Answer sheet Processing lab 1

Instructions: Fill out your answers below. Make a PDF of the complete file, and upload that **PDF** on Blackboard.

Student 1 (Full name & student number): Sterre van Strien (6138942)

Student 2 (Full name & student number): Vito Vekic (1091719)

Blackboard question 1

1A: For example 3: what is the predicted task time for the model that does all steps as fastman in milliseconds?

The predicted task time for the model which does all steps as fastman is 180 ms. This is calculated by adding the time for the perceptual step twice (each being 50 ms, so together they're 100 ms), the time for the cognitive step is added twice (which are 25 ms respectively, resulting in a total time of 150 ms together with the perceptual step) and the motor step is added once (which is 30 ms), resulting in a total of 180 ms.

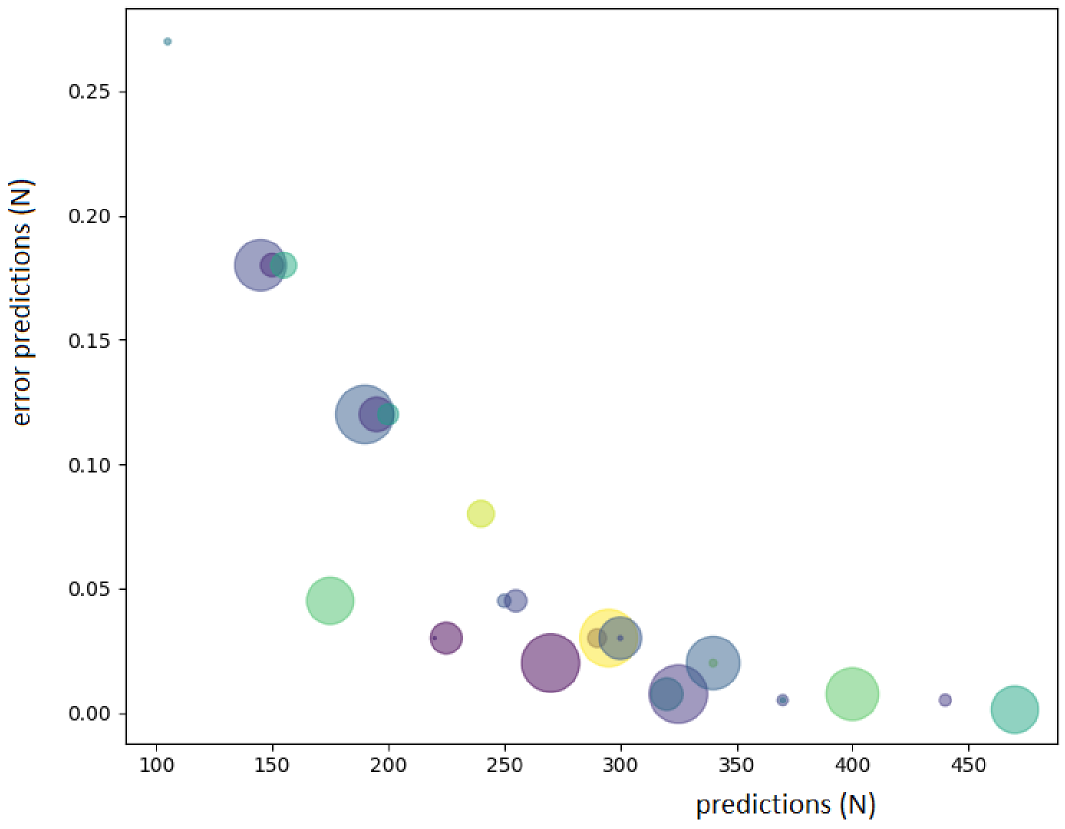
1B: For example 3: What is the predicted task time for a model that has the fastest perception, takes average time for cognitive steps, but is the slowest in motor execution?

The predicted task time for the model which does the steps as described above is 340 ms. In this case, the time for the fast perceptual step is 50 ms, the average cognitive step is 70 ms and the slow motor step is 100. We know that the total time is twice the perceptual time (so 2x 50 ms equals 100 ms) plus twice the cognitive step (so 2x 70 ms, which is 140, resulting in a total of 240 ms) and once the motor step (which is 100 ms), resulting in a total predicted task time of 340 ms.

1C: For example 4: What is the predicted slowest time that we might observe in this experiment?

The predicted slowest time we might observe is 1040 ms. The predicted slowest time is the case in which all steps are the slowest, so the perceptual step takes 200 ms, the cognitive step takes 170 ms and the motor step takes 100 ms. We then calculate the total predicted slowest time by adding the slowest timing for the stimulus, 240 ms, to twice the perceptual step (240 + 400 = 600 ms), twice the cognitive step (600 + 340 = 940 ms) and once the motor step (940 + 100 = 1040 ms). So the total is 1040 ms.

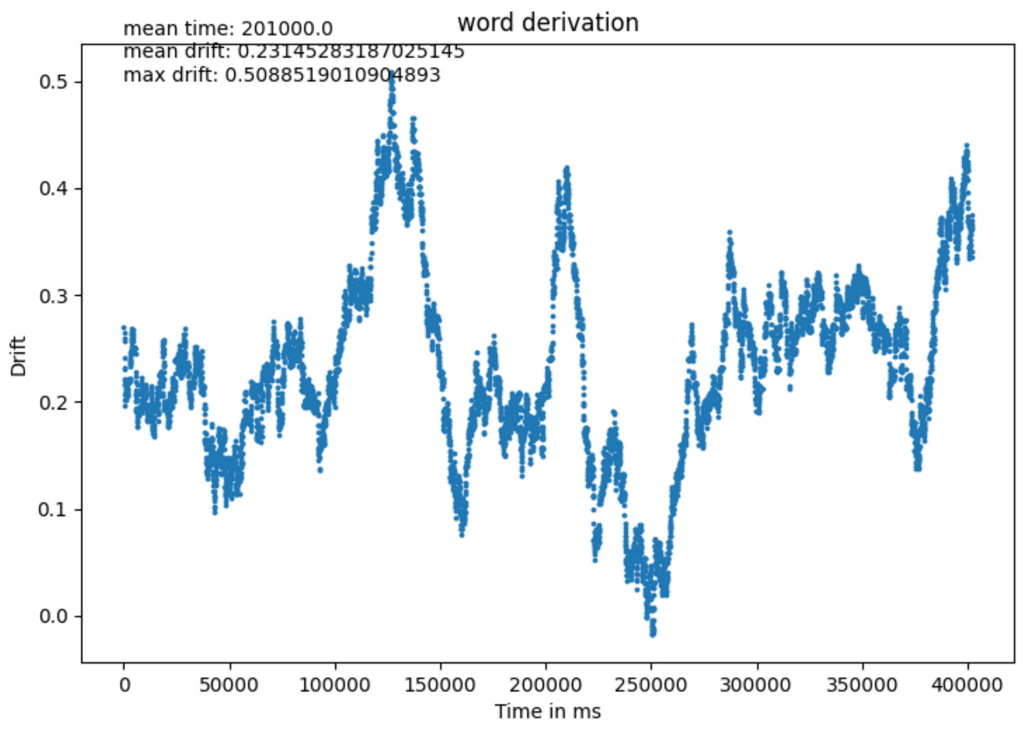
1D (paste a picture or screenshot below): For example 5: add a picture or screenshot of the scatterplot (as an image), Make sure to clearly label the x- and y-axis (give name and measurement unit, e.g. "time (ms)") and to use an appropriate range of values on each axis.



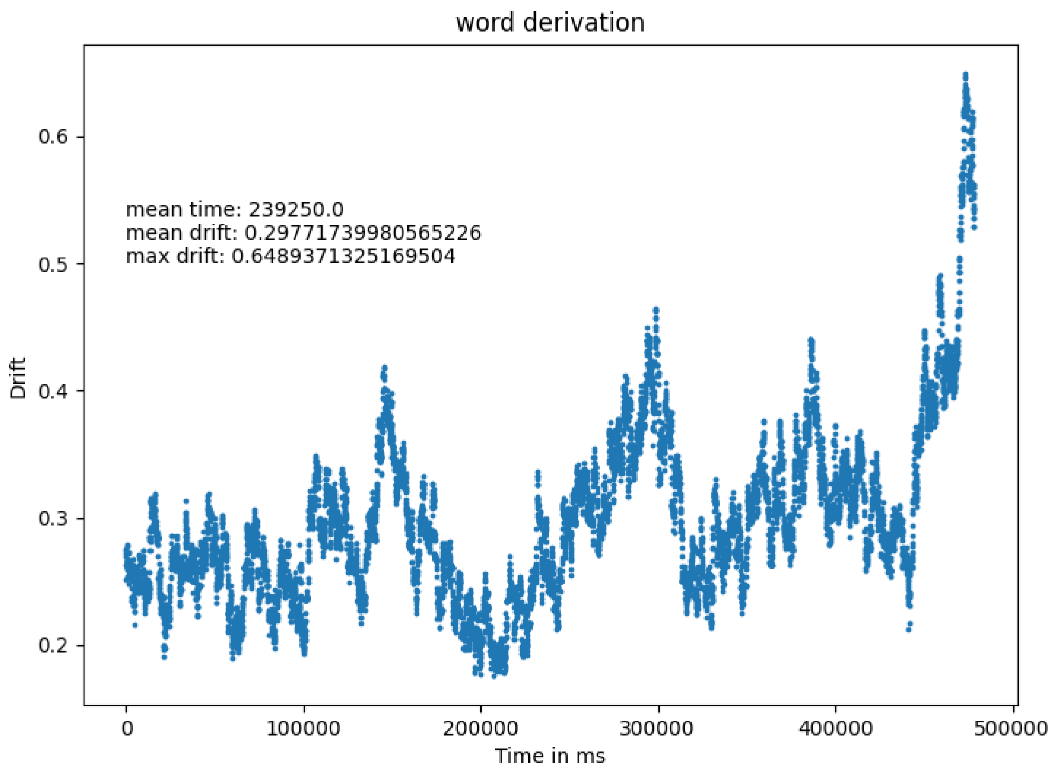
Blackboard question 2:

