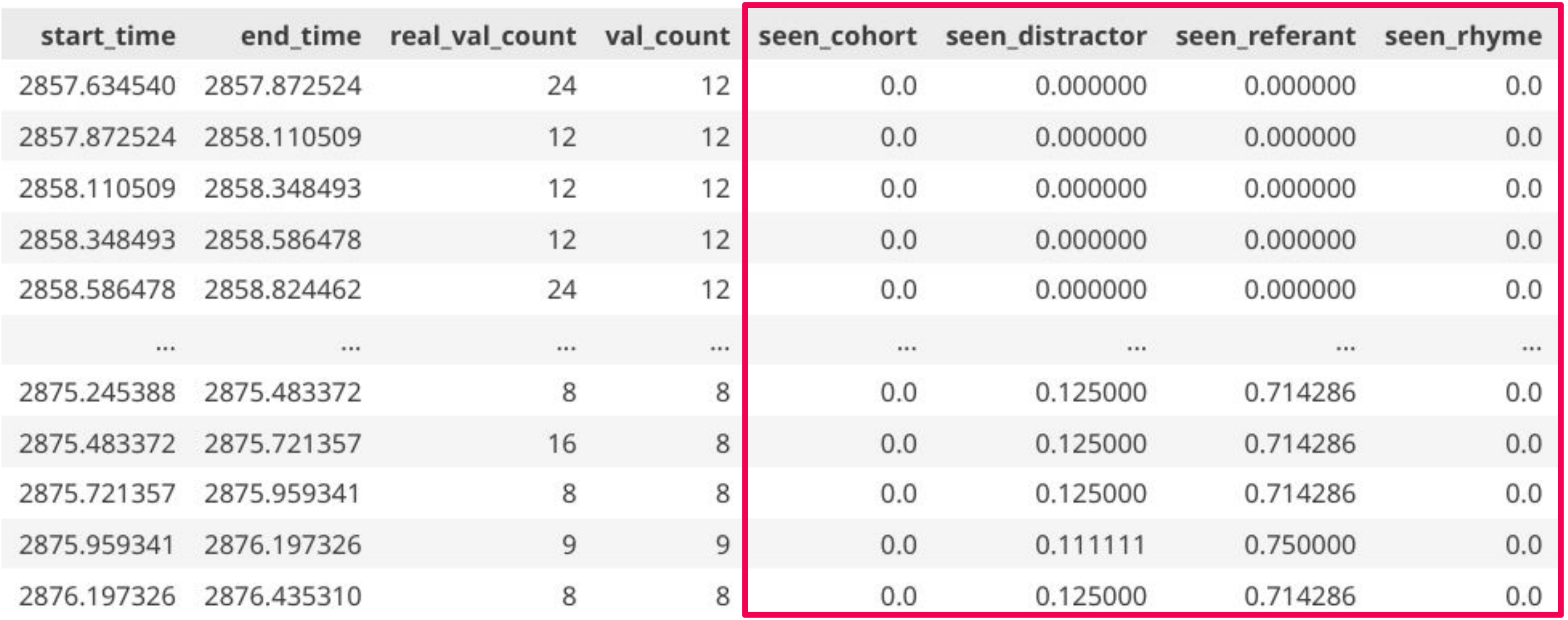
- Group by the columns `bin_start` and `bin_end` across all trials and calculate the sum of fixations

```
groupby_time_bins_df = one_hot_count_df.groupby(['bin_start', 'bin_end']).sum().reset_index()
```

- Calculate the fixation probabilities by dividing by the `val_count` column

```
# ignore rows where val_count is 0
groupby_time_bins_df['seen_referent'] = groupby_time_bins_df.apply(
      lambda x: x['seen_referent'] / x ['value_count']
      if x['value_count'] != 0 else 0, axis=1
      )
```

A Few Final Steps (contd.)

