Child Language Analyser

FIT9133 - Python Programming

Roopak Thiyyathuparambil Jayachandran
Student Id : 29567467

Assignment #2

Table of Content:

Introduction:

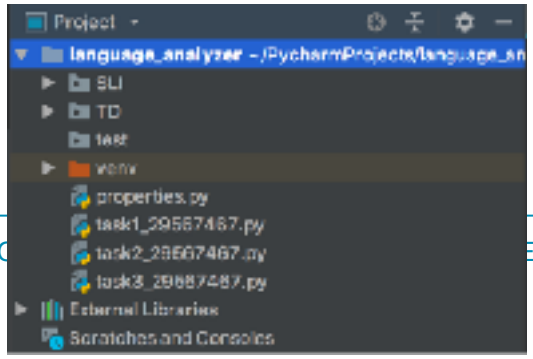
Child Language Analyser is a Python project implemented to investigate the linguistic characteristics of children with some form of language disorder. The program includes data cleansing, data analysis and data visualisation. It is divided into 3 parts which is divided into 3 different files.

Task 1: Handling with File Contents and Preprocessing

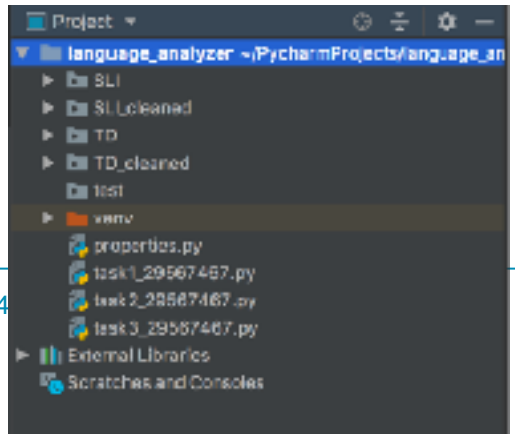
First phase of the project ,i.e, data cleansing is implemented in task1_29567467.py.
Program includes a function called file_wrangling() which takes a list of files as input, extracting lines starting with "*CHI :" and finding patterns which needs to be removed or changed for data analysis.
After cleaning of data, the data is written to a new file with the same name but in new folder, SLI_cleaned or TD_cleaned based on the type of input file.
So in order to process the data, the user has to put files in SLI or TD folder. If the folders (SLI_cleaned or TD_cleaned) are already present, it will not create new folder. **OS** module is used to list and make new directories in the project structure.
Python code to execute the function:

```
input_files1 = os.listdir("TD")        # Lists all the files in TD folder
t1.file_wrangling(input_files1)        # Passed as an argument in file_wrangling function
                                       # After execution : TD_cleaned folder is created with
                                         all cleaned files in TD
```

Before



After



Task 2: Building a Class for Data Analysis

Second Phase of the project is implemented using class DataAnalyzer. In this class two instance variables and 5 modules have been defined.

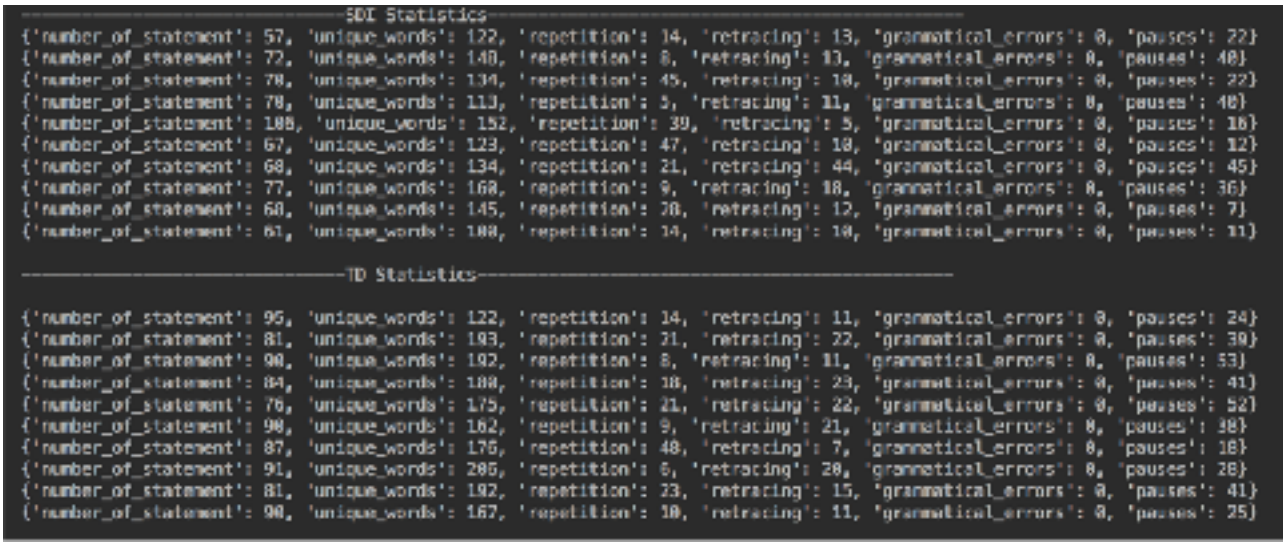
Instance variables are **statistics_SLI** and **statistics_TD** which are list of dictionaries. Getters are defined for these two instance variables.