<u>2A (Copy screenshots or pictures of 10 different plots below): Create 10 plots and paste a picture or screenshot of them in the answer sheet. Each of plot should be a different outcome of running the "runTrial" function with parameters nrWordsPerSentence=17, nrSentences = 10, nrSteeringMovementsWhenSteering = 4, and interleaving="word". Please mind the criteria that were mentioned at the end of the engineering assignment for that model.</u>

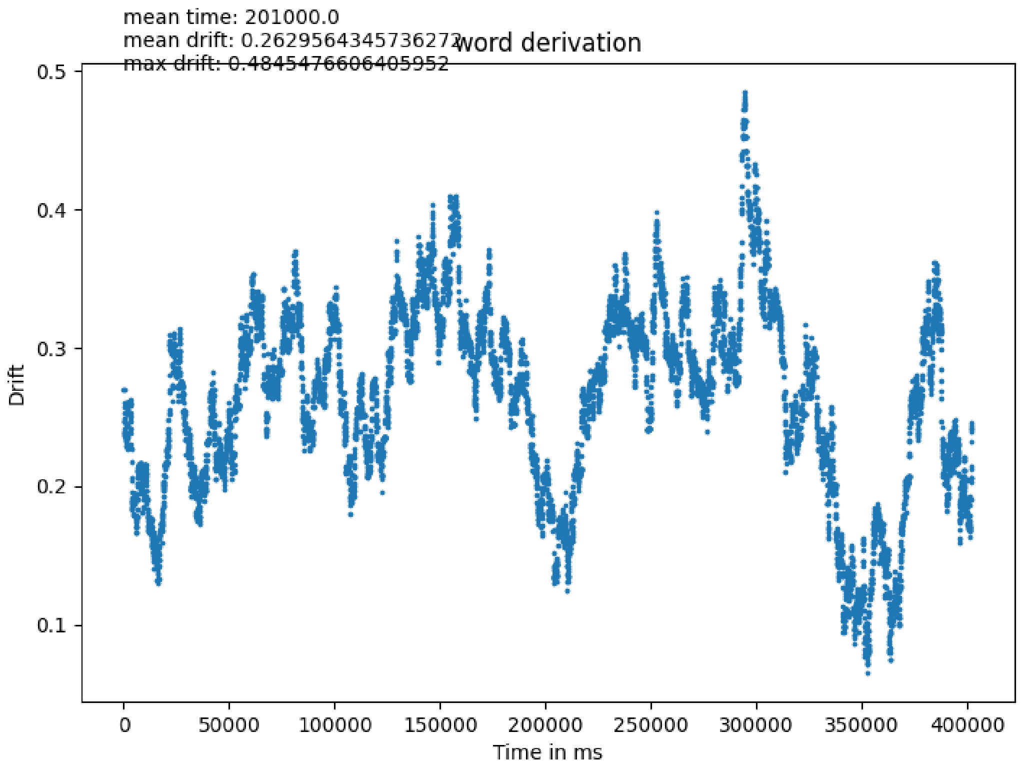
(See next pages)

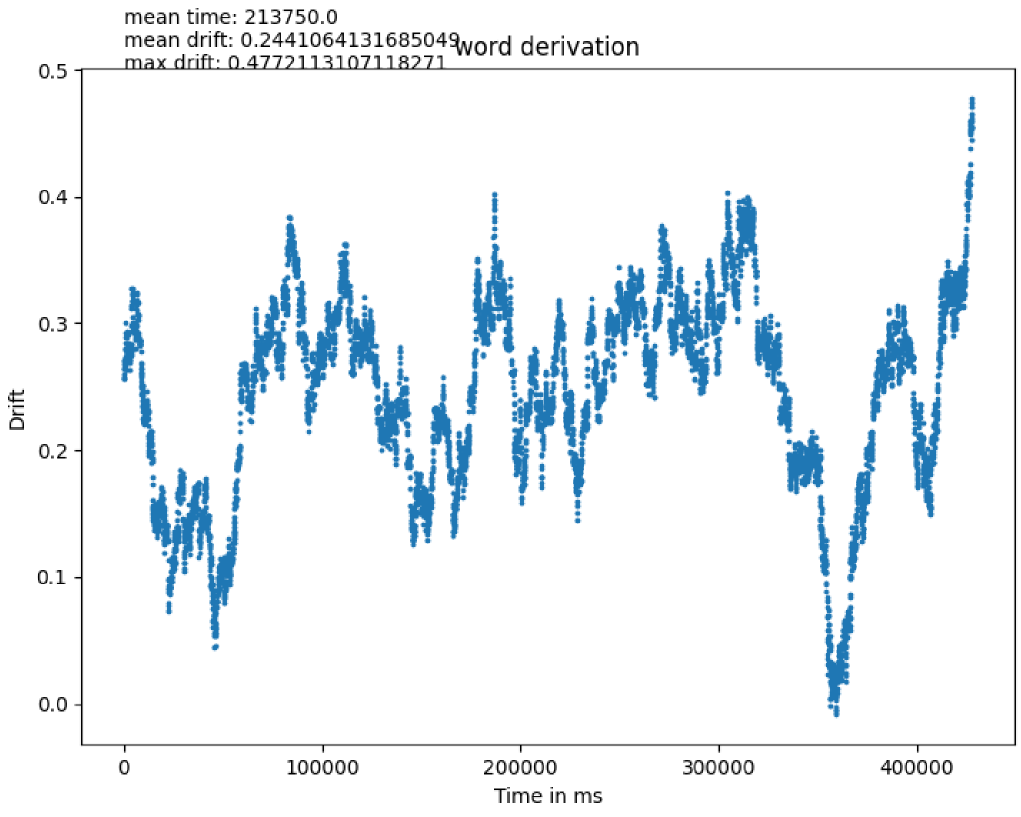


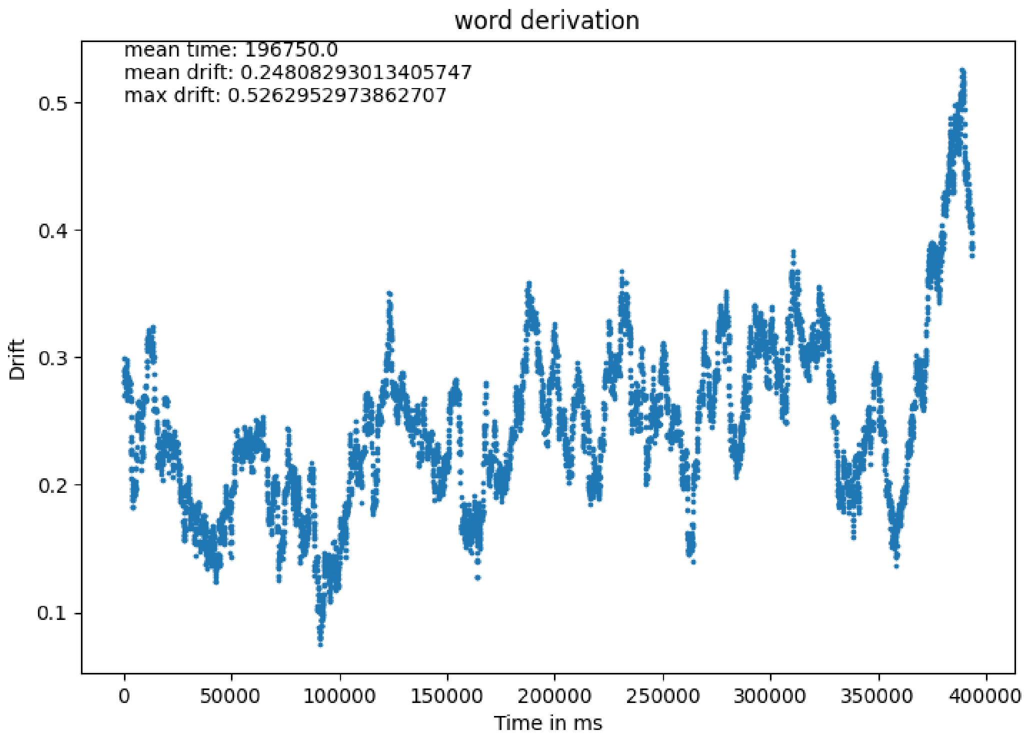
word derivation
mean time: 201000.0
mean drift: 0.23145283187025145
max drift: 0.5088519010904893
Drift
0.0
0.1
0.2
0.3
0.4
0.5
0
50000
100000
150000
200000
250000
300000
350000
400000
Time in ms