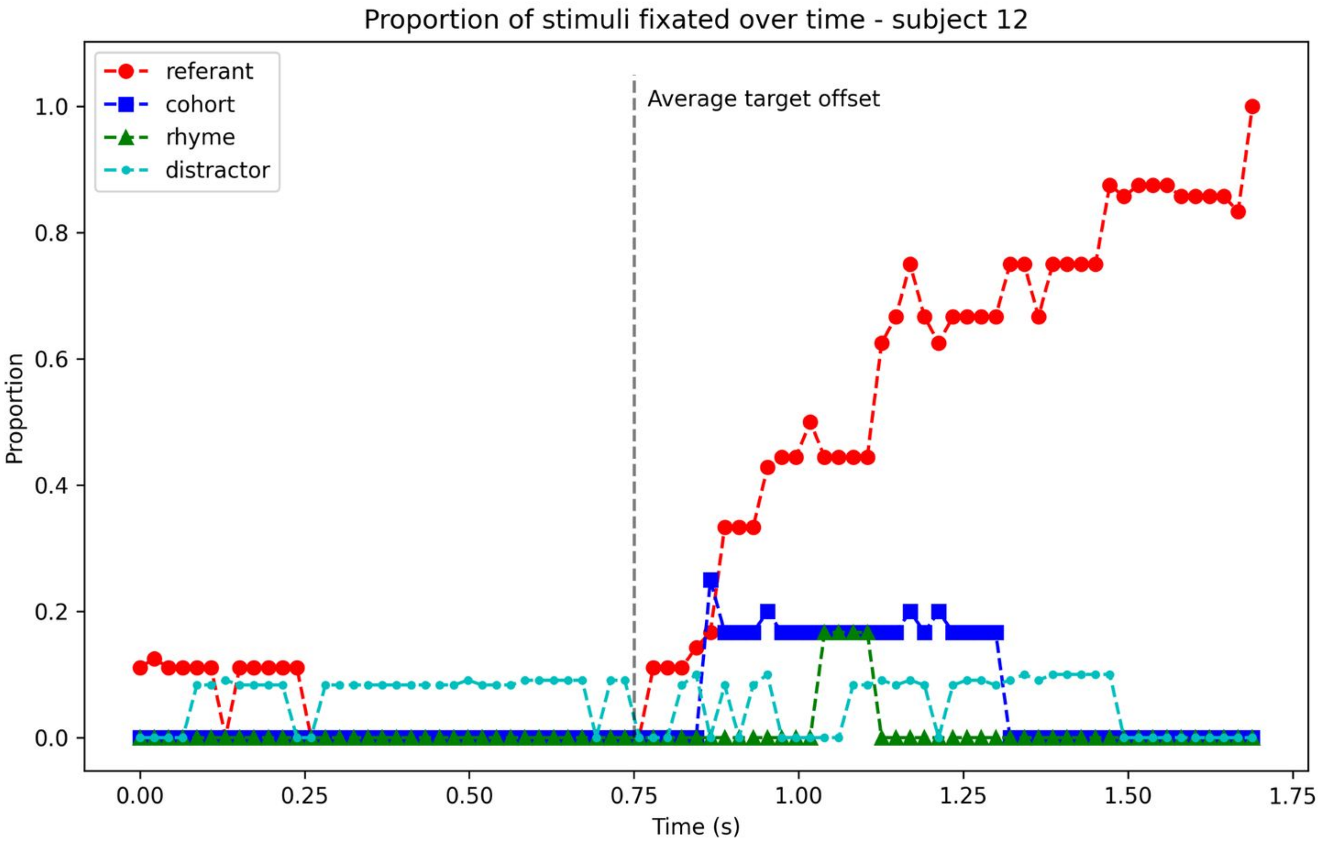


start_time	end_time	real_val_count	val_count	seen_cohort	seen_distractor	seen_referant	seen_rhyme
2857.634540	2857.872524	24	12	0.0	0.000000	0.000000	0.0
2857.872524	2858.110509	12	12	0.0	0.000000	0.000000	0.0
2858.110509	2858.348493	12	12	0.0	0.000000	0.000000	0.0
2858.348493	2858.586478	12	12	0.0	0.000000	0.000000	0.0
2858.586478	2858.824462	24	12	0.0	0.000000	0.000000	0.0