External package used : **os**

```
def __str__(self):
```

```
# Overrides the str method to print Instance variables in proper format
# Returns : A string(containing lists) in proper format to print
```

```
d = DataAnalyzer()                      # Initialises an object
d.analyse_script("SLI_cleaned").        # Specifying the cleaned directories for
d.analyse_script("TD_cleaned")          # analysis
print(d)                                # Printing of object
```




```
def analyse_script(self, dir_name):
```

```
# analyse_script function iterates through all the files in the user mentioned folder,
extracts repetition, grammatical error, pauses, unique count and retraces from each of the
file and saves as a list of dictionaries in Instance variables
# Arguments : dir_name - Directory name
```

Patterns considered for counting characteristics are as below:
[/] - repetition
[/ /] - retracing

[* m:+ed] - grammatical errors
(.) - Pauses
Unique words are calculated by passing the values to a set() and getting the count of the set.
Number of statements is taken as the length of the input file as in Task 1, we had appended all multiple lined strings starting with "*CHI" to a single line.
After calling analyse_script function, if we print the object, it will show the count of all characteristics in each file as shown in the screenshot above

Task 3: Building a Class for Data Visualisation

In the third phase of implementation, we have created a Visualisation class with 4 functions which will Visualise the cleaned and analyse data into 7 data visual representation which the User can select.

Packages used : **os, pandas, matplotlib, prettytable**

Instance modules defined :

```
def __init__(self):
```

Init module of Visualise class does the cleaning and data analysis by itself so that we can directly start with the visualisation. For this it does the following:

- Use os module to get the list of all files inside TD and SLI
- Pass the list as a parameter for file_wrangling function imported from task1 (Cleaned folders will be created after this execution)
- Create an object of DataAnalyzer class, call analyse_script passing the Cleaned file names as parameter. Store the list of dictionaries of SLI and TD in instance variables of Visualise and create data frames from them.

So in order to run the complete project, just create an object from Visualise class and call visualise_statistics function

```
> v = Visualise()
> v.visualise_statistics()
```

```
def print(self):
```

Argument : None
This module will print the dictionaries from SLI and TD in tabular form using pretty print

SLI Data For Visualisation

number_of_statement	unique_words	repetition	retracing	grammatical_errors	pauses
57	122	14	13	1	22
72	148	8	13	0	40
70	134	45	10	0	22
70	113	5	11	2	40
106	152	39	5	0	16
67	123	47	10	1	12
68	134	21	44	0	45
77	160	9	18	0	36
68	145	28	12	0	7
61	100	14	10	0	11

TD Data For Visualisation

number_of_statement	unique_words	repetition	retracing	grammatical_errors	pauses
95	122	14	11	0	24
81	193	21	22	1	39
90	192	8	11	0	53
84	180	18	23	0	41
76	175	21	22	0	52
90	162	9	21	0	38
87	176	48	7	0	18
91	206	6	20	0	28
81	192	23	15	0	41
90	167	10	11	0	25

