FIT9133 Assignment #2
Building a Child Language Analyser
Semester 2 2018

Dr Jojo Wong
Lecturer, Faculty of IT.
Email: `Jojo.Wong@monash.edu`

September 26, 2018

Revision Status

```
$Id: FIT9133-Assignment-02.tex, Version 1.0 2017/02/14 13:50 Jojo $
$Id: FIT9133-Assignment-02.tex, Version 1.1 2018/09/03 18:14 Gavin $
$Id: FIT9133-Assignment-02.tex, Version 1.2 2018/09/11 20:30 Jojo $
$Id: FIT9133-Assignment-02.tex, Version 1.3 2018/09/15 19:55 Jojo $
$Id: FIT9133-Assignment-02.tex, Version 1.4 2018/09/19 10:45 Jojo $
$Id: FIT9133-Assignment-02.tex, Version 1.5 2018/09/26 14:55 Jojo $
```

Contents

1 Introduction

This assignment is due on **12 October 2018 (Friday) by 17:00pm**. It is worth **20% of the total unit mark**. A penalty of 10% per day will apply for late submission. Refer to the FIT9133 Unit Guide for the policy on extensions or special considerations.

Note that this is **an individual assignment** and **must be your own work**. Please pay attention to Section 4.2 of this document on the university policies for the *Academic Integrity, Plagiarism and Collusion*.

This second assignment consists of **three tasks** and all the tasks should be submitted as **separate Python modules (classes or executable programs)** with supporting documentation on its usage. All the Python files and any supporting documents should be compressed into one single file for submission. (The submission details are given in Section 4.)

Assessment: Each of the three tasks has an equal weightage of marks.

2 The Assignment

In this assignment, you will implement a basic language analyser to investigate the linguistic characteristics of children with some form of language disorders. The analyser is able to perform basic statistical analysis on a number of linguistic features and also to present the analysis results using some form of visualisation.

2.1 The Dataset: ENNI

Before you get started with any of the programming tasks, you should read through the description of the dataset that we will be using for the purpose of this assignment.

The dataset is known as ENNI [`https://childes.talkbank.org/access/Clinical-MOR/ENNI.html`] which is a collection of narrative transcripts gathered for a clinical study carried out in Alberta, Canada, to study children with language disorders. Two sets of data were collected: the first set is from children diagnosed with Specific Language Impairment (SLI) — one form of language disorders; and the second set is from children with the typical development (TD). A subset of the original corpus is used in this assignment with 10 selected transcripts for each group of children.

Each of the narrative transcripts is a record of the story-telling task performed by a child for the two groups (SLI and TD), under the supervision of an examiner (investigator). The stories are elicited by presenting pictures with a number of different animal characters to the children participating in the study. See Figure 1 is an excerpt extracted from the transcript produced by a SLI child.

You should note that there are many details recorded in each of these transcripts. However, for the purpose of this assignment, the data required for processing and analysis is the narrative produced by the children, which are those statements (or lines) indicated by the label of '*CHI:' in the transcripts (as highlighted in the excerpt).

As a side note, the format of the transcripts is based on the CHAT Transcription Format. You may want refer to the CHAT manual [`http://talkbank.org/manuals/CHAT.pdf`] for the explanation of the various CHAT symbols, such as [//], [/], [*], (.), <...>, etc.