...	...	...	...	...	...	...	...
2875.245388	2875.483372	8	8	0.0	0.125000	0.714286	0.0
2875.483372	2875.721357	16	8	0.0	0.125000	0.714286	0.0
2875.721357	2875.959341	8	8	0.0	0.125000	0.714286	0.0
2875.959341	2876.197326	9	9	0.0	0.111111	0.750000	0.0
2876.197326	2876.435310	8	8	0.0	0.125000	0.714286	0.0

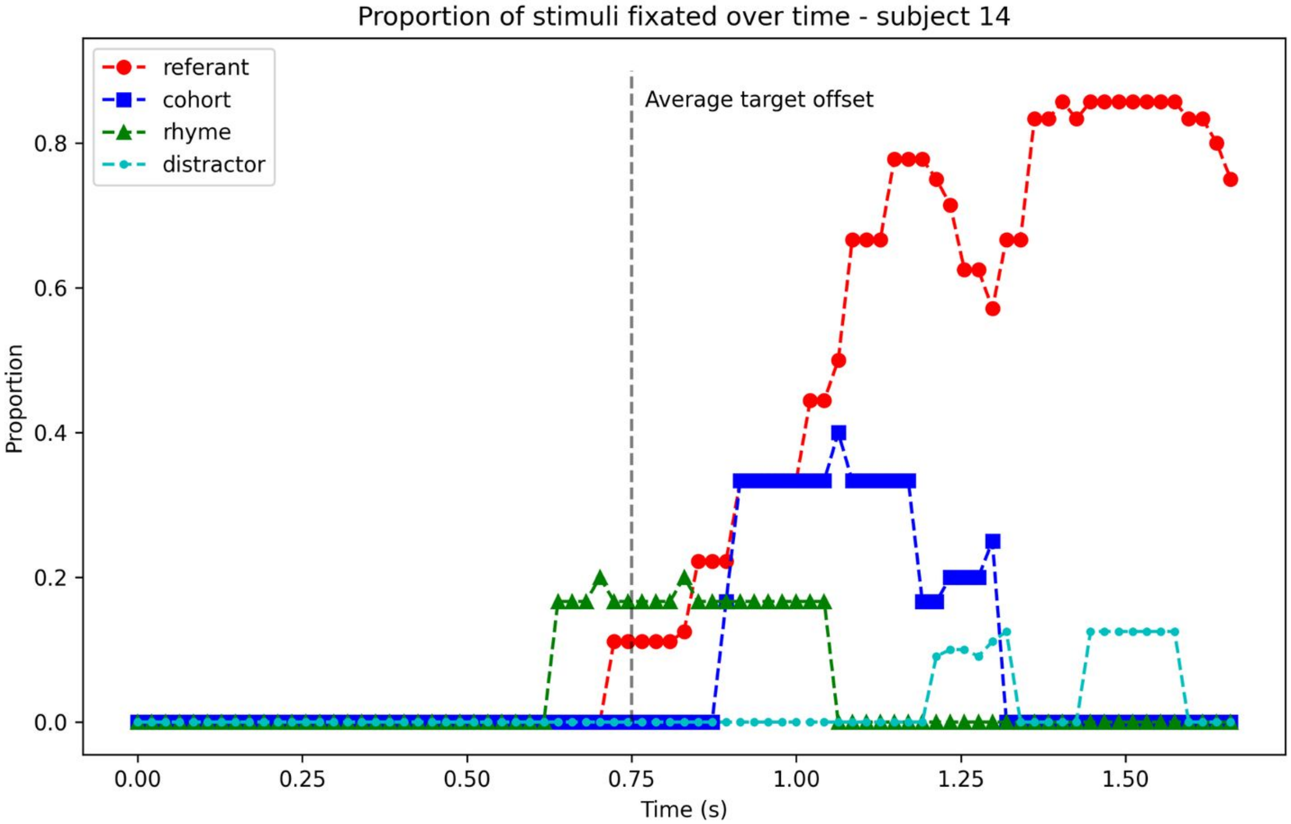
A Few Final Steps (contd.)