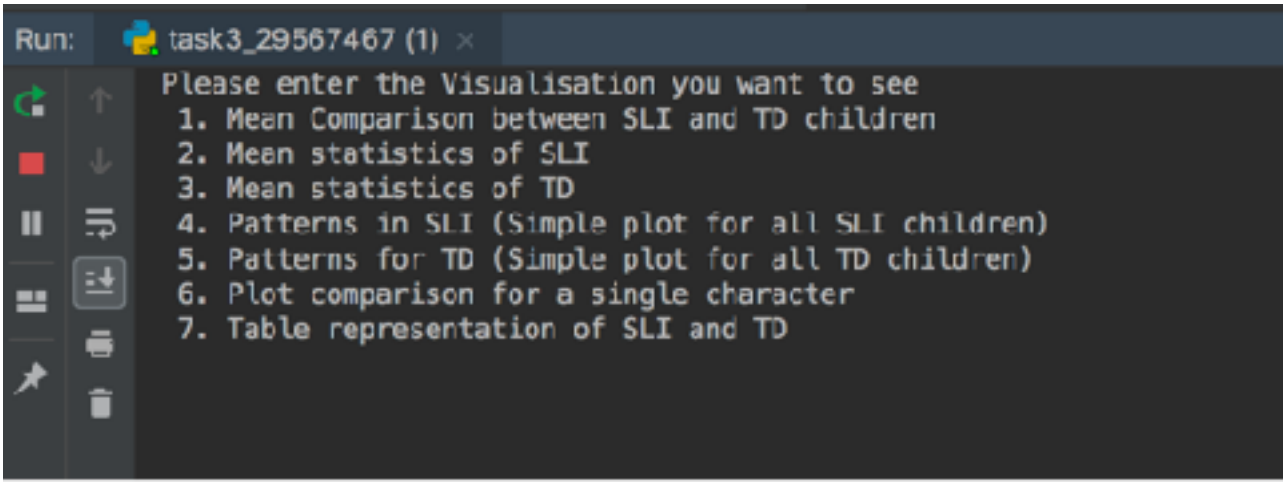
def computer_averages(self):

Argument : None
This module is used to compute average value from each of the column representing a characteristic. Since the data is stored in pandas.DataFrame data structure, column.mean() will return the mean for that column. This variable will be stored in the instance variables of the Visualise class and will be used for visualisation.

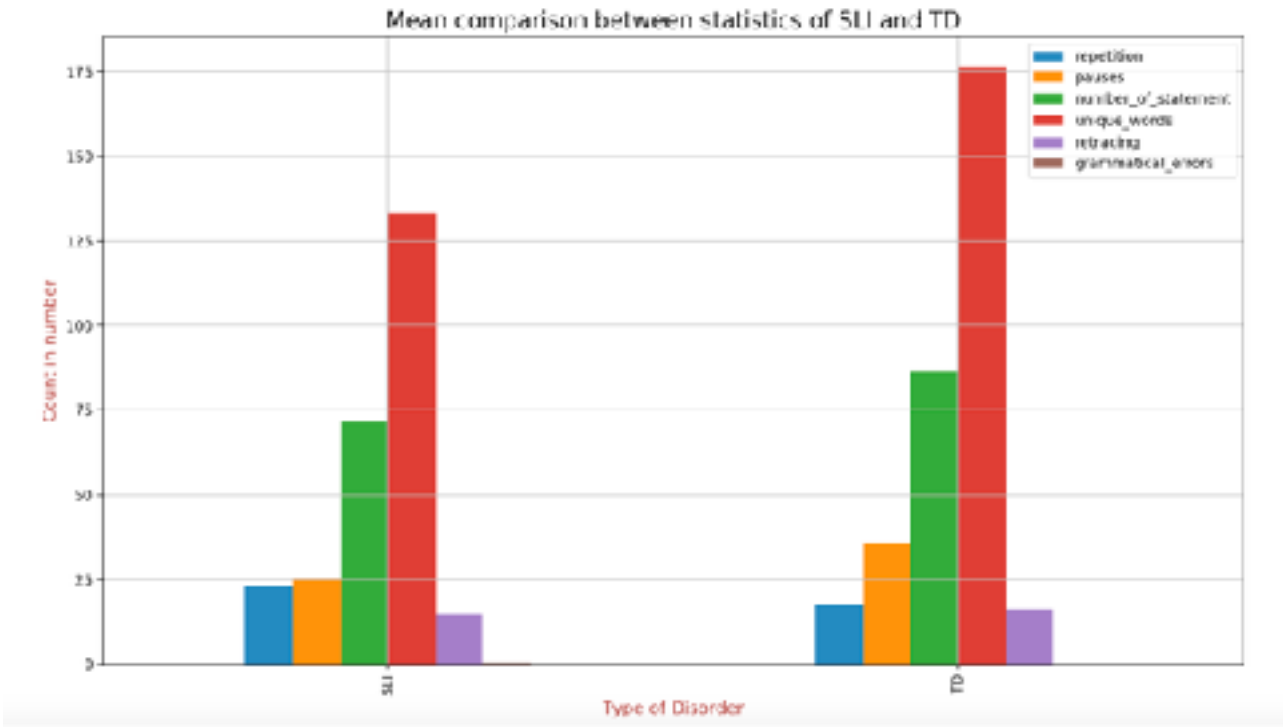
def visualise_statistics(self):