Note: You should download the dataset from the FIT9133 S2 2018 Moodle site before attempting the following tasks. The SLI transcripts are organised under the folder "SLI" while the TD transcripts are under the folder of "TD".

```
@UTF8
@PID:    11312/c-00036109-1
@Begin
@Languages:      eng
@Participants:   CHI Target_Child, EXA Investigator
@ID:     eng|ENNI|CHI|4;11.16|male|SLI||Target_Child|||
@ID:     eng|ENNI|EXA|||||Investigator|||
@Comment:        Birth of CHI is 9-MAY-1995
@Date:   25-APR-2000
@Tape Location:  Disk L10 Track 3
@Bg:     A1
*CHI:    I saw a giraffe and a elephant .
%mor:    pro:sub|I v|see&PAST det:art|a n|giraffe coord|and det:art|
a
         n|elephant .
%gra:    1|2|SUBJ 2|0|ROOT 3|4|DET 4|2|OBJ 5|4|CONJ 6|7|DET 7|5|
COORD 8|2|PUNCT
*CHI:    <that> [/] (.) that (i)s it . [+ bch]
%mor:    pro:dem|that cop|be&3S pro:per|it .
%gra:    1|2|SUBJ 2|0|ROOT 3|2|PRED 4|2|PUNCT
*CHI:    I saw an elephant go swimming .
%mor:    pro:sub|I v|see&PAST det:art|a n|elephant v|go part|swim-
PRESP .
%gra:    1|2|SUBJ 2|0|ROOT 3|4|DET 4|5|SUBJ 5|2|COMP 6|5|OBJ 7|2|
PUNCT
*CHI:    <I saw eleph> [//] I saw the <g> [/] giraffe and the
elephant <s>
         [//] drop ball in the pool .
%mor:    pro:sub|I v|see&PAST det:art|the n|giraffe coord|and
det:art|the
         n|elephant n|drop n|ball prep|in det:art|the n|pool .
%gra:    1|2|SUBJ 2|0|ROOT 3|4|DET 4|2|OBJ 5|4|CONJ 6|9|DET 7|9|MOD
8|9|MOD
         9|5|COORD 10|9|NJCT 11|12|DET 12|10|POBJ 13|2|PUNCT
*CHI:    I saw giraffe swimming in the pool to get that ball .
%mor:    pro:sub|I v|see&PAST n|giraffe part|swim-PRESP prep|in
det:art|the
         n|pool inf|to v|get pro:dem|that n|ball .
%gra:    1|2|SUBJ 2|0|ROOT 3|2|OBJ 4|2|XJCT 5|4|JCT 6|7|DET 7|5|POBJ
8|9|INF
         9|4|XCOMP 10|11|DET 11|9|OBJ 12|2|PUNCT
*CHI:    the giraffe got to get out of that pool .
%mor:    det:art|the n|giraffe v|get&PAST inf|to v|get adv|out prep|
of
         pro:dem|that n|pool .
%gra:    1|2|DET 2|3|SUBJ 3|0|ROOT 4|5|INF 5|3|XCOMP 6|5|JCT 7|6|JCT
8|9|DET
         9|7|POBJ 10|3|PUNCT
*CHI:    the [?] giraffe always get wet .
%mor:    det:art|the n|giraffe adv|always v|get part|wet&PASTP .
%gra:    1|2|DET 2|4|SUBJ 3|4|JCT 4|0|ROOT 5|4|PRED 6|4|PUNCT
*CHI:    that (i)s the end . [+ bch]
%mor:    pro:rel|that cop|be&3S det:art|the n|end .
%gra:    1|2|LINK 2|0|ROOT 3|4|DET 4|2|PRED 5|2|PUNCT
```

Figure 1: An excerpt from a SLI transcript in the ENNI dataset

2.2 Task 1: Handling with File Contents and Preprocessing

In the first task, you will begin by reading in all the transcripts of the given dataset, for both the SLI and TD groups. We will then conduct a number of pre-processing tasks to extract only the relevant contents or texts needed for analysis in the subsequent tasks (Task 2 and 3) in this assignment. Upon completing the pre-processing tasks, each of the cleaned transcript should be saved as an individual output file. This would be a more efficient approach whenever we need to manipulate the cleaned dataset without having to repeat the pre-processing task.

As mentioned earlier, for the purpose of this assignment, the data required for processing and analysis is the narrative produced by the children, which are those statements (or lines) indicated by the label of '*CHI:' in the transcripts. The first step is that, for each original transcript, extract only the statements which are prefixed or begin with '*CHI:'. (Note that there are some statements that extend to the next line, you should ensure that you take those into account.)

The next step is to perform a set of pre-processing or filtering tasks. We want to remove certain words (generally referred as *tokens*) in each statement that consist of some CHAT symbols as either prefixes or suffixes, but retaining certain symbols and words for analysis in Task 2. For this part of the implementation, you should consider *splitting* each statement into a list of words or tokens before you begin with the filtering process.