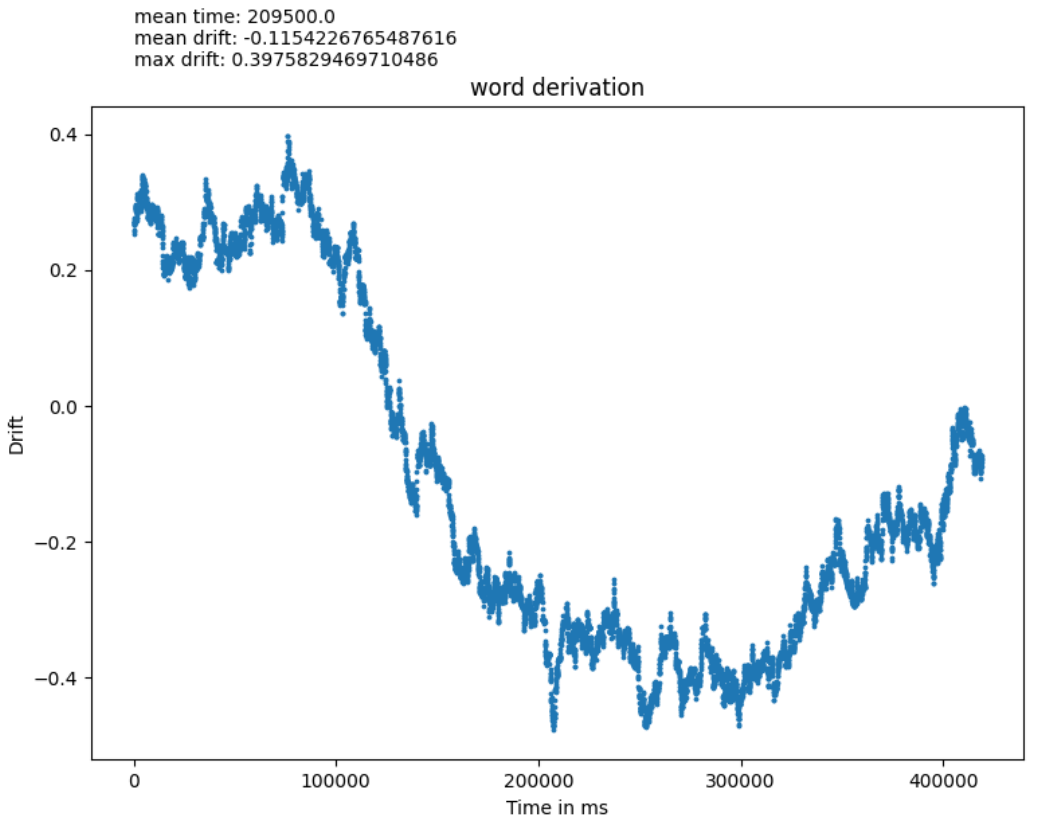


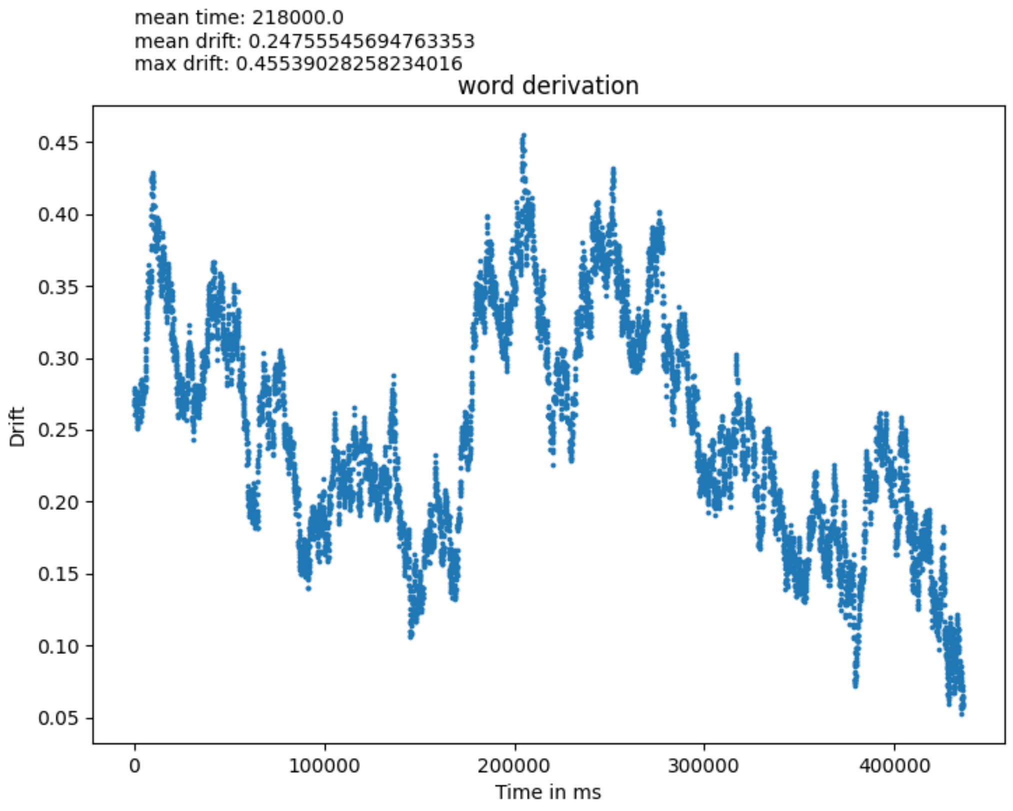
word derivation
mean time: 239250.0
mean drift: 0.29771739980565226
max drift: 0.6489371325169504
Drift
0.2
0.3
0.4
0.5
0.6
0
100000
200000
300000
400000
500000
Time in ms

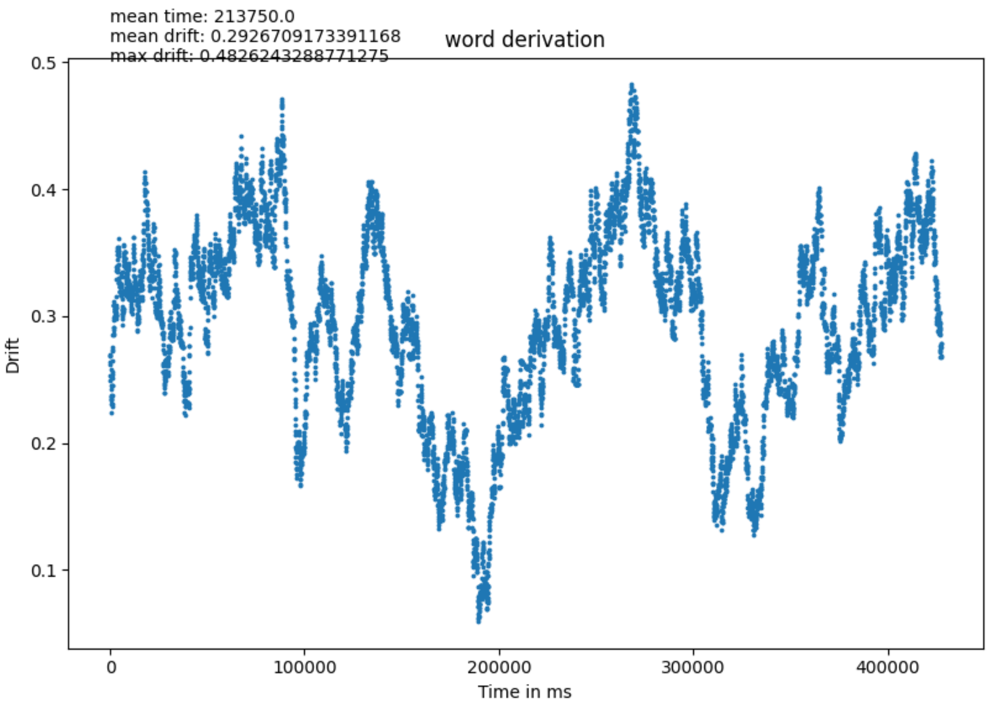

mean time: 201000.0
mean drift: 0.2629564345736272
max drift: 0.4845476606405952
word derivation
Drift
Time in ms
0.1
0.2
0.3
0.4
0.5
0
50000
100000
150000
200000
250000
300000
350000
400000


mean time: 213750.0
mean drift: 0.2441064131685049
max drift: 0.477211310711827
word derivation
Drift
Time in ms
0.0
0.1
0.2
0.3
0.4
0.5
0
100000
200000
300000
400000

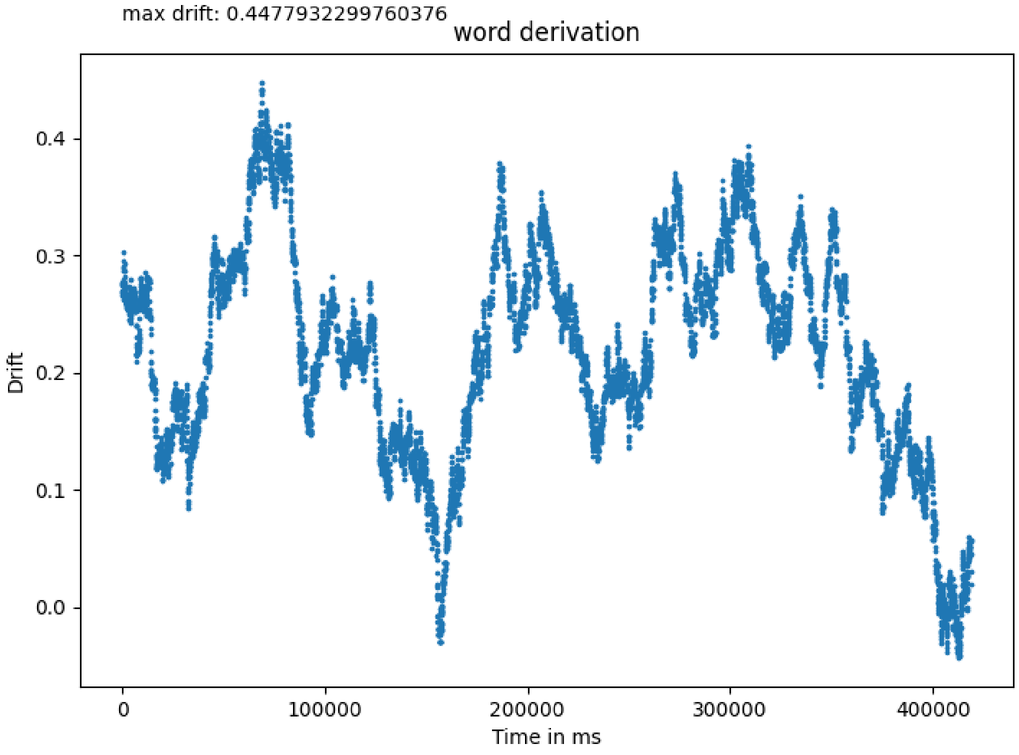

word derivation
mean time: 196750.0
mean drift: 0.24808293013405747
max drift: 0.5262952973862707
Drift
0.1
0.2
0.3
0.4
0.5
0
50000
100000
150000
200000
250000
300000
350000
400000
Time in ms