visualise_statistics function is used to visualise the statistics to get some insights. This function follows a menu-driven approach, taking input from the user on the type of visualisation they want to have.
Argument : None
Return : Plot and print functions

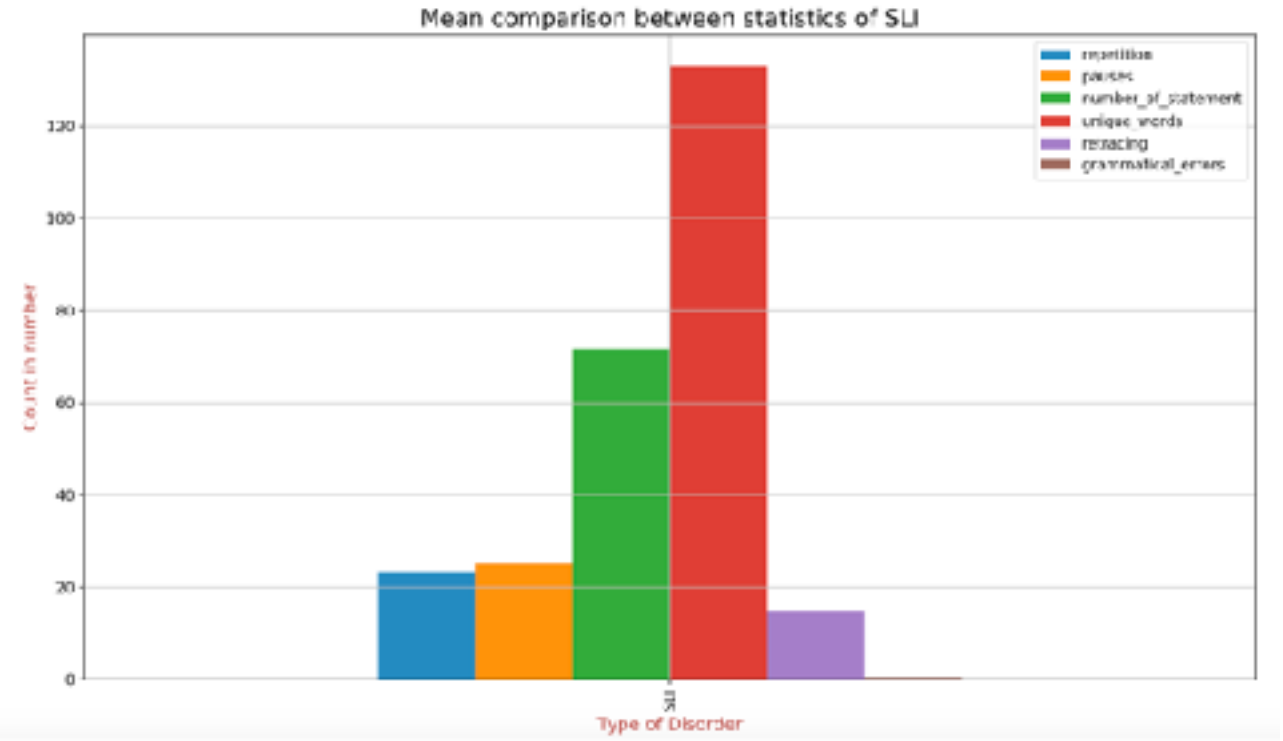
On creating an instance from this class and calling instance.visualise_statistics function, User will get many option for visualisation as shown in screenshot below:



