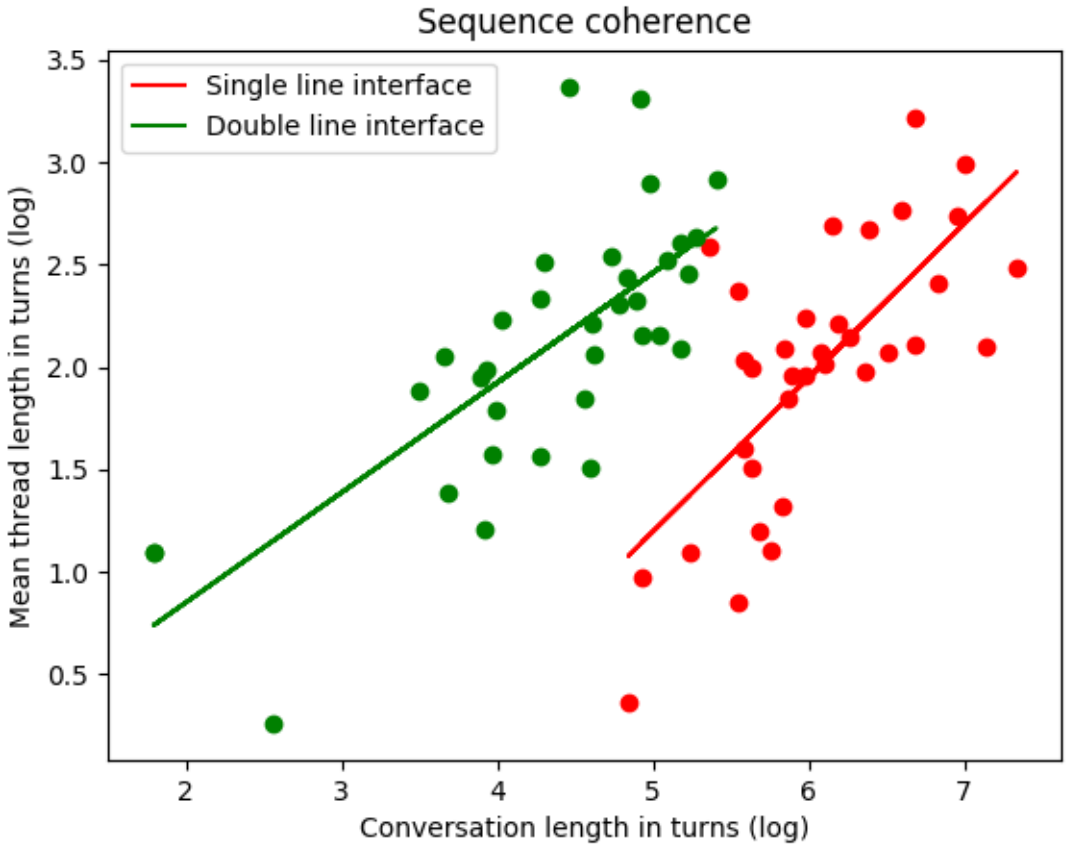
Social Media – Assignment 2: Quantitative