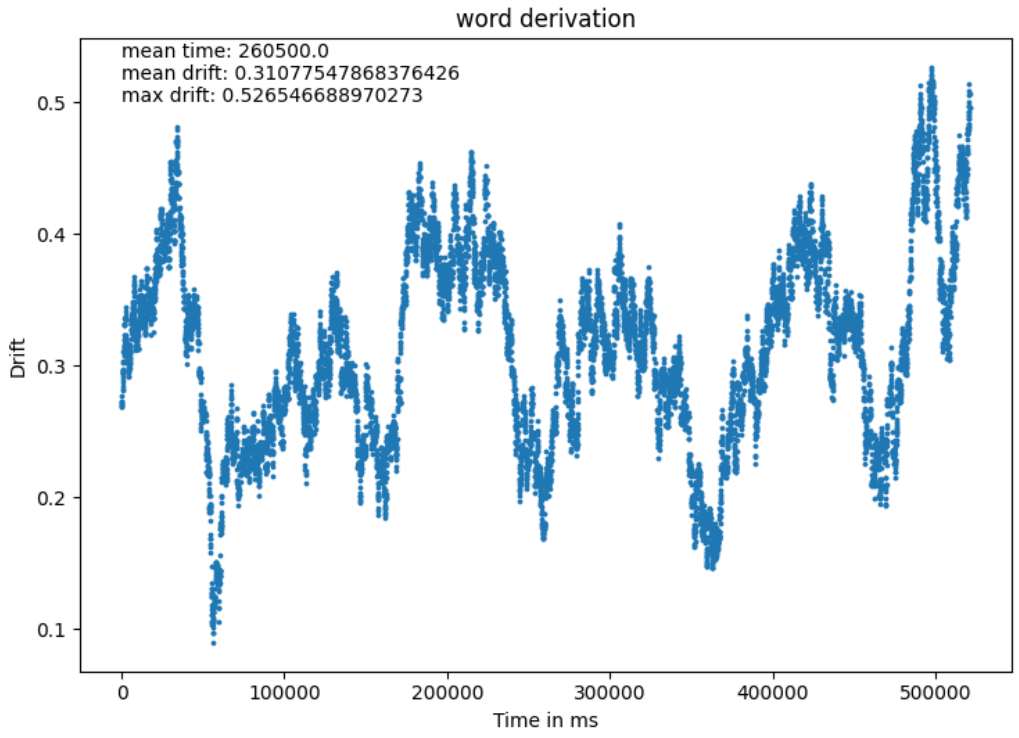

mean time: 209500.0
mean drift: -0.1154226765487616
max drift: 0.3975829469710486
word derivation
Drift
0.4
0.2
0.0
−0.2
−0.4
0
100000
200000
300000
400000
Time in ms


mean time: 218000.0
mean drift: 0.24755545694763353
max drift: 0.45539028258234016
word derivation
Drift
Time in ms
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0
100000
200000
300000
400000


mean time: 213750.0
mean drift: 0.2926709173391168
max drift: 0.4826243288771275
word derivation
Drift
Time in ms
0.1
0.2
0.3
0.4
0.5
0
100000
200000
300000
400000

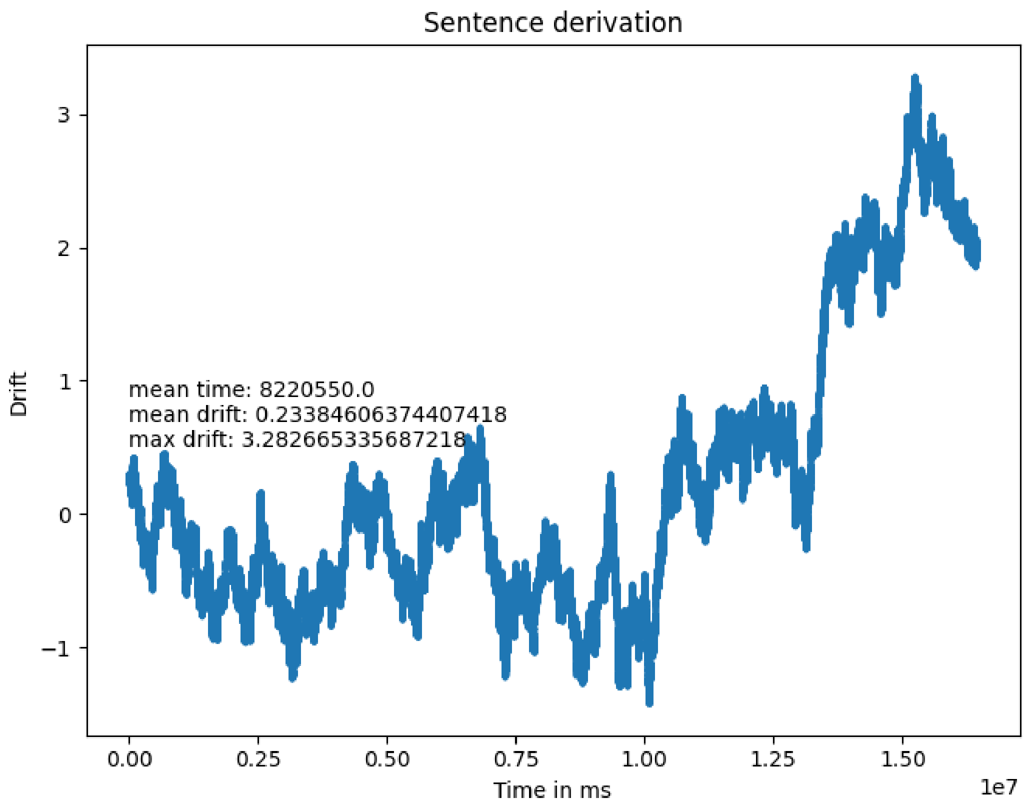
mean time: 209500.0
mean drift: 0.21806895806111654
max drift: 0.4477932299760376

