Comparing Readability and Eye Movement Patterns Between Uppercase and Lowercase Fonts in English: An Eye-Tracking Study

Sveva Battisti (3756089)   Raheleh Soltani (3635045)
Aishwarya Pandurang Chaudhari(3643749)

August 14, 2024

1 Abstract:

This study investigates the readability of uppercase versus lowercase fonts in English sentences through an eye-tracking experiment. We aimed to determine whether uppercase text is more difficult to process than lowercase text by L2 speakers of English. Twelve participants were tested on various sentences presented in both uppercase and lowercase fonts. Key metrics, including the number of fixations, fixation duration, and landing spot, were recorded and analysed. Our results provide insights into how font case affects reading efficiency and eye movement patterns, contributing to the broader understanding of typographic design and readability. Our final results indicated indistinct patterns of behavior between the two font cases. These findings align closely with those of (White and Liversedge [6]), who observed no significant difference in the readability between the two font formats.

Table of contents

2 Introduction

2.1 Background

Eye tracking technology has a wide range of applications in research on visual system, psychology, psycholinguistics and human-computer interaction, among other fields. Slattery (2016) explores the interplay between eye movements and various aspects of visual processing, including the implications for both psycholinguistic research and practical applications in font design. This study highlights how different font characteristics, such as size and style, can significantly impact reading efficiency and comprehension, maintaining that certain fonts are better suited for facilitating smooth and efficient reading, thereby reducing cognitive load. Different types of eye trackers are available, varying in precision and suitability depending on the specific goals of the experiment. These tools, coupled with advanced data processing techniques, allow researchers to gather detailed information about visual and cognitive processes. In the field of linguistics, eye tracking has become increasingly common as it provides a precise method for determining where attentional resources are allocated during language processing tasks. By directly measuring cognitive processing effort, eye tracking helps researchers understand how the brain engages with and interprets language. This study focuses on the readability of uppercase versus lowercase fonts in English sentences, using eye tracking to assess whether uppercase text presents more challenges than lowercase text. The motivation for our study is deeply rooted in the seminal findings of Tinker and Paterson (1939). Their pioneering research, utilizing the photographic technique, revealed that reading uppercase text resulted in a 7% increase in reading time compared to lowercase text, accompanied by a greater number of fixations. Intriguingly, despite the increased number of fixations, the duration of these fixations was 20 milliseconds shorter for uppercase text. In contrast, a later study by White and Liversedge (2006), which employed a different experimental design and aimed to explore the topic from another angle, observed a slight decrease in reading time for uppercase text compared to lowercase, amounting to a mere 2%. Their findings indicated no significant difference in the number of fixations between the two font cases. These contrasting results underscore the complexity of reading processes and the influence of text case on reading dynamics, providing a rich foundation for our current investigation. Additionally, White and Liversedge's findings on the landing spot revealed that readers tended to land closer to the beginning of words when reading uppercase text. This particular insight has become a cornerstone for our investigation, guiding our primary objective to explore how text case influences reading behaviour. While Tinker and Paterson's study definitely demonstrates that case type significantly impacts reading behaviour, with uppercase text proving more difficult to read than lowercase, the findings of White and Liversedge presented a different narrative. Their study indicated only limited difference in eye movement behaviour when participants read uppercase text compared to lowercase. This contrast in results highlights the nuanced and complex nature of how text case influences reading dynamics, further justifying our current exploration into this intriguing area. In the same vein, we aim to explore how typographic variations impact reading efficiency by analysing key metrics such as the number of fixations, fixation duration, and landing spots. Understanding these factors contributes to our knowledge of effective typographic design and its implications for

readability. The experiment involves twelve participants who read sentences presented in both uppercase and lowercase fonts. Through this investigation, we hope to shed light on the cognitive and visual mechanisms underlying text processing, ultimately informing better design choices for written materials. One important factor in understanding comprehension and readability of fonts is the preferred location on words where the eyes fixate. [2] McConkie et al. (1988) propose five principles of perceptuo-oculomotor control to explain why the landing spot is often to the left side of the word's centre. This specific location appears to facilitate easier word comprehension. According to Kerr and Zola, readers develop the ability to adjust their gaze to position their eyes optimally. The convenient viewing hypothesis suggests that forward saccades can be influenced by the position of the eyes relative to the word. This highlights the word-based nature of eye guidance. Essentially, readers fine-tune their oculomotor behaviour to achieve optimal performance, taking into account the constraints of the visual system and the properties of the text stimuli. Results from [3] O'Regan & Jacobs (1992) showed distance of landing spot from optimal position, which they believe to be near the middle or just left of the middle of words, does impact lexical decision time. In the same vein, it is logical to consider how differences in font case, and consequently the landing spot, influence readability. The authors suggest a connection between the strength of optimal viewing point and the hypothesis that there is a fall-off in acuity within the fovea. It can be argued that the inconsistency of the landing spot in lower and uppercase fonts might be related to peripheral, rather than foveal, acuity in uppercase font.

2.2 Research question/hypothesis

In alignment with the results of White and Liversedge, the final hypothesis formulated predicts no significant difference in reading behaviour between uppercase and lowercase text. We aim to confirm their findings, suggesting that the case of the text does not markedly affect eye movement and overall reading dynamic.

3 Experimental Methods

3.1 Participants

Twelve L2 (second language) speakers of English participated in this experiment, chosen due to the better availability of L2 speakers in Germany compared to native speakers, and to compare their results with those of native speakers. English was their second language with proficiency of A2 or higher. They come from different language backgrounds, such as German, Azeri, Hindu, Italian etc. The participants age was between 20-30. Each participant was provided with two easy-to-comprehend dialogues, presented either in uppercase or lowercase.

3.2 Stimuli

The dialogues were displayed to each participant in either uppercase or lowercase format through four loops generated in Open Sesame. The sentences of each dialogues appeared one at a time on the screen, and participants could proceed to the next sentence by pressing the space key. Before each line of the dialogue, a fixation dot appeared on a sketch pad on the left side of the screen, cantered vertically. After each sentence, a fixation dot appeared on the right side of the screen. Fixation dots disappeared upon gaze fixation of the participant, and they were then able to observe the next pad. Each trial, or dialogue, lasted about 5 minutes.

Participants were asked to answer two comprehension questions after reading each dialogue. They responded by pressing the up key for 'yes' and the down key for 'no'. A logger recorded the onset and another one was used for the end of each dialogue to track reading behaviours and responses.

Each participant read each dialogue either in uppercase or lowercase format, facilitated by the four loops in Open Sesame. The design ensured that the presentation format varied within participants to control for individual reading speed and comprehension differences. This way the dialogues were both counterbalanced and randomized in terms of case.

3.3 Software and Hardware

- **Software**: The experiment is designed in OpenSesame[**mathot_opensesame**] which is a graphical experiment building software to create experiments for psychology, neuroscience, and experimental economics. Eye trackers can be integrated with OpenSesame to record the eye movements of the participants, which is finally used to analyze the data. It is available on Windows, Mac OS and Linux.