J.F.P. (Richard) Scholtens
s2956586

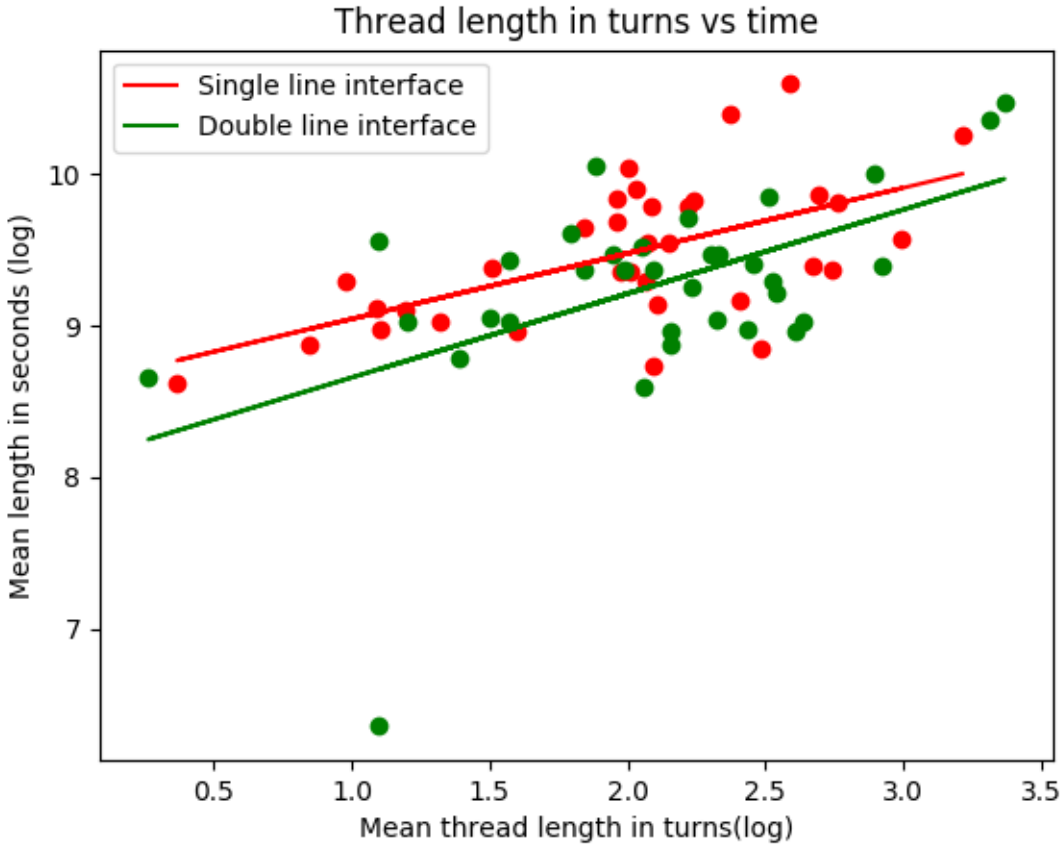
R.P. (Rolf) Daling
s2344343

Q1: The more number of turns it will take before an agreement is reached, the less efficient the program is. Therefore the mean thread length in turns and the conversation length in turns, are used to define the measure of efficiency.



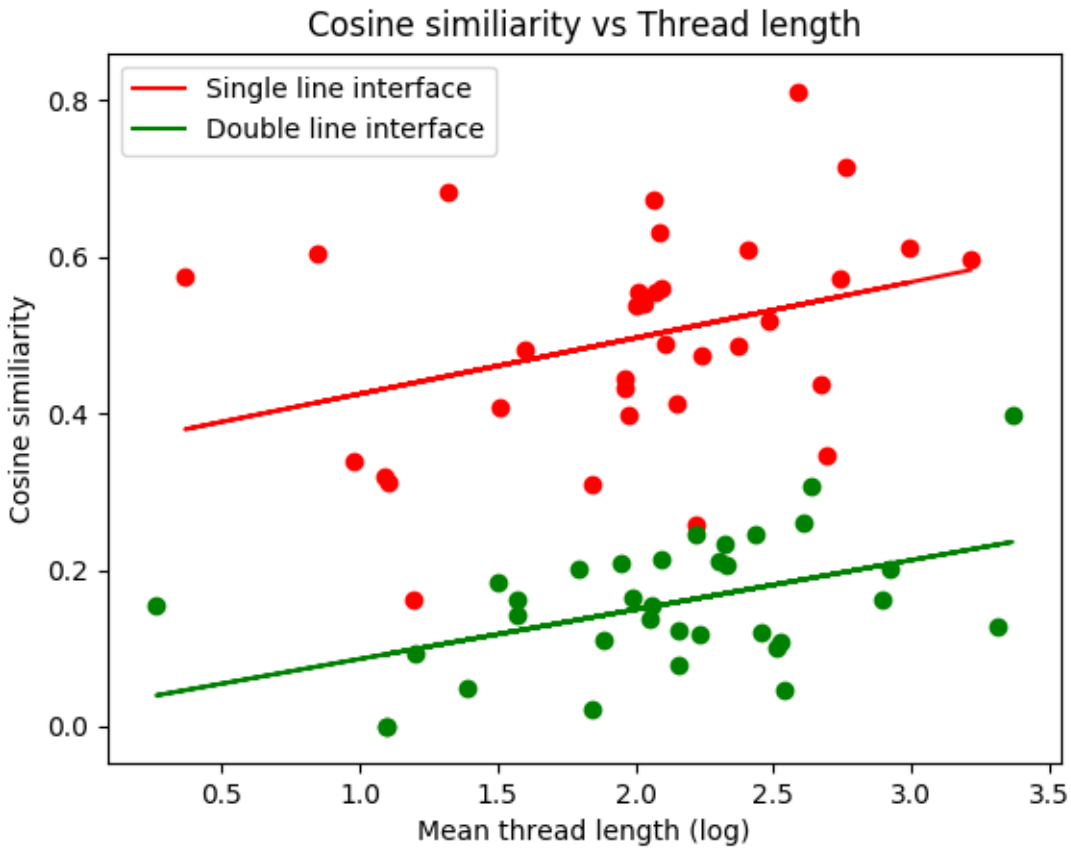
The graph shows a scatter plot of the log of the mean thread length and the log of the conversation length in turns. Since our data is skewed, we use the log on the data. The graph shows that the double line interface had less turns for each conversation, and the line plotted for the double line interface is less steep. The double line interface is therefore more efficient.