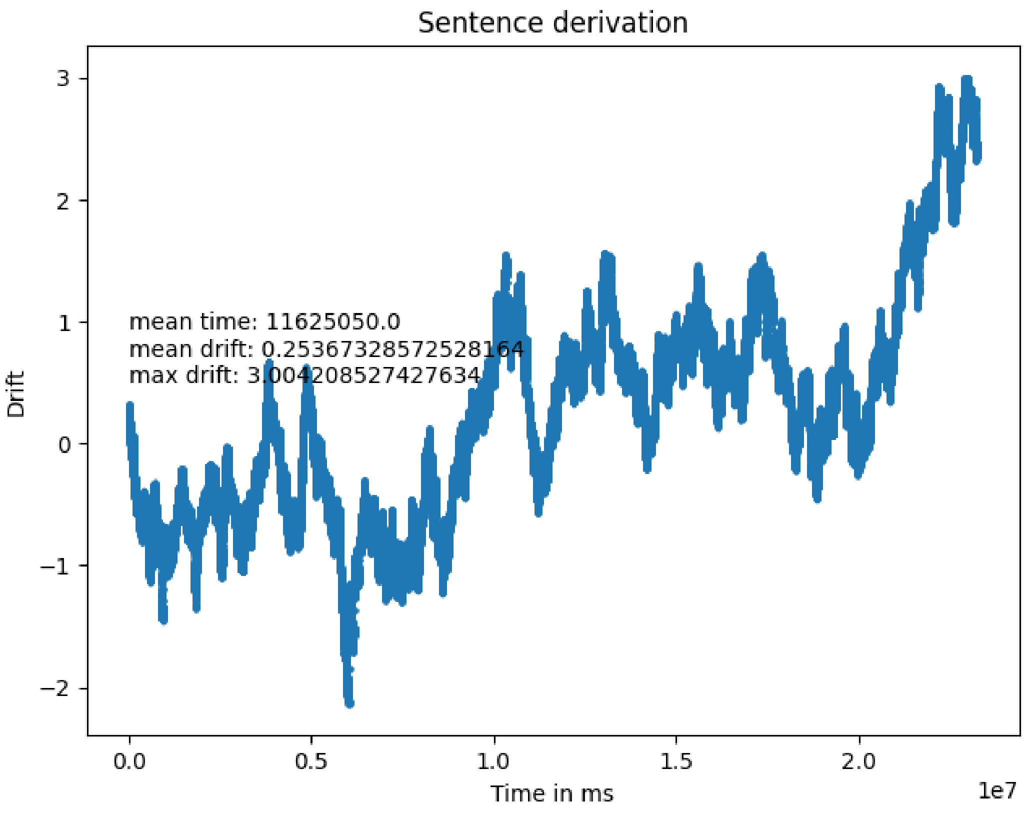


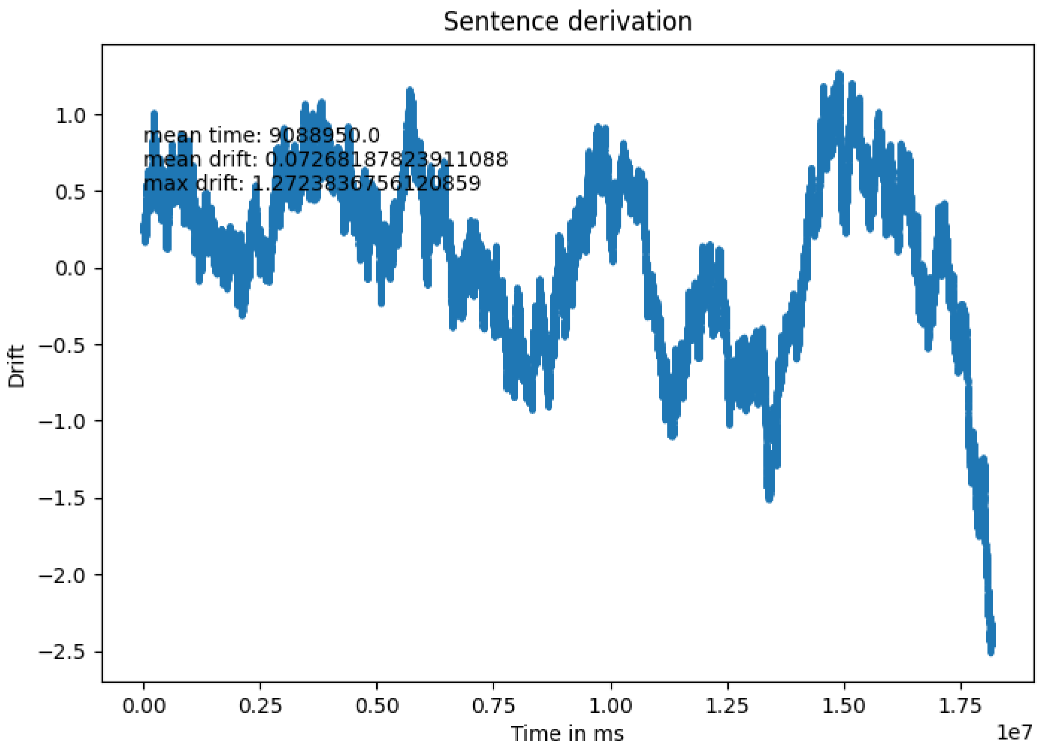

mean time: 260500.0
mean drift: 0.31077547868376426
max drift: 0.526546688970273

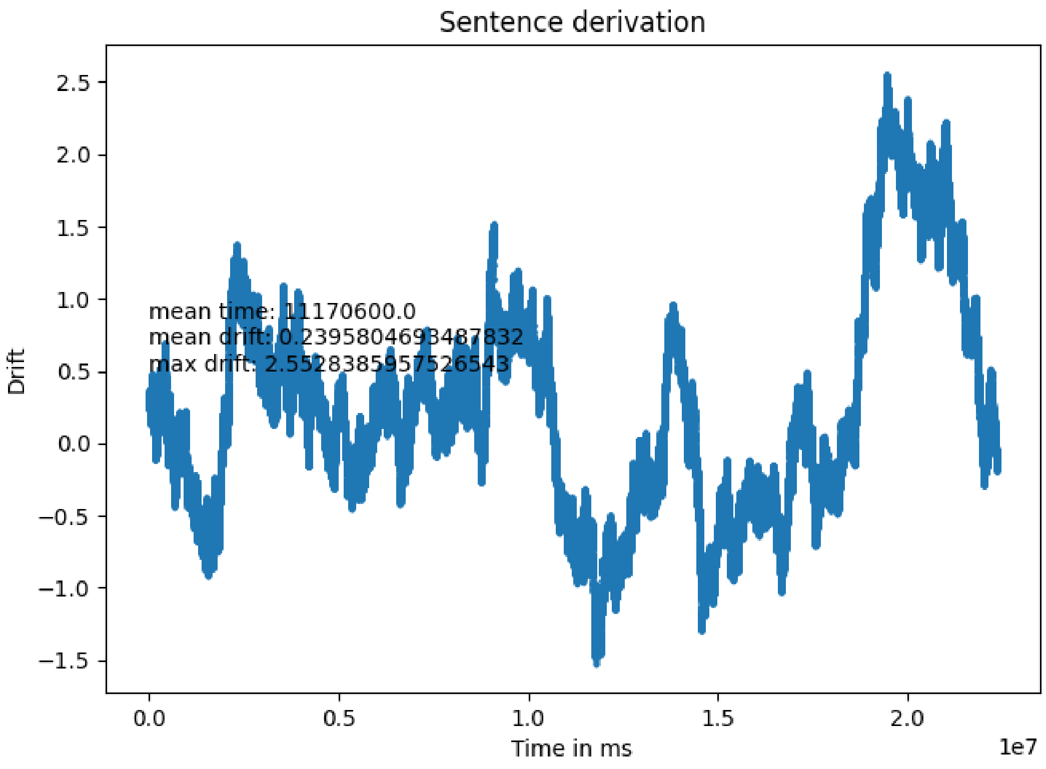
2B (Copy screenshots or pictures of 10 different plots below): Create another 10 plots and paste a picture or screenshot of them in the answer sheet. These plots should be 10 different outcomes of running the "runTrial" function with the same parameter settings as for question 2A, except that the interleaving strategy should be "sentence".

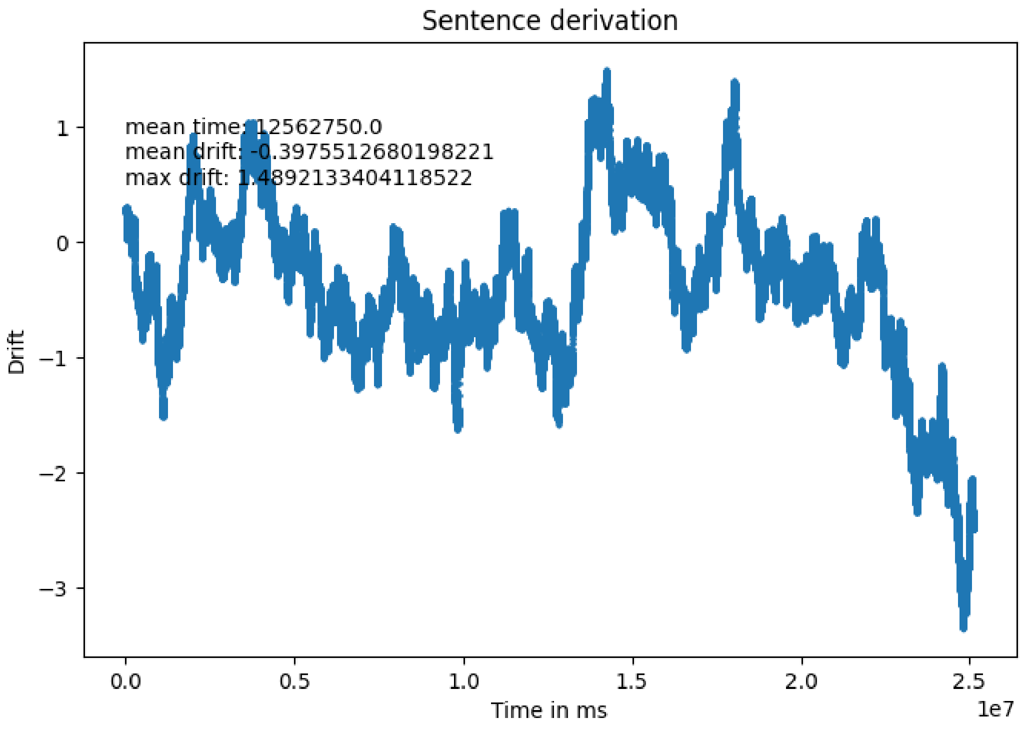
(See next pages)

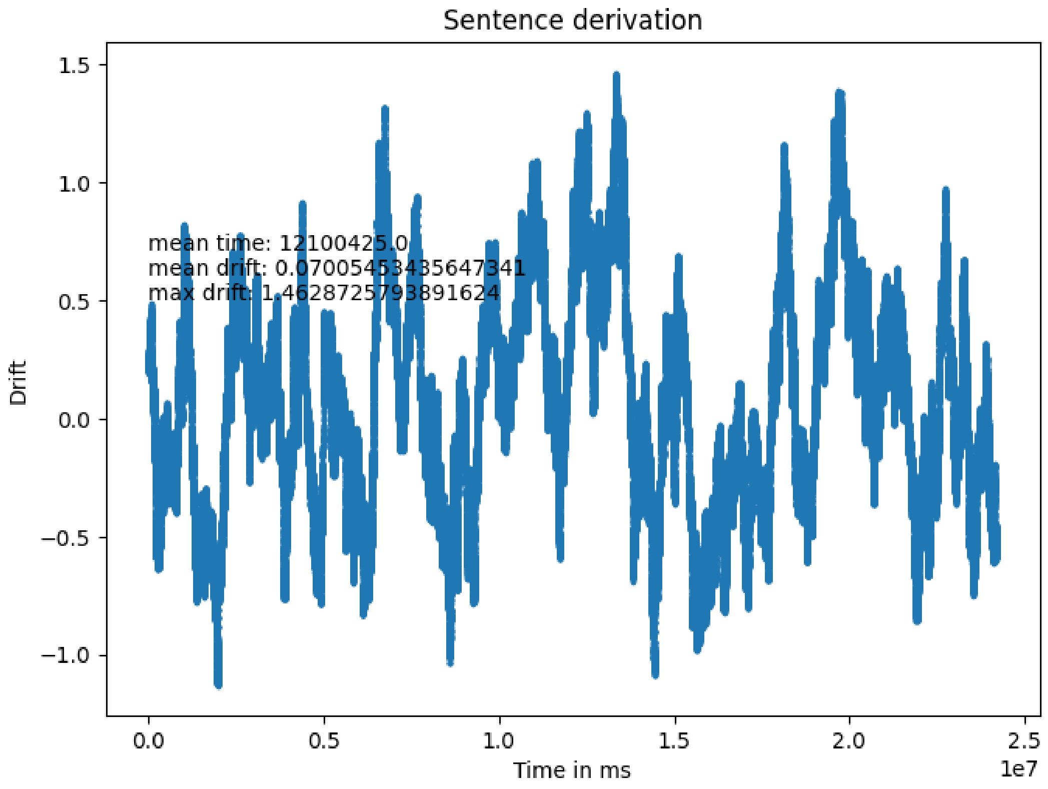

Sentence derivation
Drift
3
2
1
0
−1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1e7
Time in ms
mean time: 8220550.0
mean drift: 0.23384606374407418
max drift: 3.282665335687218