- **Language**: Python was used along with the OpenSesame to create the experiment. Libraries like Pandas, Numpy, Matplotlib were used to analyze the data. Psychopy is used in Open Sesame. Other options available for backends are pyGame, Expyriment, etc.

- **Eye Tracker**: The experiment is conducted using the GazePoint GP3 eye tracker. It is a binocular eye tracker that can record at 150 Hz. The eye tracker is connected to the computer and the participants are seated at a distance of around 60 cm from the screen. The experiment was conducted in a dimly lit laboratory setup to avoid any external light source that might interfere with the eye tracking. The GazePoint API [**gazepoint_api**] is referred for the anaylsis of the eye tracking data.

3.4 Eye Tracker Calibration

The GazePoint GP3 eye tracker requires accurate calibration for reliable data. Calibration was conducted before each participant's session, adjusting for factors such as glasses, contact lenses, and individual participant differences. To ensure data quality, calibration was repeated if the

tracker indicated significant deviations. For participants unable to calibrate successfully, additional measures included adjusting the tracker's settings and providing multiple calibration attempts.

Eye tracker calibration:

- Challenge: The eye tracker calibration was a challenge as the participants were not able to calibrate with the eye tracker, due to many reasons like contact lenses, glasses, height of the participants, body posture, etc., during the experiment.

- Solution: The number of participants were increased to more than 10 to ensure that we have at least 9-10 participants with proper calibration and the experiment was run a few times before the participant starts.

4 Experiment

4.1 Overview

The main idea of this experiment was to investigate whether font types impact eye movement patterns, such as fixation and duration, and consequently affect overall readability. We primarily came up with the idea that font types that promote clear letter shapes and spacing would reduce the occurrence of fixating eye movements, thus enhancing readability.

To test this, we used font format (uppercase and lowercase) as independent variables, and fixation duration, comprehension, and regressive eye movements (number and duration) as the dependent variables. We also decided to measure the landing spot, which we reckoned to serve as a benchmark for measuring reading time.

The conditions were tested using within-subjects paradigm, meaning each participant was exposed to both uppercase and lowercase formats. We set up the experiment in Open Sesame, which, through iterative development over the semester, was refined to better meet our goals.

In the experiment, participants first saw a fixation dot, followed by sentences of a dialogue presented one at a time, and then comprehension questions. We collected data on the number of fixations, fixation durations, and the coordinates of each word on the screen to create bounding boxes for each word. Employing these bounding boxes, showing the position of each word and the sentence, facilitated by a monospaced font, we could accurately measure the landing site.

Using the pilot data, we generated some graphs for sanity check. One of the graphs we used plotted time against x-position, with added vertical lines to check the triggers/user logs. This allowed us to determine how engaged the participant was throughout the experiment and to identify specific points of interest or disengagement.

Since we believed that if one type of font case might requires more fixations and longer fixation durations to read, we visualized both the number of fixations and the fixation durations. The

rationale for analysing fixation durations is that the length of fixations might be influenced by the distinct character shapes in the two font cases.

Despite observing more fixations in lowercase segment, the fixation durations for the two subjects in our preliminary experiment did not show much difference between the two font cases. The variations in landing spots further suggest that initial eye movements differ between uppercase and lowercase text. This indicates that different font types influence how eyes are moved during reading, leading to various processing behaviours.
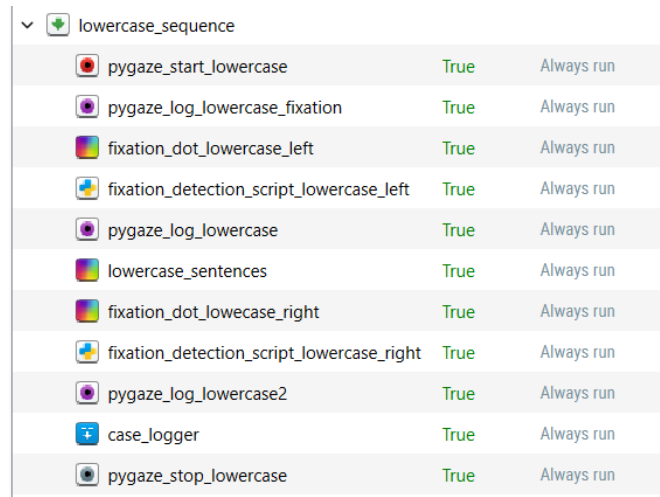
These opposing results motivated us to strengthen our findings by running the experiment on more subjects.

4.2 Design of the Experiment

The experiment (Figure 1) follows the following structure in OpenSesame:

1. **Introduction to the experiment**: It contains some preliminary instructions for the participants to understand the experiment. It also mentions that each progression will require a key press. The foreground color is set to black and the background color is set to white throughout the experiment.

2. **Initialization of variables**: The position variables (center) is initialized and is used to set the position of the sentence in the grid. The `pygaze` module is also intialized to record the eye movements of the participants.

3. **Trial Loop Items**: This loop runs the experiment for 2 trials. The trial loop contains the following sequence of events:

   - **First Fixation Dot**: A fixation cross is displayed at the left center of the screen. It is a black dot on a white background which is displayed to ensure that the participants are looking at the centeral left side of the screen before the sentence appears. It is displayed using the sketchpad item in OpenSesame.

   - **Stimulus**: The stimuli that is a sentence, either in lowercase or uppercase, will appear. Participants will see one sentence at a time. There are two dialogues of analogous difficulty and the same number of words in each line, i.e. 10. Each of the two dialogues will be seen in either uppercase and lowercase. Each participant will see two dialogues. Each response is captured with a keypress. The keypress is recorded and logged using the `keypress_response` item in OpenSesame.

   - **Second Fixation Dot**: A second fixation dot appears after each sentence on a new sketchpad. It is a black dot located in the center and right side of the screen.

   - **Logging**: The onset and offset of the fixation instruction and the stimulus (sentence) are logged for each trial. Additionally, there is loggings for the fixation dot, case type, and prior and post eye-tracker loggings.

4. **End of Experiment**: The experiment ends with a thank you message for the participants. Below an image of the trial sequence is observable.

Figure 1: Trial Timeline

4.3 Logic of the experiment

1. Stimuli are chosen as per the discussions and consultations throughout the semester. It is worth mentioning that the reason we chose dialogues over individual sentences was to make the experiment more engaging for the participants, encouraging them to continue reading.

   - For e.g., One criteria of choosing these senetences is equality in the length of each sentence and degree of difficulty of dialogues. The two dialogues are as follows:

Dialogue in lowercase hi, sarah! how was your weekend? what did you do? good! how was your weekend? did you hike with friends? yes, we went to the nearby mountain trail. and you? i have been wanting to go there some time soon. you should go there. it is a really nice trail.

Dialogue in lowercase hey, lisa! did you finish the book i lent you?
hi, john! yes, i did it. it was really amazing! i'm glad you liked it. what was your favourite part? i loved the ending scene but it was so unexpected! that twist at the end scenario was brilliant, wasn't it?