Q2: Better turn-taking can be defined as having more and longer turns. When someone takes more time to formulate their answer, and has more turns in a single thread, that would mean he/she is better at turn-taking.



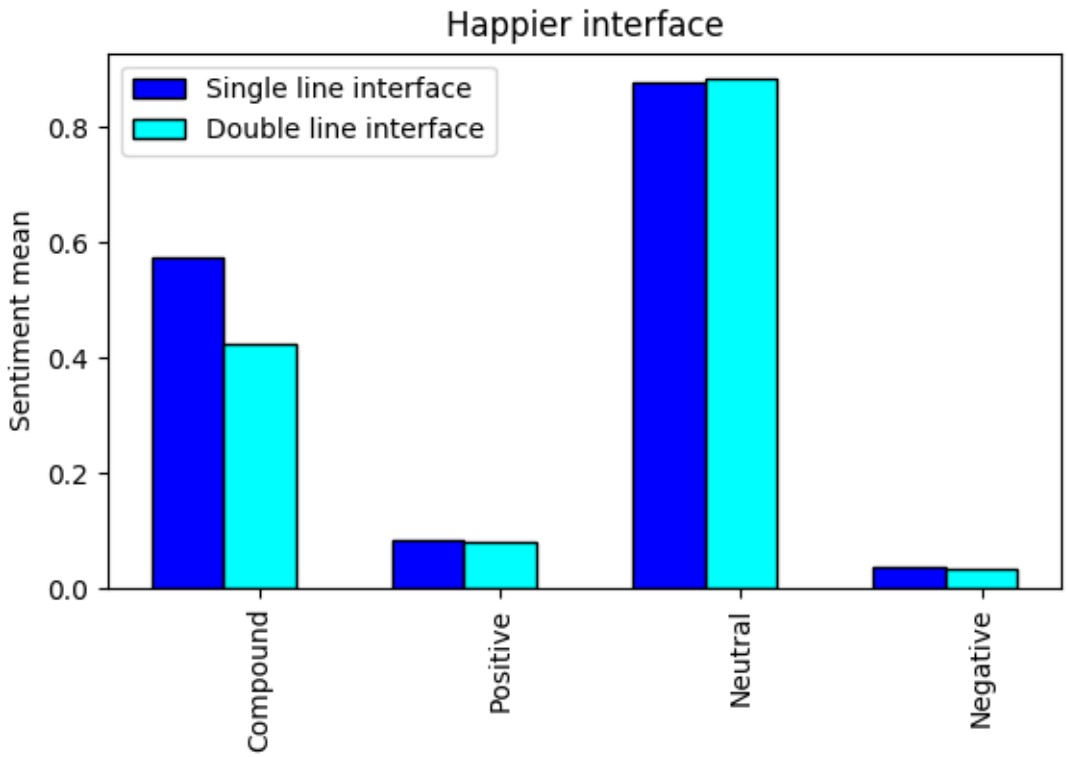
The graph shows the log of the mean thread length in turns versus the log of the mean thread length in seconds. The linear fit line show that the single line interface has better turn-taking. This is as expected, since the single line interface forces one to take turns.

Q3: Language copying can be measured by calculating the cosine similarity on both the ‘s’ and ‘o’ conversation.