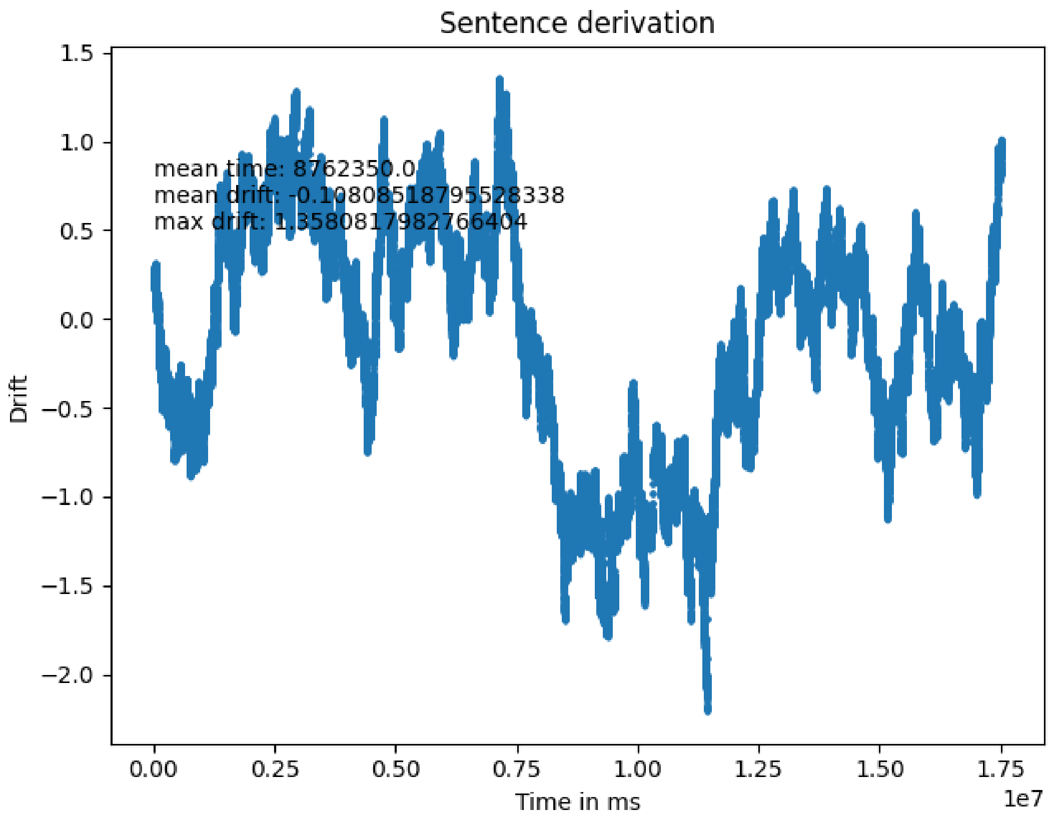

Sentence derivation
Drift
3
2
1
0
−1
−2
0.0
0.5
1.0
1.5
2.0
1e7
Time in ms
mean time: 11625050.0
mean drift: 0.25367328572528164
max drift: 3.004208527427634

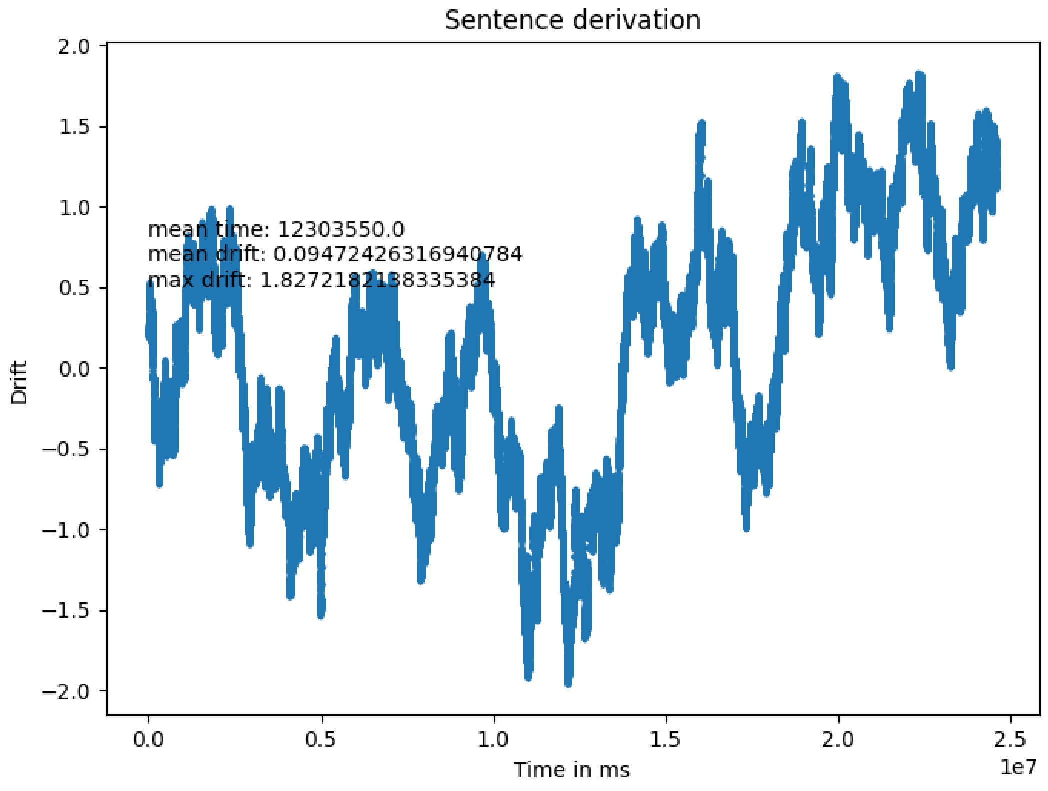

Sentence derivation
mean time: 9088950.0
mean drift: 0.07268187823911088
max drift: 1.2723836756120859
Drift
Time in ms
1e7
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75

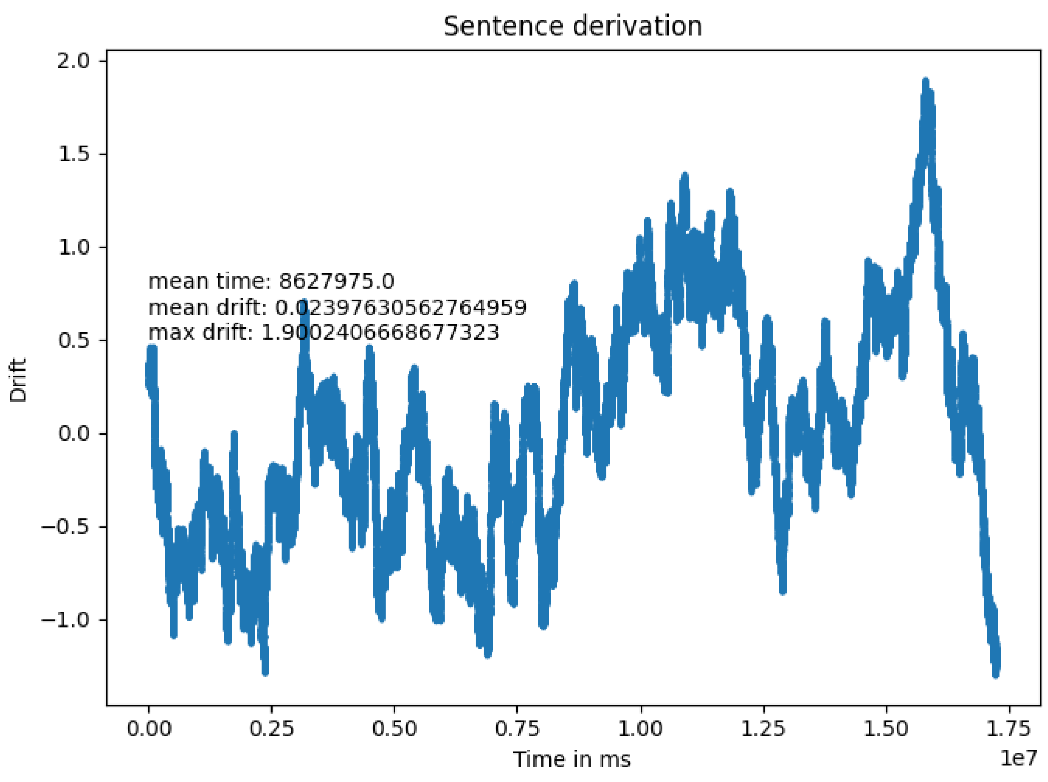

Sentence derivation
mean time: 11170600.0
mean drift: 0.2395804693487832
max drift: 2.5528385957526543
Drift
Time in ms
1e7
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
0.5
1.0
1.5
2.0


Sentence derivation
mean time: 12562750.0
mean drift: -0.3975512680198221
max drift: 1.4892133404118522
Drift
Time in ms
1e7
1
0
−1
−2
−3
0.0
0.5
1.0
1.5
2.0
2.5