- We plot the proportion of fixations graph for the trials where the referent is the target.
  - Fixations to the referent were averaged across the full competitor trials, the cohort-only and rhyme-only competitor trials.
  - Fixations to the cohort objects were averaged across the full competitor and cohort-only conditions and fixations to the rhyme were averaged over the full competitor and rhyme-only trials.

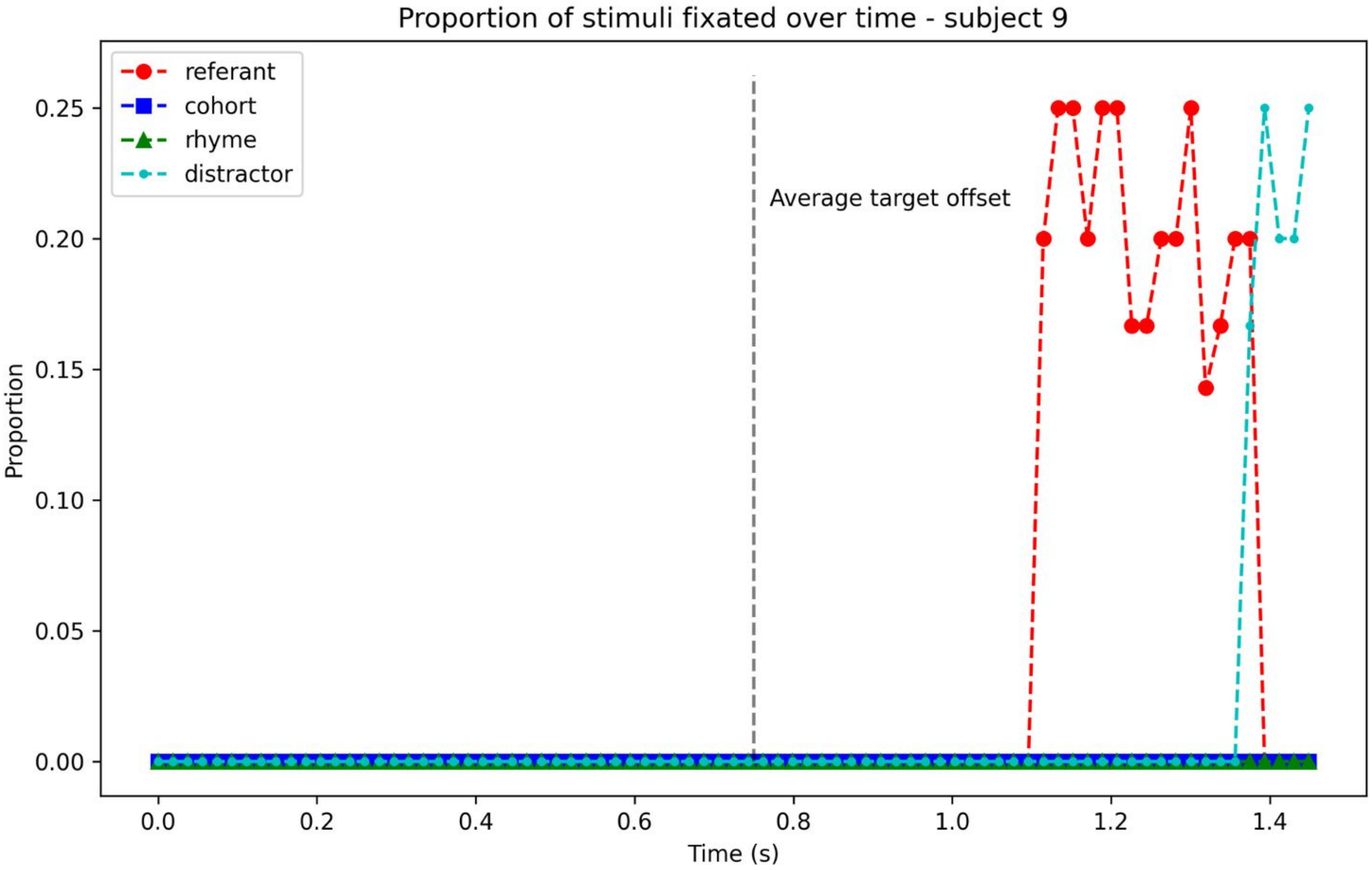
Analysis plots (individual subjects)