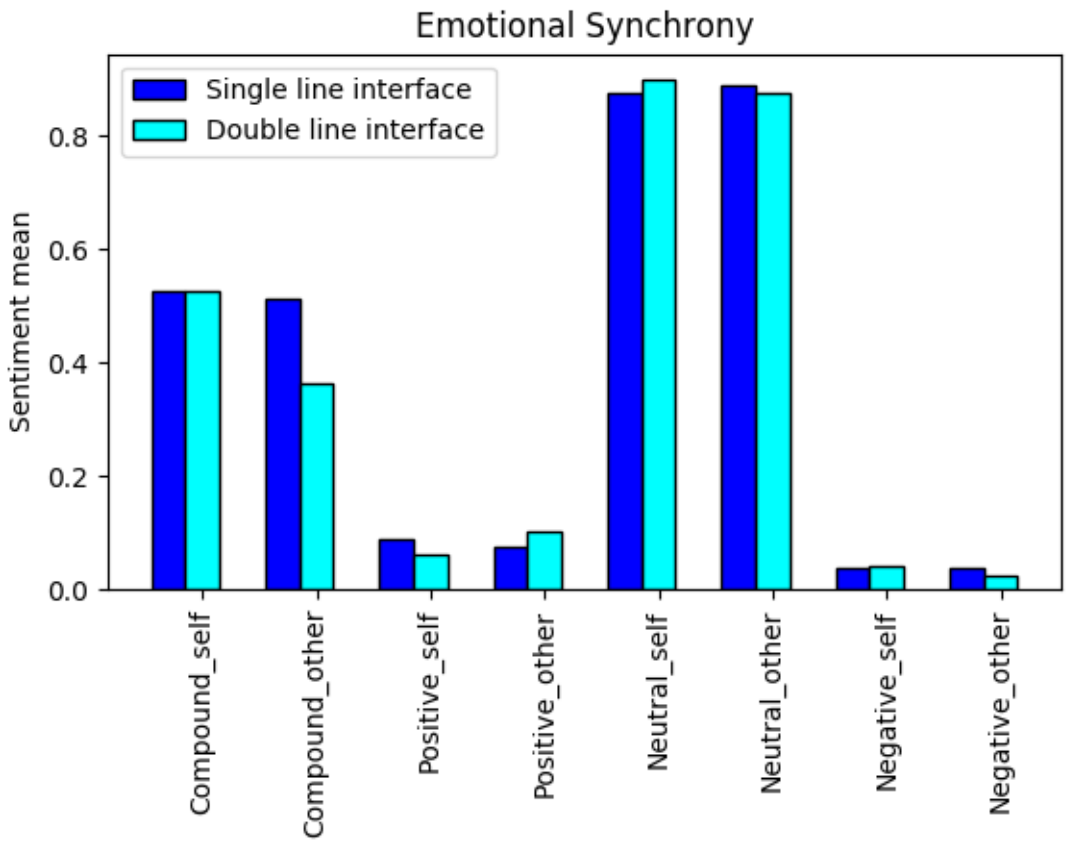
This graph shows the cosine similarity calculated on both parts of the conversation versus the log of the mean thread length. The data shows that the single line interface has higher cosine similarity scores, which means that on that interface the language is more copied.

Q4: Happiness in a conversation can be defined using sentiment analysis. When the sentiment is more positive, or less negative, that means that it is happier.



The bar graph shows that the sentiment compound is higher for the single line interface. Also the positive is slightly higher, while the negative sentiment is also a tad bit higher. Therefor we could say that the single line interface is slightly happier.

Q5: Emotional synchrony can be defined using sentiment analysis. If the values for the sentiment analysis for both participants are closer to each other, that means that the participants are emotionally synchronous.