Dialogue in uppercase HEY, LISA! DID YOU FINISH THE BOOK I LENT YOU?
HI, JOHN! YES, I DID IT. IT WAS REALLY AMAZING!
I'M GLAD YOU LIKED IT. WHAT WAS YOUR FAVOURITE PART?
I LOVED THE ENDING SCENE BUT IT WAS SO UNEXPECTED!
THAT TWIST AT THE END SCENARIO WAS BRILLIANT, WASN'T IT?

Dialogue in uppercase HI, SARAH! HOW WAS YOUR WEEKEND? WHAT DID YOU DO? GOOD! HOW WAS YOUR WEEKEND? DID YOU HIKE WITH YOUR FRIENDS? YES, WE WENT TO THE NEARBY MOUNTAIN TRAIL. AND YOU? I HAVE BEEN WANTING TO GO THERE SOME TIME SOON. YOU SHOULD GO THERE. IT IS A REALLY NICE TRAIL.

2. Fixations, at the centre left side before the sentence and at the center right side after the sentence, on the screen mark the start and end of each sentence as stimuli units. It will also helps the participants anticipate/expect the next sentence and be ready to cognitively relate and comprehend them.

5 Analysis

5.1 Stimuli Preprocessing

The stimuli needed to be randomized, in loops, and preprocessed before they could be employed in the experiment. The preprocessing steps are as follows:

1. Definition of the stimuli appearance on a sketchpad with sentence as a unit. The two dialogues are randomized in four loops so that both are seen in both type cases, upper and lower.

2. Left side fixation dot had to be programmed using the following code (Figure 2).

```
 1 import math
 2 # Get the start time
 3 t0 = time.time()
 4 # Compute the left center coordinates
 5 screen_width = 1920
 6 screen_height = 1080
 7 left_centre = (screen_width / 4, screen_height / 2)  # Left center of the screen
 8
 9 central_fixation = False
10 while not central_fixation:
11     # Get a current (x,y) sample from the tracker
12     gazepos = eyetracker.sample()  # Replace with the actual method to get gaze position
13     # Check distance between the sample and the left center
14     d = math.sqrt((gazepos[0] - left_centre[0])**2 + (gazepos[1] - left_centre[1])**2)
15     # Check if distance is "small enough"
16     if d < 100:
17         central_fixation = True
18     # Check for a timeout (10 seconds)
19     if time.time() - t0 > 10000:  # 10 seconds
20         break
21
22
```

Figure 2: Left fixation-dot script

3. Using math and time libraries, the right side fixation dot was set (Figure 3).

```
 1 import math
 2 import time
 3 # Function to get the gaze position from the eye-tracker
 4 def get_gaze_position():
 5     # Replace with the actual method to get the current gaze position from your eye-tracker
 6     return eyetracker.sample()
 7 # Get the start time
 8 t0 = time.time()
 9 # Compute the right center coordinates
10 screen_width = 1920
11 screen_height = 1080
12 right_centre = (3 * screen_width / 4, screen_height / 2)  # Right center of the screen
13 central_fixation = False
14 while not central_fixation:
15     # Get a current (x, y) sample from the tracker
16     gazepos = get_gaze_position()  # Replace with the actual method to get gaze position
17     # Check distance between the sample and the right center
18     d = math.sqrt((gazepos[0] - right_centre[0])**2 + (gazepos[1] - right_centre[1])**2)
19     # Check if distance is "small enough"
20     if d < 100:
21         central_fixation = True
22     # Check for a timeout (10 seconds)
23     if time.time() - t0 > 10:  # 10 seconds
24         break
```

Figure 3: Right fixation-dot script

The functions are employed to detect the gaze before the sentence appears preventing any odds of outliers in fixations. Below are the images of how it looks like on Open Sesame (fig. 4,5,6):

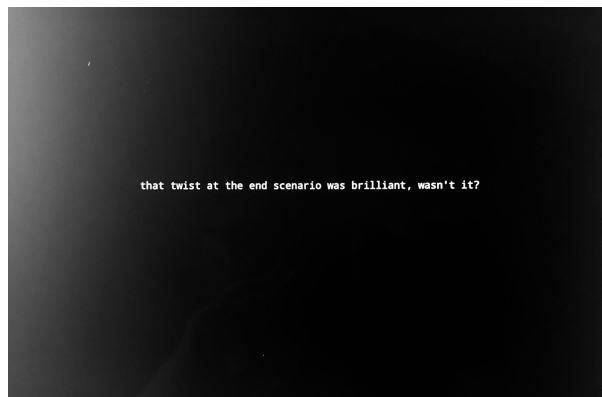Figure 4: Description for Fixation Dot Left
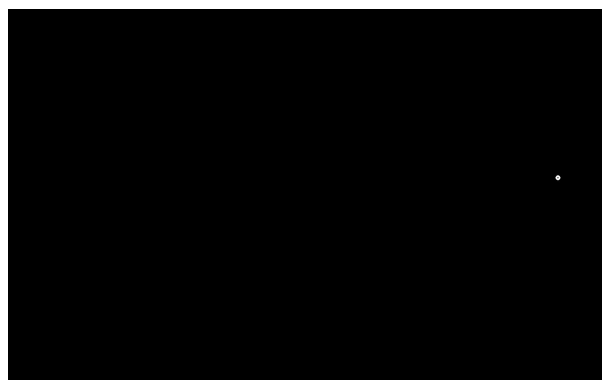

Figure 5: Description for Stimuli Sentence

Figure 6: Description for Fixation Dot Right

5.2 Sanity check

Sanity checks are crucial in an eye-tracking experiment to ensure data quality and reliability. These checks help identify and correct potential issues that may compromise the validity of the experiment. For our experiment, the sanity checks were designed in line with the metrics we needed to analyze later. Specifically, our sanity checks included verifying the presence of logs/triggers and checking the FPOGX and FPOGY values. Logs were a fundamental part of our experiment because they allowed us to determine the beginning and end of each case type dialogue. In the OpenSesame experiment, we inserted the following logs:

- SENTENCE_UPPERCASE_ONSET
- SENTENCE_LOWERCASE_ONSET
- SENTENCE_LOWERCASE_OFFSET_BOOKS
- SENTENCE_LOWERCASE_OFFSET_WEEKEND
- SENTENCE_UPPERCASE_OFFSET_BOOKS
- SENTENCE_UPPERCASE_OFFSET_WEEKEND