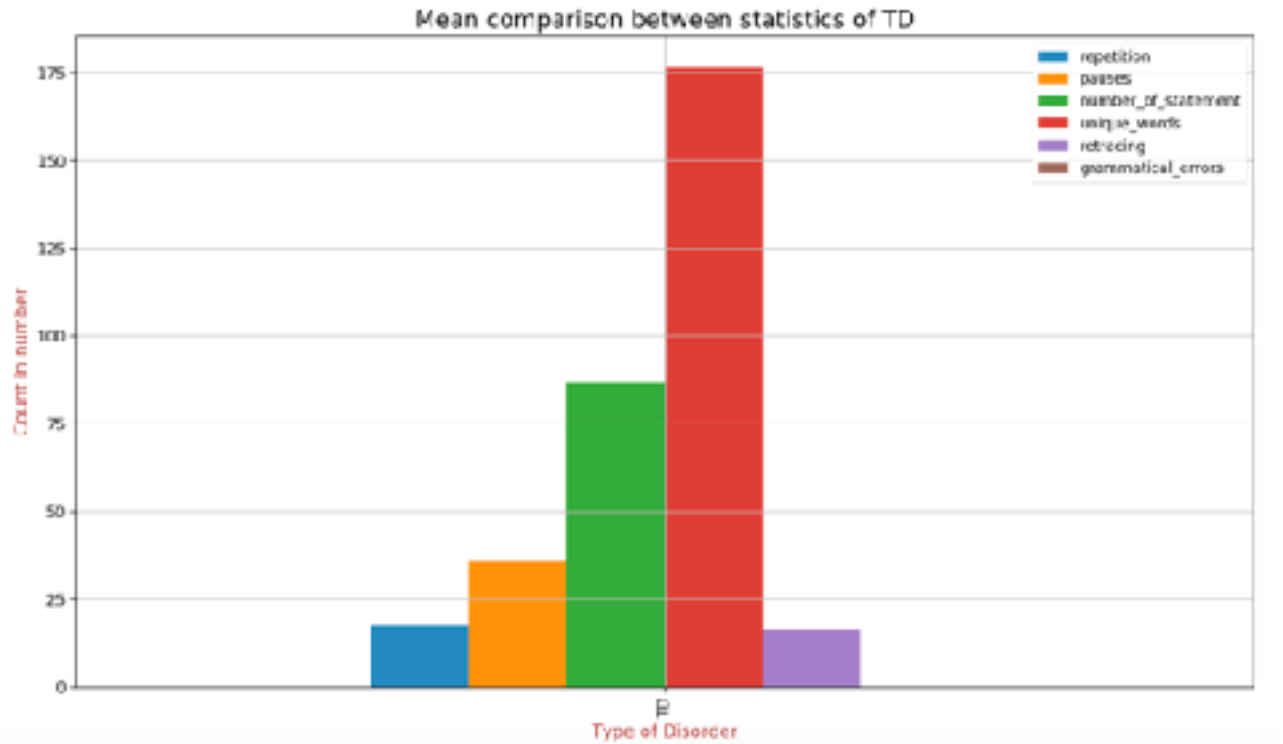
1. Mean Comparison between SLI and TD children (using bar-chart)