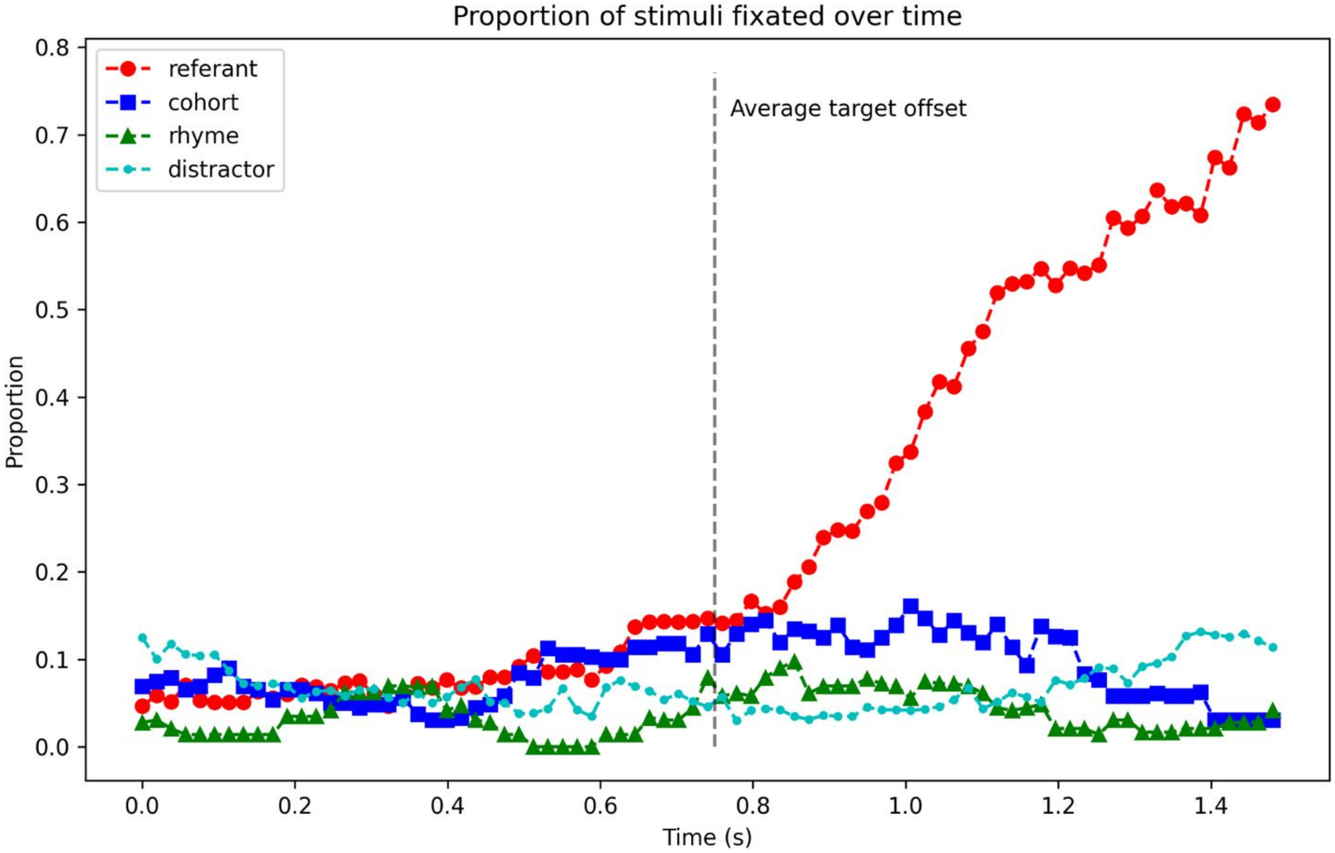
Analysis plots (individual subjects)