We created two versions of each dialogue and included the words "books" and "weekend" in the logs' names to quickly determine which version of the dialogue the participant saw. The code used to check that all the logs were accounted for is shown in Figure 7.

```python
import pandas as pd

# Load the CSV file into a DataFrame
df = pd.read_csv("/Users/svevabattisti/Desktop/MA in Linguistics/classes/I SEMESTER/Eye Tracking/FINAL REPORT /VS CODE/data/subject-2.csv")

# Strip whitespace from column names
df.columns = df.columns.str.strip()

# List of logs to check
logs_to_check = [
    'SENTENCE_LOWERCASE_ONSET',
    'SENTENCE_LOWERCASE_OFFSET_BOOKS',
    'SENTENCE_UPPERCASE_ONSET',
    'SENTENCE_UPPERCASE_OFFSET_WEEKEND',
    'SENTENCE_LOWERCASE_OFFSET_WEEKEND',
    'SENTENCE_UPPERCASE_OFFSET_BOOKS'
]

# Ensure the 'USER' column exists
if 'USER' in df.columns:
    # Get unique values in the 'USER' column
    user_column_values = df['USER'].unique()

    # Check which logs are present in the 'USER' column
    missing_logs = [log for log in logs_to_check if log not in user_column_values]

    if not missing_logs:
        print("All specified logs are present in the 'USER' column.")
    else:
        print("The following logs are missing from the 'USER' column:")
        for log in missing_logs:
            print(f"- {log}")
else:
    print("The 'USER' column does not exist in the DataFrame.")
```
✓ 1.4s

```
The following logs are missing from the 'USER' column:
- SENTENCE_LOWERCASE_OFFSET_WEEKEND
- SENTENCE_UPPERCASE_OFFSET_BOOKS
```

Figure 7: Check logs script

In the example above, if a participant saw the dialogue about books in lowercase and the dialogue about the weekend in uppercase, the missing logs will only show the versions of the dialogues that the participant did not see. This ensures we can quickly determine which dialogues were presented to each participant.

We additionally plotted this for each participant, as follows (Figure 8).
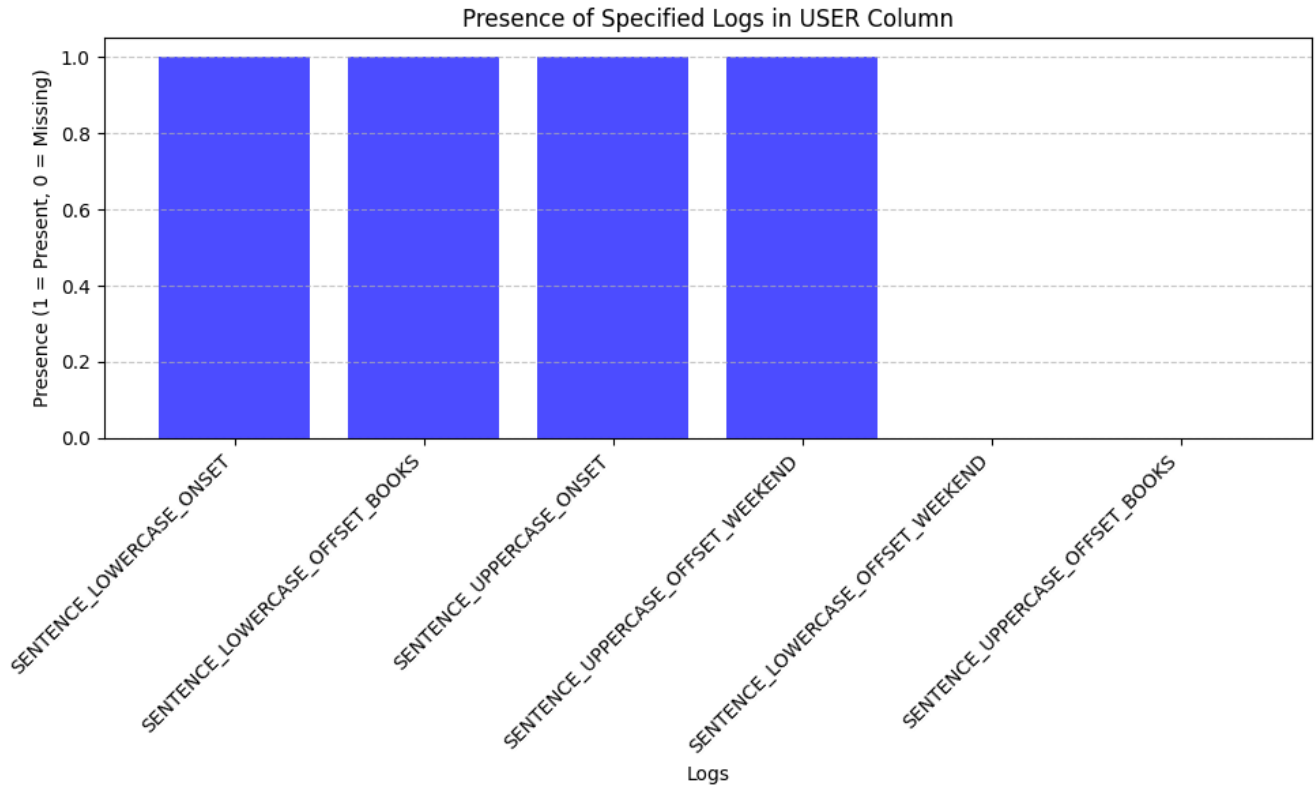
Figure 8: Subject 2 number of logs

To ensure that the logs were triggered correctly and functioned as intended, we plotted the FPOGX values against the sample index. We enhanced the plot with vertical lines and annotations to highlight the specified USER log/triggers, providing a clear visualization of the events during the experiment (Figure 9 and Figure 10).

Figure 9: Subject 2 Logs

Figure 10: Subject 11 Logs

Next, we proceeded to plot FPOGX (Fixation Point of Gaze X) and FPOGY (Fixation Point of Gaze Y). These metrics are critical as they represent the participant's gaze coordinates on the screen. Ensuring these values are correctly logged is essential for verifying that the eye tracker is