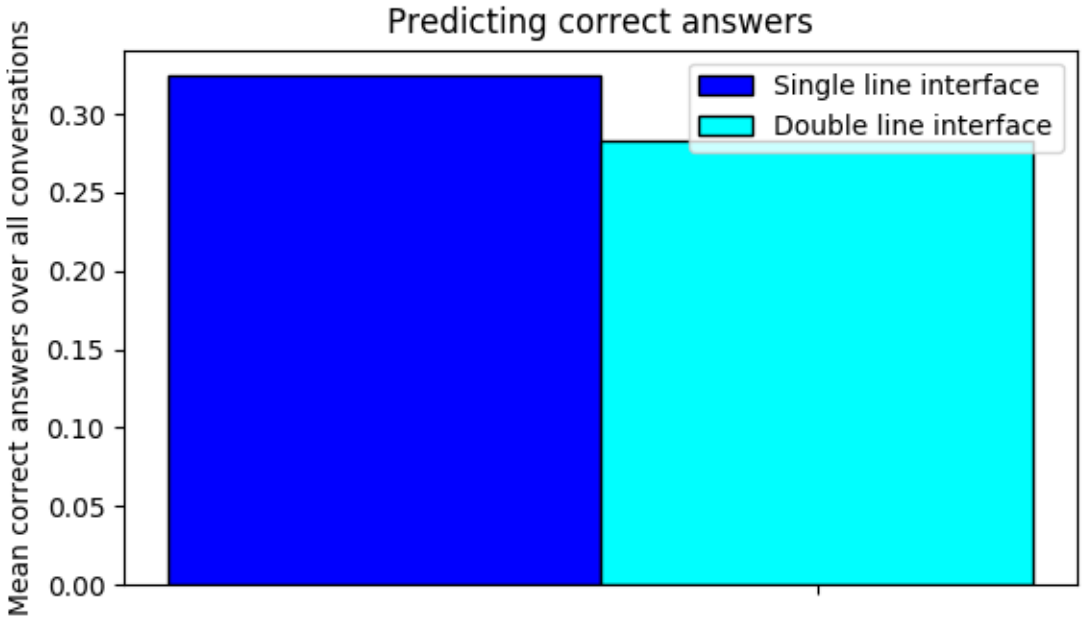
The sentiment values for both interfaces and for both parties are displayed in this bar graph. In the graph we observe that each of the sentiment values of the single line interface are closer to each other than the ones for the double line interface.
This is as expected, since the single line interface asks for better coordination of the participants, and therefor are more emotional synchronous.

Q6: Once again we can use sentiment analysis. In this case we can use it to try to determine which of the choices were correct. If a thread is more positive, or less negative, that may mean that the previous choice was correct.
To determine this, we need to split each conversation into threads.

```
def thread_splitter(matrix):
    """ Splits the conversation into single threads. Each thread is seperated
        by a slash followed by either a 'd' or a 's'. So we look at the
        previous and current character of both conversationists, and find the
        index. Then we split every array of the matrix at this index and append
        it to the list.
        The result is a list of matrices for every thread. """
    thread_matrix = []
    thread = []
    thread_start = 0
    prev_s = None
    prev_o = None
    sd = ['s', 'd']
    for i in range(len(matrix[0])):
        s = matrix[0][i]
        o = matrix[1][i]
        if prev_o == '/' and o in sd or prev_s == '/' and s in sd:
            thread_end = i
            for j in range(len(matrix)):
                thread.append(matrix[j][thread_start+1:thread_end-1])
            thread_matrix.append(thread)
            thread = []
            thread_start = thread_end
        prev_s = s
        prev_o = o
    for i in range(len(matrix)):
        thread.append(matrix[i][thread_start:])
    thread_matrix.append(thread)
    return thread_matrix
```

The thread_splitter() function splits the conversation lists in separate threads. Then we can determine the sentiment for every thread. We then use a counter to find out how often the positive value exceeds the negative value.

```
def sentiment_threads(matrix):
    """ Splits the text into threads, gets the text for each thread of both
        parties, finds the sentiment for each thread and counts how often the
        positive value is higher than the negative value.
        Returns the number of times it was positive, and the number of threads.
        """
    thread_matrix = thread_splitter(matrix)
    s_conv = ""
    o_conv = ""
    pos_counter = 0
    for thread in thread_matrix:
        s_conv, o_conv = get_text(thread)
        conv = s_conv + " " + o_conv
        comp, pos, neu, neg = sentimentfinder(conv)
        if pos > neg:
            pos_counter += 1
    return (pos_counter, len(thread_matrix))
```



This bar graph shows the mean of the ‘correct’ answers over all the conversations. It shows that the single line interface has a slightly higher rate of ‘correct’ answers. The participant needed to cooperate better, and that can lead to more ‘correct’ answers.