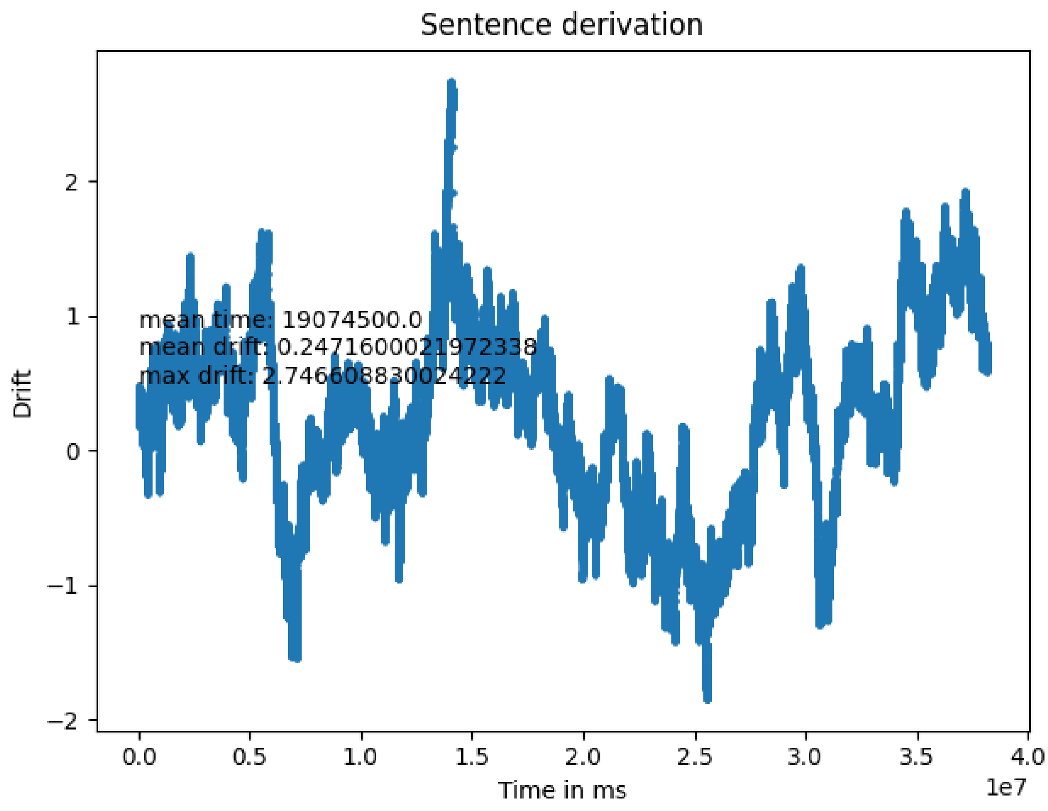

Sentence derivation
mean time: 12100425.0
mean drift: 0.07005453435647341
max drift: 1.4628725793891624
Drift
Time in ms
1e7
1.5
1.0
0.5
0.0
−0.5
−1.0
0.0
0.5
1.0
1.5
2.0
2.5


Sentence derivation
mean time: 8762350.0
mean drift: -0.10808518795528338
max drift: 1.3580817982766404
Drift
Time in ms
1e7
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75


Sentence derivation
mean time: 12303550.0
mean drift: 0.09472426316940784
max drift: 1.8272182138335384
Drift
Time in ms
1e7
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0
0.5
1.0
1.5
2.0
2.5



Sentence derivation
mean time: 8627975.0
mean drift: 0.02397630562764959
max drift: 1.9002406668677323
Drift
Time in ms
1e7
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75



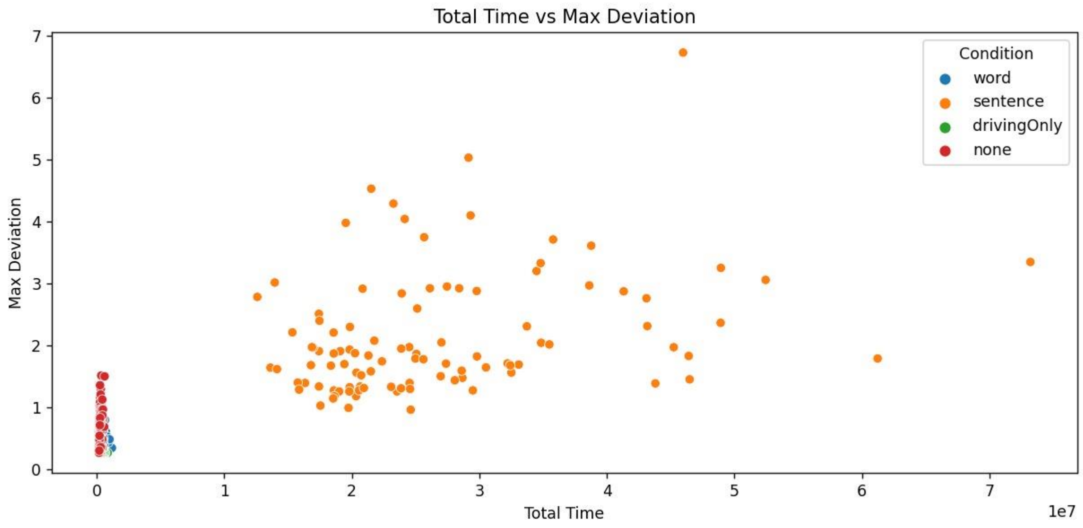
Sentence derivation
mean time: 19074500.0
mean drift: 0.2471600021972338
max drift: 2.746608830024222
Drift
Time in ms
1e7
2
1
0
−1
−2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

<u>2C: Explain what you see in the Figures. Specifically: i. How does the pattern of the different strategy show itself in the Figures? (e.g., how does this differ between the file you submitted for A and the file you submitted for B) ii. Within one type of strategy: what causes that each individual plot is different from the others? What causes these differences?</u>

In the sections where the trial is set to “word” we can see that the deviations are closer together and more sporadic. In the sections where the user is correcting after each sentence the time before each steering update is larger. For both strategies, this is caused by the amount of time needed to finish a sentence than it is to finish a word and then steer, as well as the lateral velocity calculated by both vehicleUpdateActiveSteering() and vehicleUpdateNotSteering(), which differ slightly every time they’re calculated.

Blackboard question 3:

<u>3A (Copy screenshot or picture of your plot below): Submit a picture or screenshot of the plot that you generated (in the function "runSimulations") which shows the predictions of individual trials (horizontal axis: total trial time, vertical axis: max lateral deviation) and their mean performance + standard deviation (see above engineering question). This should be a plot for 100 simulations per condition (so 400 simulations total). If there are grounds why your computer could not do this, submit a plot based on fewer trials (e.g., 50 trials).</u>



<u>3B: Explain using your plot from question A whether you would recommend drivers to interleave after every word or after every sentence. In your explanation make sure to refer to patterns that can be seen in the Figure, such as average performance in a condition, variability within that condition, and how the strategies impact both time for writing an e-mail and maximum deviation of the car.</u>

A footnote should be placed by our graph, that it may not be completely reliable (we believe there might be a flaw, as all but the sentence condition are located in the bottom left corner). However, the highest amount of deviation should come where the setting is set to 'none', as the user is only ever typing, never asserting control over the car. Next, it'd be the sentence condition, where the user steers after every sentence, asserting some but still not a lot of control. Next is the 'words' condition. While still unsafe, this condition would have more control and thus be saver. The best score would of course be the 'drivingOnly' condition, as the user does not look at their phone, and only drives.

However, if we have to pick: we would recommend the driver to interleave after every word over after every sentence. In this case there'd be the best average performance and least variability (and thus a lower maximum deviation of the car). However, the 'words' condition does have a higher total time for writing the email. Due to safety, however, we think the user should accept that consequence!

3C: The model could be expanded in many ways. One aspect to consider is that people do not always have their phones in their hands, but sometimes further away, such as on their lap or in a cradle. They would then have to reach for the phone before typing, and reach back to the steering wheel afterwards. Explain if and how such reaching behavior (a motor "switch cost") impacts each of the four strategies that you simulated (none, drivingOnly, word, sentence). You do not need to implement this model, but rather explain what pattern you would expect and why

In this case the user would need to make more motions to grab the phone again, recall the the sentence being typed and the current word. We would expect this would make the biggest difference for the strategy 'words' as this is the one that switches the most. The Switch cost would be higher, meaning it'd take more time to switch and thus create an even larger drift. The strategy 'sentences' would be slightly less influenced (although still influences in the same way), whereas the strategies 'drivingOnly' and 'none' would not be influenced at all by this change, as they do not switch.