accurately recording gaze positions. This step involved mapping the gaze points to ensure they aligned with our expected outcomes. By examining the gaze direction data, particularly focusing on the last row in each fixation ID, we were able to not only verify the coordinates but also gain insights into the behavior and patterns in the gaze data. The scatter plot below (Figure 11) illustrates this relationship for a single participant:
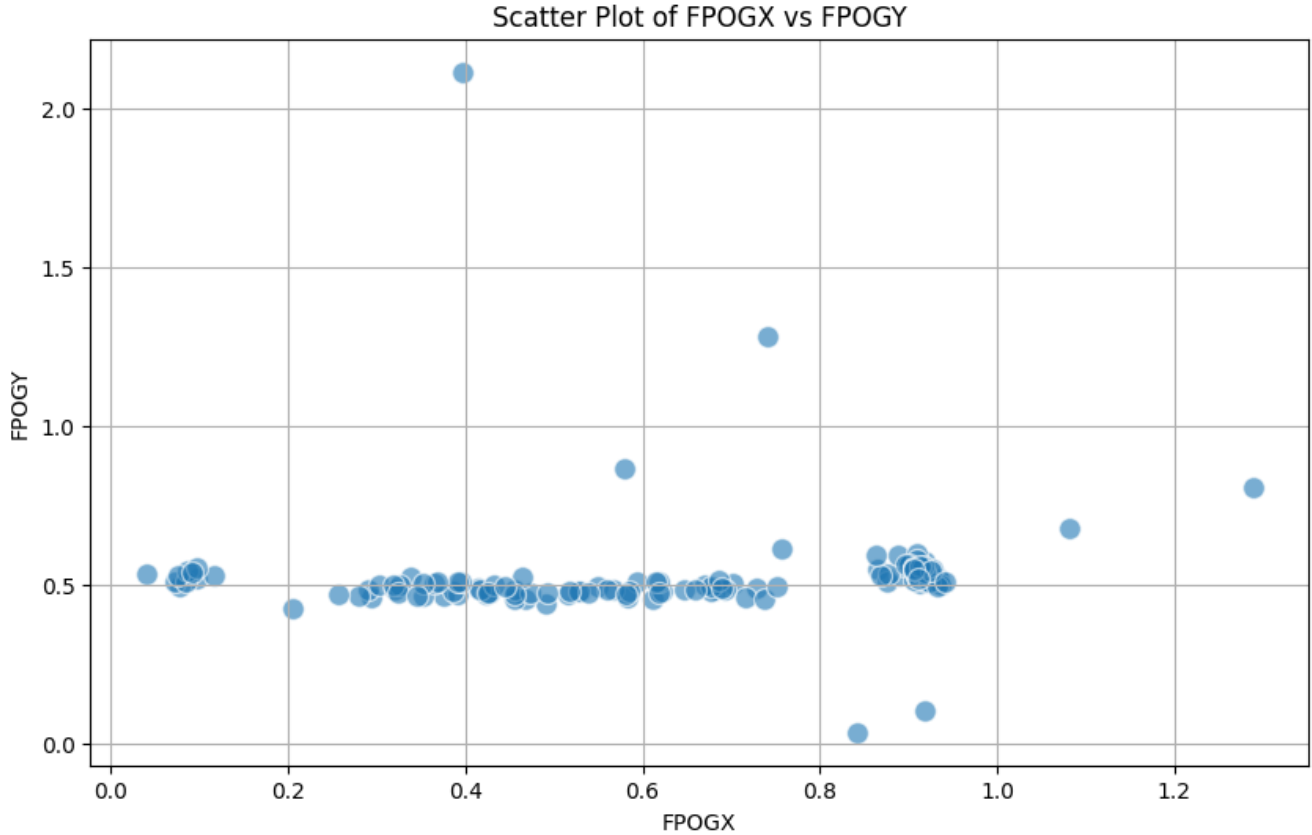

Figure 11: Subject 11 FPOGX vs FPOGY

To accurately assess the participant's gaze behavior, we plotted the FPOGX and FPOGY. Since each FPOGX and FPOGY value in the dataset corresponds to a sample within a fixation rather than the fixation itself, we filtered the dataframe to retain only the final FPOGID for each fixation. Additionally, we filtered out negative values, as they might indicate that the participant was not looking directly at the screen. With these precautions, we observed that the majority of data points are tightly clustered around the 0.5 mark on the FPOGY axis. This suggests that the participant's gaze was primarily focused at the middle vertical position of the screen, as expected. However, there are several outlier points with significantly higher FPOGY values. Potential reasons for these high FPOGY values include:

- Data Collection Artifacts: There may have been errors or noise during the data collection process, causing some gaze points to be recorded incorrectly.

- Participant Behavior: The participant may have looked away from the screen or at something outside the intended area of interest, leading to these high FPOGY values.

- Calibration Issues: The eye-tracking system might not have been properly calibrated for this participant, resulting in inaccurate gaze position data.

These sanity checks are a fundamental part of the experimental process, safeguarding the quality and validity of the research findings. Unfortunately, during this phase, 6 out of 12 participants had to be excluded from the experiment. For five participants, the lowercase onset log was not activated, and for one participant, no logs were recorded at all. Identifying the precise reasons for these issues requires a detailed review of the experimental setup, the protocols followed, and the raw data files.

We believe that the issue with the one participant who had no logs recorded might have been due to data corruption, meaning the log files might have been corrupted during the recording or saving process. For the other five participants, we suspect they performed the experiment with an older version of the software where the logging settings might have been configured incorrectly, causing certain events not to be logged.

5.3 Analysis Pipeline

We evaluated three criteria to determine the readability of uppercase and lowercase fonts: As an illustration of our analysis procedure, we present the plots from two participants. However, these analyses were conducted for all individuals who participated in our experiment and whose data met the criteria established by our sanity checks.

5.3.1 Fixation Duration:

Fixation duration is the length of time a participant's gaze remains fixed on a single location. For Subject 2 (Figure 12), the average duration of all fixations is calculated to provide an overall measure of how long each fixation lasts on average. Similarly, for Subject 6 (Figure 12), the average duration of all fixations is determined in the same manner.Both Subject 2 and Subject 6 exhibit the highest frequency of fixations within the 0 to 0.5-second range, indicating that most of their fixations fall within this short duration. Regarding long fixations, Subject 2's fixation durations ranged up to 6 seconds, whereas Subject 6's fixation durations range up to 5 seconds.

The interpretation of these findings is as follows:

- Short Fixations: The fact that the majority of fixations are under 1 second suggests that both participants found most parts of the text readable and easy to process.

- Peak Fixation Duration: The highest count of fixations occurring in the 0 to 0.5-second range indicates that participants engaged in efficient reading for most text segments.

- Longer Fixations: The fewer but longer fixations, up to 6 seconds for Subject 2 and up to 5 seconds for Subject 6, highlight areas where participants experienced more difficulty or required extra attention to understand the text.
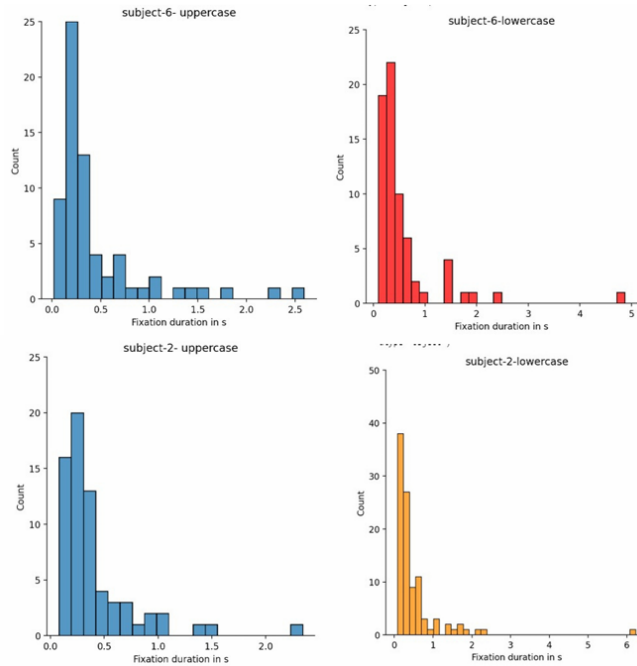

Figure 12: Fixation Duration for Subjects 2 and 6

5.3.2 Number of Fixations:

The total number of fixations refers to the total number of times a participant's gaze is fixed on various locations within the text. To illustrate this analysis, we'll refer to Figure 12. For Subject 2 (Fig. 12), the total number of fixations recorded during the reading tasks is the sum of all individual fixations observed. Similarly, for Subject 6 (Fig. 12), the total number of fixations is the sum of all fixations noted during their reading tasks. Both histograms indicate a high frequency of short fixations (0 - 0.5 seconds) and fewer longer fixations. Regarding longer fixations, which are defined as those lasting over 1 second, Subject 2 has fixations recorded within the range of over 1 second, extending up to 6 seconds. In contrast, Subject 6 has longer fixations within the range of over 1 second, but only up to 5 seconds. Differences in the number of fixations between uppercase and lowercase text are minimal, suggesting that text case does not significantly impact reading behavior.