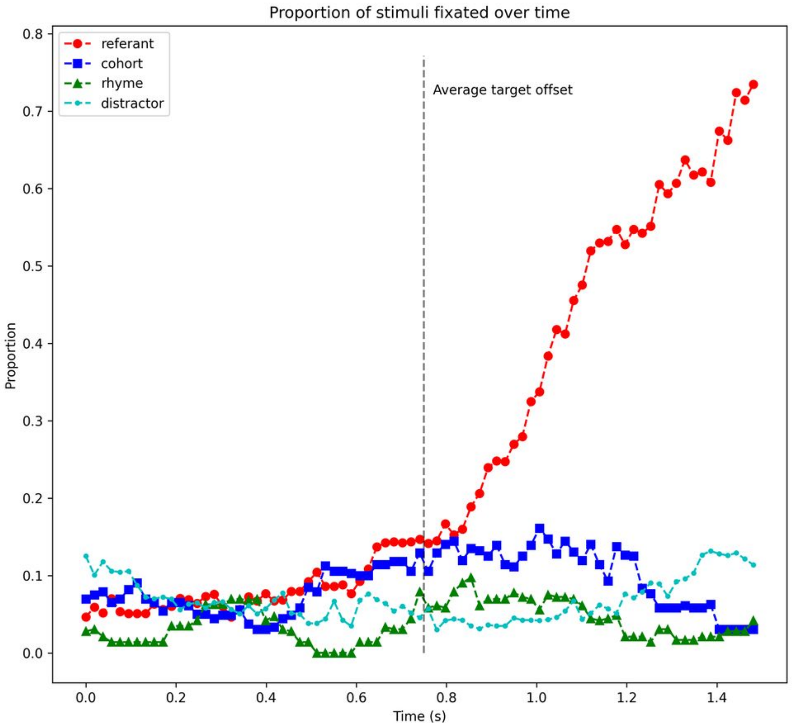
Analysis plots (individual subjects)

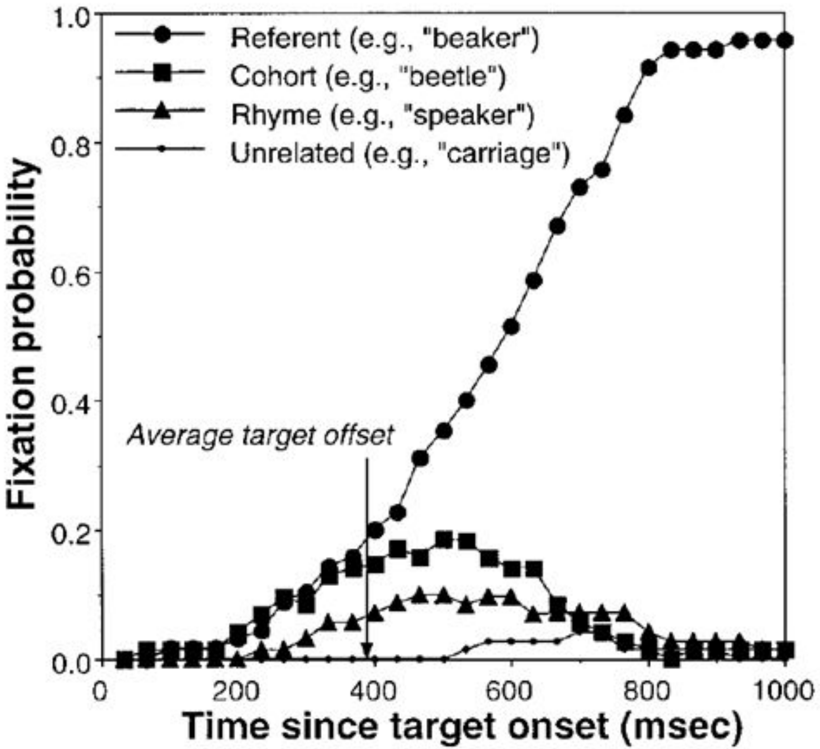


Analysis plots (overall)