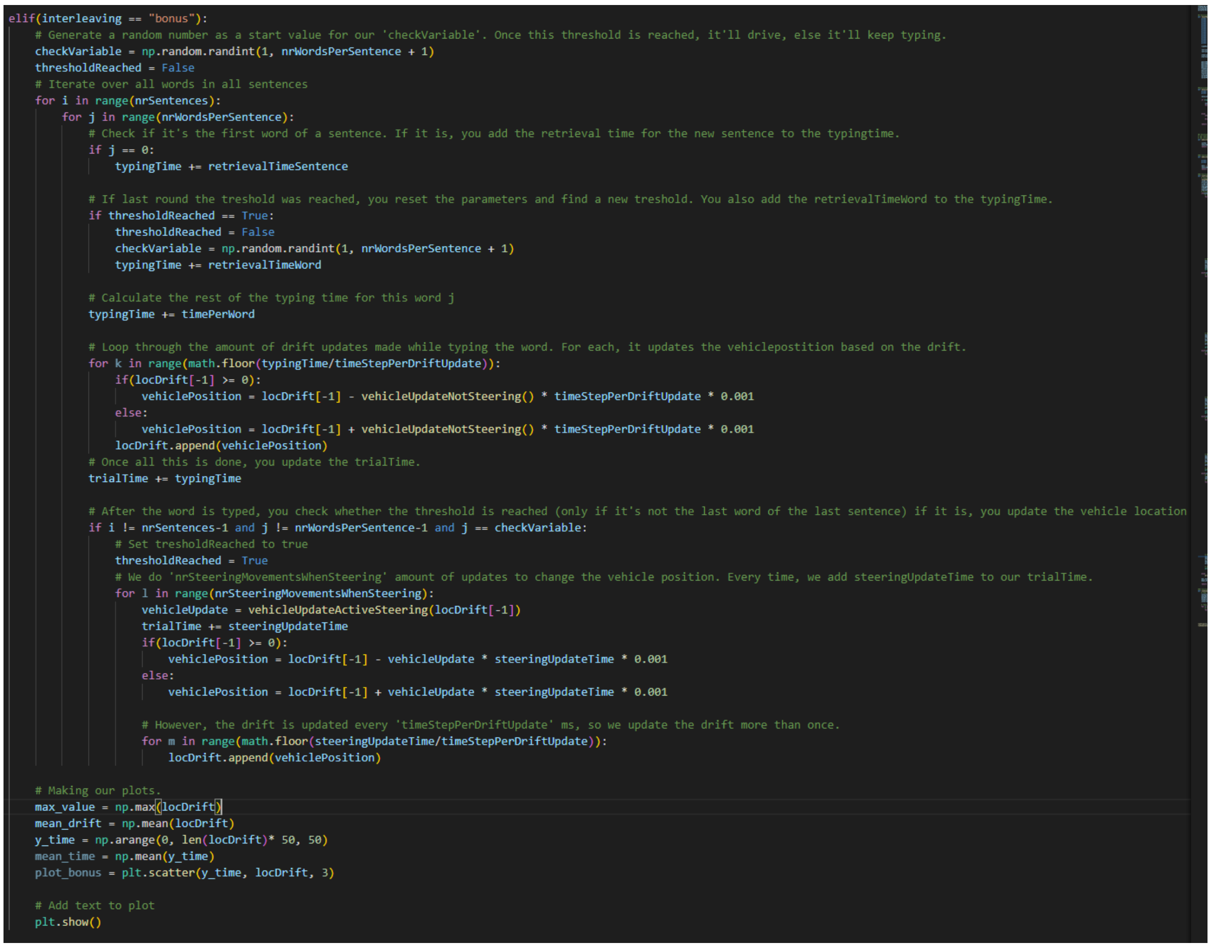
Section 4: Bonus question

If you complete a bonus question (optional), please answer the questions of the assignment below.

1.What scenario did you model (interleaving after X words, or error making)

Interleaving after X words. We implemented it where it will interleave after a random amount of words, where this random number is at least 1 and at most equal to the maximum amount of words in this sentence. We chose to implement it where it interleaves after a random amount of words because this would be most realistic (and thus would create a better model of reality). A person might return to steering after every word or every sentence as these are logical cutoffs, but a certain number of words would not be a logical cutoff. A person would be more likely to suddenly remember they were driving or return to driving due to outside influences (a car horn or people yelling), which would be after a random amount of words. We chose to make it scale from 1 to the total number of words, so they'd always still check at least once every sentence. We started at 1 instead of 0, because else the person could switch before having typed a single word (since either the beginning or last check).

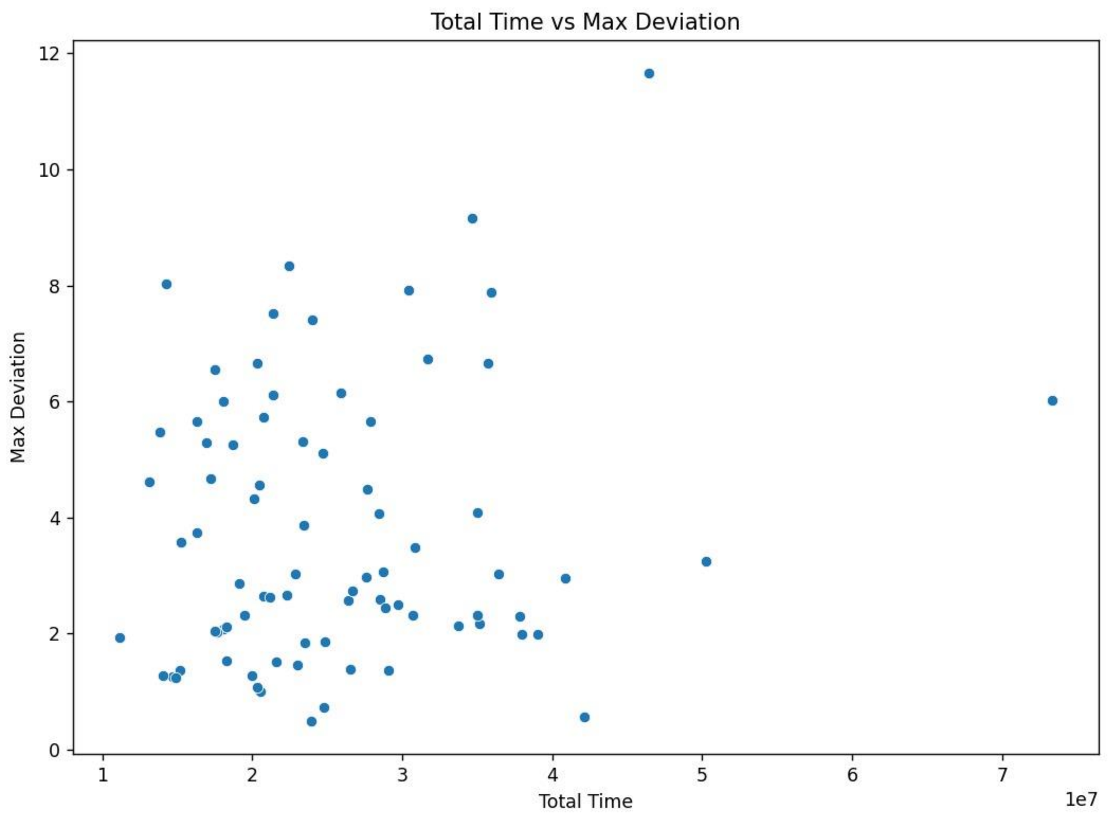
2.Submit a screenshot or PDF of the critical part of your code



```python
elif(interleaving == "bonus"):
    # Generate a random number as a start value for our 'checkVariable'. Once this threshold is reached, it'll drive, else it'll keep typing.
    checkVariable = np.random.randint(1, nrWordsPerSentence + 1)
    thresholdReached = False
    # Iterate over all words in all sentences
    for i in range(nrSentences):
        for j in range(nrWordsPerSentence):
            # Check if it's the first word of a sentence. If it is, you add the retrieval time for the new sentence to the typingtime.
            if j == 0:
                typingTime += retrievalTimeSentence

            # If last round the treshold was reached, you reset the parameters and find a new treshold. You also add the retrievalTimeWord to the typingTime.
            if thresholdReached == True:
                thresholdReached = False
                checkVariable = np.random.randint(1, nrWordsPerSentence + 1)
                typingTime += retrievalTimeWord

            # Calculate the rest of the typing time for this word j
            typingTime += timePerWord

            # Loop through the amount of drift updates made while typing the word. For each, it updates the vehiclepostition based on the drift.
            for k in range(math.floor(typingTime/timeStepPerDriftUpdate)):
                if(locDrift[-1] >= 0):
                    vehiclePosition = locDrift[-1] - vehicleUpdateNotSteering() * timeStepPerDriftUpdate * 0.001
                else:
                    vehiclePosition = locDrift[-1] + vehicleUpdateNotSteering() * timeStepPerDriftUpdate * 0.001
                locDrift.append(vehiclePosition)
            # Once all this is done, you update the trialTime.
            trialTime += typingTime

            # After the word is typed, you check whether the threshold is reached (only if it's not the last word of the last sentence) if it is, you update the vehicle location
            if i != nrSentences-1 and j != nrWordsPerSentence-1 and j == checkVariable:
                # Set tresholdReached to true
                thresholdReached = True
                # We do 'nrSteeringMovementsWhenSteering' amount of updates to change the vehicle position. Every time, we add steeringUpdateTime to our trialTime.
                for l in range(nrSteeringMovementsWhenSteering):
                    vehicleUpdate = vehicleUpdateActiveSteering(locDrift[-1])
                    trialTime += steeringUpdateTime
                    if(locDrift[-1] >= 0):
                        vehiclePosition = locDrift[-1] - vehicleUpdate * steeringUpdateTime * 0.001
                    else:
                        vehiclePosition = locDrift[-1] + vehicleUpdate * steeringUpdateTime * 0.001

                    # However, the drift is updated every 'timeStepPerDriftUpdate' ms, so we update the drift more than once.
                    for m in range(math.floor(steeringUpdateTime/timeStepPerDriftUpdate)):
                        locDrift.append(vehiclePosition)

    # Making our plots.
    max_value = np.max(locDrift)
    mean_drift = np.mean(locDrift)
    y_time = np.arange(0, len(locDrift)* 50, 50)
    mean_time = np.mean(y_time)
    plot_bonus = plt.scatter(y_time, locDrift, 3)

    # Add text to plot
    plt.show()
```

3.Submit a plot of what performance now looks like. The plot should be in the style of Blackboard question 3, but adjusted to the scenario at hand.

(See next page)



<u>4.Explain how and why performance of the model has (not) changed compared to the strategies that you modelled before.</u>

The performance has changed compared to the strategies we modelled before. This is because the user interleaves less often than in the ‘words’ condition and more often than in the ‘sentence’ condition. The results are more condensed, meaning there’s less variability. There’s a lower deviation of the car than in the ‘sentences’ and ‘none’ conditions, though it is higher than in the ‘drivingOnly’ and ‘sentences’ conditions.