5.3.3 Code Explanation

- Lowercase Code

19

This code processes eye-tracking data from a specified subject file to focus on segments where the subject read lowercase text. It loads the data, identifies the start and end points of the lowercase reading segments, and extracts these segments into a single DataFrame. Duplicate rows are removed to retain only the last occurrence of each fixation point. A histogram is then created to visualize the distribution of fixation durations during the lowercase reading segments. Finally, the histogram is customized and displayed to show how long the subject's eyes stayed on one point while reading lowercase text.

---

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

#Function to load cleaned subject data def load_cleaned_subject_data(file): return pd.read_csv(file)

#List of cleaned subject files cleaned_subject_files = ['subject-6.csv'] #'subject-3.csv','subject-5.csv','subject-6.csv','subject-7.csv','subject-8.csv']

#Process each subject's data for file in cleaned_subject_files: subject_id = file.split('.')[0] # Extract subject ID from the file name df_subject = load_cleaned_subject_data(file) # df_subject = df_subject.dropna() # Remove rows with NaN values

```
# Keep only the last occurrence of each FPOGID in the entire dataframe
#df_subject_duplicate = df_subject.drop_duplicates(subset='FPOGID', keep='last')
print(df_subject.columns)
df_upparcasestart=df_subject['USER']=='SENTENCE_LOWERCASE_ONSET'
df_upparcaseend=df_subject['USER']=='SENTENCE_LOWERCASE_OFFSET_BOOKS'
# Initializing an empty list to store the extracted segments
extracted_segments = []

# Iterating over the DataFrame to extract segments between start and end markers
start_index = None
for index, row in df_subject.iterrows():
    if df_upparcasestart[index]:
        start_index = index
    elif df_upparcaseend[index] and start_index is not None:
        end_index = index
        extracted_segments.append(df_subject.loc[start_index:end_index])
        start_index = None
```

```
# Concatenate all extracted segments into a single DataFrame
df_extracted = pd.concat(extracted_segments, ignore_index=True)
df_subject_duplicate = df_extracted.drop_duplicates(subset='FPOGID', keep='last')

sns.displot(df_subject_duplicate, x="FPOGD",color='red')
plt.title(f'{subject_id}-lowercase')
plt.ylabel('Count')
plt.xlabel('Fixation duration in s')
plt.ylim(0, 25)
plt.show()
```

- Uppercase code

This code processes eye-tracking data from 'subject files to focus on uppercase text reading segments. It loads the data and identifies the start and end markers for uppercase text ('SENTENCE_UPPERCASE_ONSET' and 'UPPERCASE_DIALOGUE_END'). The code extracts the segments between these markers, removes duplicate fixations, and combines them into a single DataFrame. A histogram is then created using Seaborn to visualize the distribution of fixation durations during the uppercase reading segments. Finally, the histogram is customized with a title, labels, and displayed to show the duration of fixations while reading uppercase text. The code used is as follows:

CODE for Upper CASE

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

#Function to load cleaned subject data def load_cleaned_subject_data(file): return pd.read_csv(file)

#List of cleaned subject files cleaned_subject_files = ['subject-6.csv'] #'subject-3.csv','subject-5.csv','subject-6.csv','subject-7.csv','subject-8.csv']

Process each subject's data for file in cleaned_subject_files: subject_id = file.split(':')[0] # Extract subject ID from the file name df_subject = load_cleaned_subject_data(file) # df_subject = df_subject.dropna() # Remove rows with NaN values

```python
# Keep only the last occurrence of each FPOGID in the entire dataframe
#df_subject_duplicate = df_subject.drop_duplicates(subset='FPOGID', keep='last')
print(df_subject.columns)
df_upparcasestart=df_subject['USER']=='SENTENCE_UPPERCASE_ONSET'
df_upparcaseend=df_subject['USER']=='UPPERCASE_DIALOGUE_END'
# Initializing an empty list to store the extracted segments
extracted_segments = []

# Iterating over the DataFrame to extract segments between start and end markers
start_index = None
for index, row in df_subject.iterrows():
    if df_upparcasestart[index]:
        start_index = index
    elif df_upparcaseend[index] and start_index is not None:
        end_index = index
        extracted_segments.append(df_subject.loc[start_index:end_index])
        start_index = None

# Concatenate all extracted segments into a single DataFrame
df_extracted = pd.concat(extracted_segments, ignore_index=True)
df_subject_duplicate = df_extracted.drop_duplicates(subset='FPOGID', keep='last')

sns.displot(df_subject_duplicate, x="FPOGD")
plt.title(f'{subject_id}- uppercase')
plt.ylabel('Count')
plt.xlabel('Fixation duration in s')
plt.ylim(0, 25)
plt.show()
```

5.3.4 Landing Spot:

Landing spot is the initial point where the participant's gaze lands on a word.

The landing spot on one of the words in each sentence was detected. In eye tracking experiments, "landing spot" refers to the precise location on a visual stimulus where a person's gaze comes to rest. It represents the point at which the individual's eye fixates for a certain duration. Eye tracking technology captures these landing spots to analyze how people view and interact with various types of visual information, including webpages, advertisements, product designs, or, in our case, text presented in different case types. When analyzing our data, the question we tried to answer was: Does the case type of a sentence (uppercase vs. lowercase) affect where the eye first lands? By identifying the very first landing spot, we aimed to gain insights into how different case

types influence attention, the processing of visual elements, and navigation through text content. In order to accurately represent this type of analysis, two main measurments were required:

1. Bounding boxes: are used to define and analyze specific regions of interest within a text, such as the bounderies of each word in a sentence.
2. First fixation's coordinates (FPOGX and FPOGY) after the sentence onset.