Analysis plots (comparison)





Inference

- For the 0 to ~400 ms interval, the fixation proportions appear to be random. No specific trend is observed at this stage.
- In the second interval (400–800 ms), there is a trend toward more fixations to the referent and cohort items compared to the other stimuli.
- The line for the referent fixations separates from the cohort in the 800 to 1000 ms interval.
- Starting with 600 to 700 ms window, there is an increase in the number of fixations to the rhyme.

Inference (contd.)

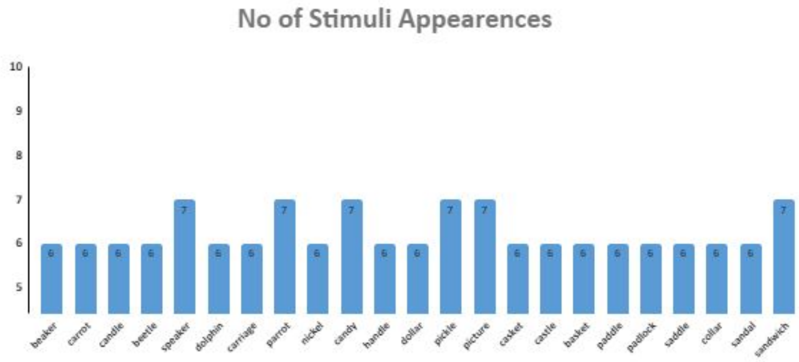
- Beyond the 1000 ms mark, the referent line starts to peak leaving the other competitors behind
- One major deviation (from source) in our study is that the distractor stimuli has small peak towards the end.

What we didn’t replicate?

- Average duration of auditory stimulus: 375 ms vs 750 ms
- Number of trials for each competitor set. Participants get tired after ~30 trials.
- Audio stimuli were digital (instead of analog)
- Use of a 3x3 grid instead of 5x5
- Participants respond with mouse clicks instead of drag-and-drop.
- No calibration functionality after each trial

Challenges Faced

- Balancing of trials
  - Generation of balanced trial conditions using python scripts
    - Unbalanced results
  - Reverted to manual creation



Challenges Faced

- Random ‘freezes’ during experiment
    - Issue with ‘fixation_check’?
        - While True loop

          ```
          while True:
                gazepos = eyetracker.sample()
          ```

        - Periodic sampling

          ```
          while True:
               if clock.time() - check_timer > diff:
                      gazepos = eyetracker.sample()
          ```

    - Switching backend to PsychoPy
        - Required the sample rate of all audio samples to be the same
    - Removal of gaze contingent features

Shortcomings of the study

- Since we used a relatively small set of pictures, participants might have become aware of the similarity among the referent–cohort–rhyme sets despite the large number of filler trials[†].
- Generalizability is affected as our study participants only include university students of a specific age group, which does not fully represent the complexities and variations of real-world spoken word recognition scenarios.
- The study also may not fully address the universality of the observed effects across different languages as the original study as well as our replication is in English.

† : from reference paper

Conclusions

- Eye movement tracking is a reliable tool for investigating the time course of spoken word recognition and capturing the mapping process while the spoken word unfolds.
- The results and plots obtained by us provide an empirical support for continuous and incremental mapping models of word recognition and not a discrete and all-or-nothing process.

Conclusions (contd.)

- Our results suggest that as the spoken word unfolds over time, the listener gradually narrows down the set of candidate words based on the contextual information. This competition among candidate words occurs until a single word is identified or a clear winner emerges.

Thank you!