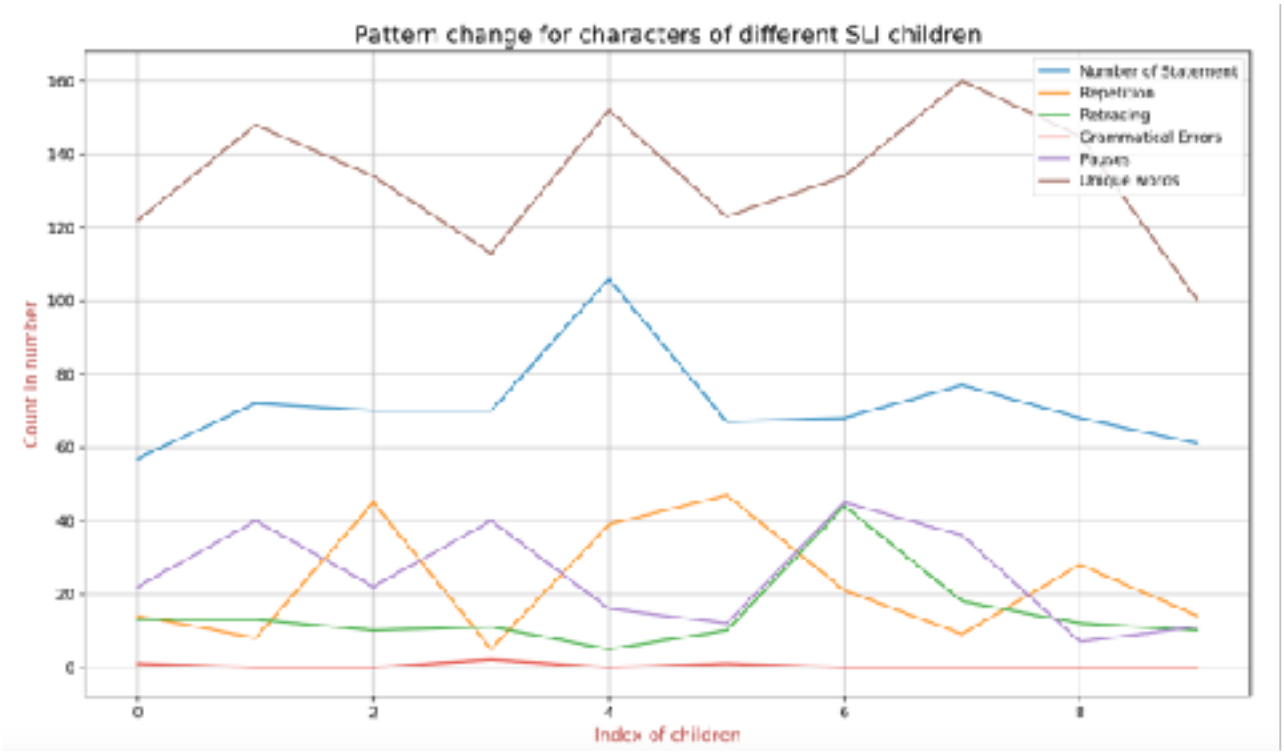
2. Mean statistics of SLI



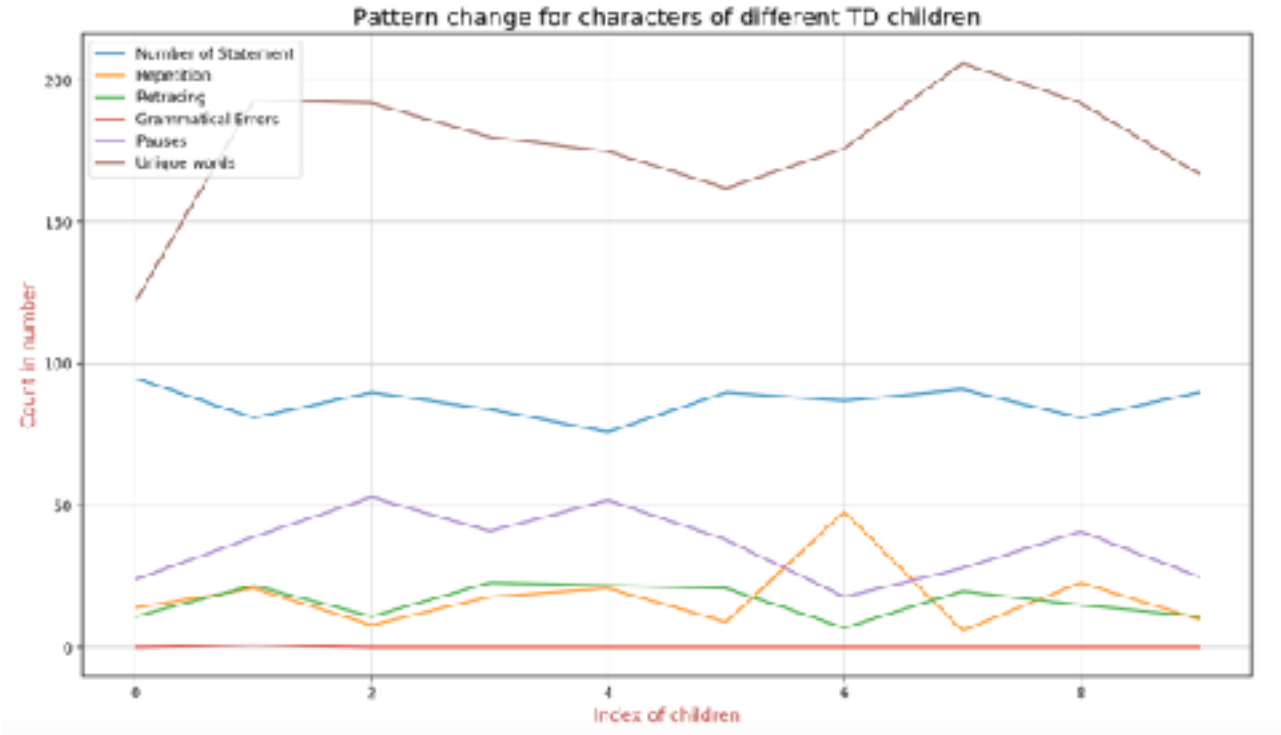
3. Mean statistics of TD



4. Patterns in SLI (Simple plot for all SLI children)

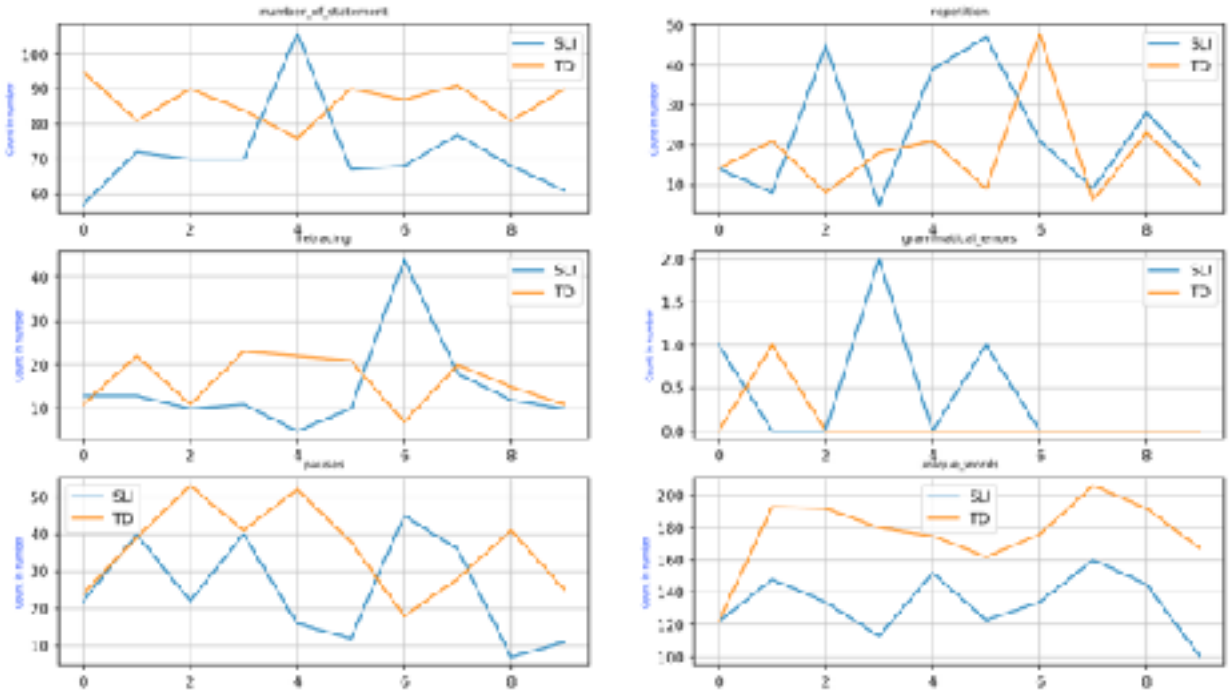


5. Patterns in TD (Simple plot for all TD children)



7. Table representation of SLI and TD - Already shown in screenshot in page 4

6. Plot comparison for a single character - Here user can select the individual comparison for SLI or TD or even can see all the plots together in a single plot like below



If User inputs any other character/number, the program will terminate will an error message.

Steps to run the program

- Put the raw files (SLI or TD) to the corresponding folder.
- Create an object of Visualise class
  > v = Visualise()
- Call visuals_statistics() function
  > v. visuals_statistics()

PS : In task3 file, these 2 steps have already written in the end. So just need to run task3 file