The following paragraph explains the various steps that lead to the coordinates of the bounding boxes.

```python
def get_word_dimensions(word, font_size):
    # Assuming a monospace font where each character width is the same
    char_width = font_size  # Adjust based on actual font characteristics

    # Calculate width and height
    width = len(word) * char_width
    height = font_size  # Assuming font height is same for all characters

    return width, height

def measure_sentence(sentence, font_size):
    words = sentence.split()
    word_dimensions = {}

    for word in words:
        width, height = get_word_dimensions(word, font_size)
        word_dimensions[word] = (width, height)

    return word_dimensions

# Example usage:
font_size = 32  # Example font size (adjust as needed)

sentence = "hi, lisa! did you finish the book i lent you?"
word_dimensions = measure_sentence(sentence, font_size)

# Print word dimensions
for word, (width, height) in word_dimensions.items():
    print(f"Word '{word}': Width={width}px, Height={height}px")
```

Figure 13: Code for width and height

First, it was essential to determine the width and height of each word in the sentence. The height of the words was established based on the font size used in OpenSesame's sketchpad (32px). For the width, we used the following code (Figure 13), which was based on the fact that our stimuli were presented in a monospaced font. Once we added the sentence we wanted to analyze and the font's heigh it would give us the width as a result.

The next step involved obtaining the coordinates of the actual bounding boxes. We achieved this using the `calculate_sentence_bounding_boxes` function, as shown in the code below (Figure

14). This function required the sentence, the font height, and the width of each word, which we had previously calculated using the methodology previously described.

```python
def calculate_sentence_bounding_boxes(sentence, x_start, y_start, word_widths, word_height):
    # Initialize an empty dictionary to store bounding boxes
    bounding_boxes = {}

    # Split the sentence into individual words
    words = sentence.split()

    # Initialize x-coordinate with starting position
    x = x_start

    # Iterate through each word in the sentence
    for word, width in zip(words, word_widths):
        # Calculate bounding box for each word
        x1 = x
        y1 = y_start
        x2 = x + width
        y2 = y_start + word_height

        # Store bounding box coordinates for the word
        bounding_boxes[word] = (x1, y1, x2, y2)

        # Update x-coordinate for the next word (considering spacing between words)
        x += width  # Add width of the current word

    return bounding_boxes

# Example usage:
sentence = "HI, SARAH! HOW WAS YOUR WEEKEND? WHAT DID YOU DO?"
x_start = -288
y_start = 0
word_widths = [96, 192, 96, 128, 128, 224, 128, 96, 96, 96]  # Hypothetical widths for each word
word_height = 32  # Hypothetical height for all words

# Calculate bounding boxes for the sentence
bounding_boxes = calculate_sentence_bounding_boxes(sentence, x_start, y_start, word_widths, word_height)

# Print bounding boxes for each word in the sentence
for word, (x1, y1, x2, y2) in bounding_boxes.items():
    print(f"Bounding box for '{word}': ({x1}, {y1}, {x2}, {y2})")
```

Figure 14: Code for bounding boxes

The final step involved extrapolate from the data frame the coordinates of the very first fixation after the sentence onset (Figure 15).

24

```python
import pandas as pd

# Load the CSV file into a DataFrame
df = pd.read_csv("/Users/svevabattisti/Desktop/MA in Linguistics/classes/I SEMESTER/Eye Tracking/FINAL REPORT /VS CODE/data/subject-11.csv")

# Strip whitespace from column names
df.columns = df.columns.str.strip()

# Ensure the necessary columns exist
required_columns = ['USER', 'FPOGX', 'FPOGY']
if not all(col in df.columns for col in required_columns):
    raise ValueError("Required columns are missing from the DataFrame.")

# Identify the index of the first uppercase sentence onset
uppercase_onset_indices = df[df['USER'].str.contains('SENTENCE_UPPERCASE_ONSET', na=False)].index

if not uppercase_onset_indices.empty:
    first_uppercase_onset_index = uppercase_onset_indices[0]

    # Get the DataFrame rows after the first uppercase onset
    df_after_onset = df.iloc[first_uppercase_onset_index + 1:]

    # Find the first occurrence of FPOGX and FPOGY after the onset
    if not df_after_onset.empty:
        first_fpgox = df_after_onset['FPOGX'].iloc[0]
        first_fpgoy = df_after_onset['FPOGY'].iloc[0]

        print(f"First FPOGX after the onset: {first_fpgox}")
        print(f"First FPOGY after the onset: {first_fpgoy}")
    else:
        print("No data available after the first uppercase sentence onset.")
else:
    print("No uppercase sentence onset found in the 'USER' column.")
```

Figure 15: First FPOGX and FPOGY after sentence onset

This code was used to get FPOGX and FPOGY values for both uppercase and lowercase sentences. Finally, all the information gathered would be put together and used to plot the landing spot for uppercase and lowercase sentences. In the figure below we can see the final plot for one of our participants (Figure 18).
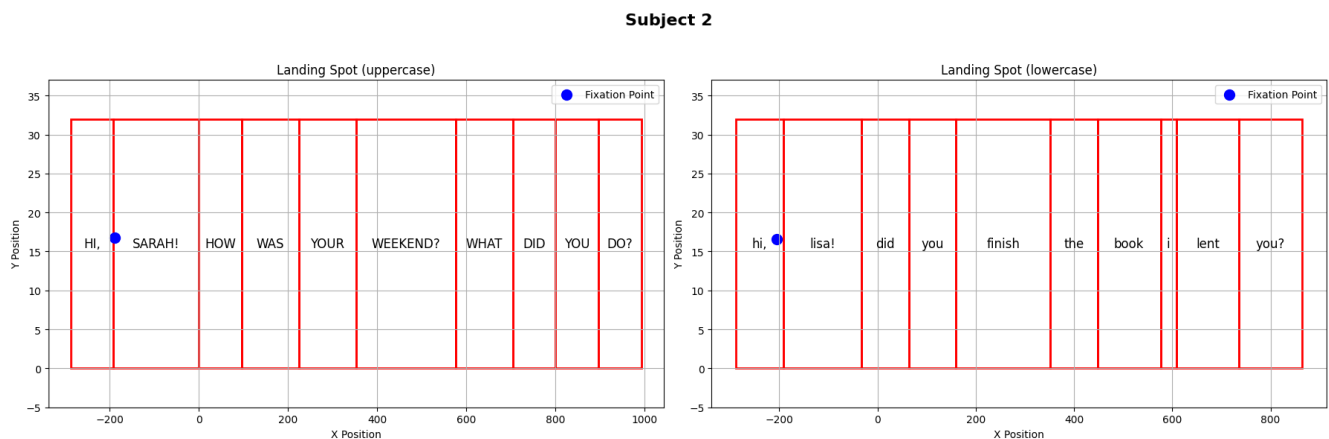

Figure 16: Subject 2 Landing Spot on uppercase and lowercase sentences

5.3.5 Observations

For what concerns the number and duration of fixations the data showed that most fixations are under 1 second. Providing that there is no particular difficulty when reading either case types. Additional analysis on landing spot also confirmed this hypothesis showing that for both case types the landing spot, which was found on one word in each sentence, was on almost the same spot. The data gathered for number and duration of fixations and landing spot showed no big difference between the two case formats. Additionally, for reasons explained previously not all participants' data could be analysed. Figures 16 and 17 reveal overall results for two of our participants in our experiment. The logs, however, are nor visible for all of them due to the defected data from the eye tracker.
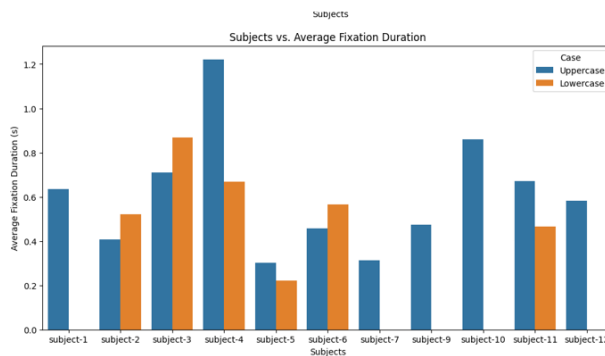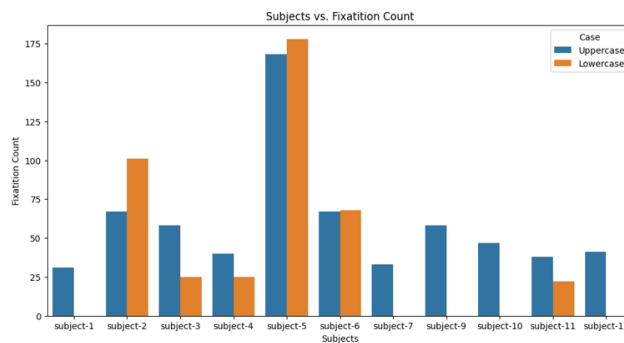

Figure 17: Fixation Duration

Figure 18: Number of Fixations

6 Results

The analysis of eye-tracking data revealed that uppercase text resulted in a marginally higher number of fixations compared to lowercase text, aligning with Tinker and Paterson's findings.

However, the duration of fixations and landing spots did not show significant differences, consistent with White and Liversedge's results. Our analysis, based on the visualisations and the data, confirmed that these variations were not significant, supporting our hypothesis that font case does not markedly affect reading behavior. The findings support previous research in the field of cognitive neuroscience and multilingualism (Andrews, 2014) [1]. Additionally, it's important to consider the optimal viewing position effect in word recognition as outlined by O'Regan and Jacobs (1992) [3], which was not thoroughly addressed in our study due to the difficulty in identifying the landing spots on each word when the stimuli were presented as entire sentences.

7 Discussion

7.1 Limitations

1. Since the stimuli units were sentences, and each sentence must be preceded and followed by a fixation dot, the odds of forgetting the content of the previous sentence could increase.

2. Answering the two follow-up comprehension questions could have been disracting and one was prone to changing the position of head and hence deficient data, even when chin rest was utilized.

3. Competency of participants in English differed and this could be the reason for variability in reading behaviours, rather than the experimental items.

4. The number of words in each sentence were equal, despite this, the general difficulty and length of the words and the two dialogues were not easy to control.

5. The bounding boxes that we calculated to find the landing spots with would not fit perfectly around the words and also the finding the landing spot on single words was not easy to resolve.

6. Due to the absence of logs for some subjects, we were unable to use their data. As a result, only 6 out of the 12 subjects provided us with usable data.

7.2 Further Improvements

This projects although replicates the results of the reference paper to a great extent, there are still some improvements that can make the results and the analysis more robust and reliable. Some of the improvements are as follows:

1. More extensive study with a larger number of participants will ensure that the results are more generalizable. The study can be conducted with participants from different age groups and educational backgrounds. Also the study can be conducted in different languages to prove the universality of the observed effects.

2. The Gazepoint GP3 eye tracker can be replaced with a more advanced eye tracker like Eyelink 1000 which can record at a higher sampling rate and thus provide more accurate results. The Eyelink 1000 eye tracker allows less head movement due to its design and thus the recording of the eye movements are more accurate.

3. The experiment requires us to study the fixation behavior of the participants. Therefore, better optimized fixation detection algorithms can be used and even integrated to the open source software like OpenSesame to improve the analysis.

8 Conclusion

In this project we attempted to investigate reading behaviours of participants in regards with two font case formats. Our final research hypothesis, in alignment with the results of White and Liversedge (2006), posits that there is no significant difference in reading behavior between uppercase and lowercase text. This finding supports previous research indicating limited differences in eye movement behavior when participants read text in different cases (White and Liversedge [6]). Similarly, Tinker and Paterson's (1939) earlier work highlighted the nuances of text readability in different cases, further emphasizing the complexity of reading processes [5].

By confirming these findings, our study contributes to the ongoing discourse on typography and reading efficiency, providing valuable insights for future research. Further research is needed to explore these findings in different languages and contexts. Previous foundational research, such as that by Tinker and Paterson (1939) [5], and recent insights by Slattery (2016) [4], that underlines the importance of continued exploration in this area, addressing the practical implications for font design, by emphasizing the importance of aligning font characteristics with human visual and cognitive processing, motivated the present experiment exponentially.

9 Contribution Table

Table 1: Contributions of each team member to the project

Task	Sveva	Raheleh	Aishwarya
Background Literature	o	x	o
Experiment Design	x	o	x
Stimulus Design	x	x	x
Piloting	x	x	x
Data-Recording	x	x	x
Non-Final-Talk presenting (who talks)	x	o	o
Non-Final-Talk presenting (who prepares)	x	o	o
Final-Talk Presenting (who talks)	x	x	x
Final-Talk Presenting (who prepares)	x	o	x
Data Analysis Scripts	x	o	x
Report Writing	o	x	o

- x: main contributor
- o: supporting contributor

References

[1] Edna Andrews. *Cognitive Neuroscience and Multilingualism*. Routledge, 2014. ISBN: 9780415713288.

[2] George W. McConkie et al. "Eye movement control during reading: I. The location of initial eye fixations on words". In: *Vision Research* 28.10 (1988), pp. 1107–1118. DOI: 10.1016/0042-6989(88)90137-x.

[3] J. K. O'Regan and A. M. Jacobs. "Optimal Viewing Position Effect in Word Recognition: A Challenge to Current Theory". In: *Journal of Experimental Psychology: Human Perception and Performance* 18.1 (1992), pp. 185–197. DOI: 10.1037/0096-1523.18.1.185.

[4] Timothy Slattery. "Eye movements: from psycholinguistics to font design". In: Feb. 2016, pp. 54–78. ISBN: 978-981-4759-53-3. DOI: 10.1142/9789814759540_0004.

[5] Miles A. Tinker and Donald G. Paterson. *Reading: A Physiological and Psychological Analysis*. University of Iowa Press, 1939.

[6] Sarah J. White and Simon P. Liversedge. "Linguistic and nonlinguistic influences on the eyes' landing positions during reading". In: *Quarterly Journal of Experimental Psychology (Hove)* 59.4 (Apr. 2006), pp. 760–782. DOI: 10.1080/02724980543000024.