Below is a list of symbols that you should filter off from each of the child statements extracted.

(a) Remove those words that have either '**[**' as prefix or '**]**' as suffix[1] but retain these three symbols: **[//]**, **[/]**, and **[*]**[2]

Example:
Before filtering: ***CHI: then [/-] the doctor give the [/] money to the man .**
After filtering: **then the doctor give the [/] money to the man .**

(b) Retain those words that have either '<' as prefix or '>' as suffix but these two symbols should be removed

Example:
Before filtering: ***CHI: <he> [/] <he> [/] he hold it .**
After filtering: **he [/] he [/] he hold it .**

[1] Please note that any words between the pair of **[ ]** should also be removed.
[2] Also take note that **[*]** is replaced by **[* m:+ed]** given that there are zero occurrences of **[*]** in the given dataset.

(c) Remove those words that have prefixes of '&' and '+'

Example:
Before filtering: ***CHI: &=sighs <and instead the> [//] and then the giraffe got it and gave it to the elephant .**
After filtering: **and instead the [//] and then the giraffe got it and gave it to the elephant .**

(d) Retain those words that have either '**(**' as prefix or '**)**' as suffix but these two symbols should be removed

Example:
Before filtering: ***CHI: and then (.) the little (.) giraffe is crying because it (i)s sinking .**
After filtering: **and then (.) the little (.) giraffe is crying because it is sinking .**

Note: The '**(**' symbol could appear as an *infix*, i.e. appears in between a word — you should also remove it from the word or token that it is attached to. However, you should not remove the symbol of '**(.)**' as this should be retained for data analysis.

Finally, once you have completed with the filtering process for all the unwanted symbols, you should now save each of the cleaned child transcripts as an individual output file. You should produce a separate output file for each cleaned transcript and each statement should be written as a separate line in the output file. You may also want to organise your cleaned dataset into two groups: save the cleaned SLI transcripts under a folder named "**SLI_cleaned**", and the cleaned TD transcripts under another new folder named "**TD_cleaned**".

Note: You should name your program for this first task as "`task1_StudentID.py`".

2.3 Task 2: Building a Class for Data Analysis

The second task is about collating the required data for analysis. The main task is to produce a number of *statistics* for the two groups of children transcripts. These statistics are those that might serve as good indicators for distinguishing between the children with SLI and the typically developed (TD) children.

The statistics for each of child transcript that we are interested in are:

- Length of the transcript — indicated by the number of statements
- Size of the vocabulary — indicated by the number of unique words
- Number of repetition for certain words or phrases — indicated by the CHAT symbol **[/]**
- Number of retracing for certain words or phrases — indicated by the CHAT symbol **[//]**
- Number of grammatical errors detected — indicated by the CHAT symbol **[*]**[3]
- Number of pauses made — indicated by the CHAT symbol **(.)**

Note: Since the length of each child transcript is measured by the number of statements, the end of each statement can be determined based on the following punctuation marks: either a full stop '`.`', a question mark '`?`', or an exclamation mark '`!`'.

The implementation of this *analyser* class should include at least the following methods (you may implement additional methods if needed):

- `__init__(self)`:
  This is the constructor required for creating instances of this class. You should define a data structure based on your choice of data type for storing each of the required statistics (data fields) as the *instance variables* in the constructor.
- `__str__(self)`:
  Re-define this method to present your data (the instance variables) in a readable format. You should return a formatted string in this method.
- `analyse_script(self, cleaned_file)`:
  This is the method that performs the analysis on a given cleaned script. This method should accept the cleaned script as the argument, and attempt to extract the required data for analysis — all the six statistics of interest (as mentioned above). The extracted data should be stored in the data structures (i.e. instance variables) defined in the constructor.

[3]As revised in Task 1, the symbol for indicating grammatical errors **[*]** is replaced by **[* m:+ed]**.

Hint: You would construct at least one object of this analyser class for each child group, i.e. one object for the SLI group and one object for the TD group.

Note: You should name your program for this second task as "`task2_StudentID.py`".

2.4 Task 3: Building a Class for Data Visualisation

In the last task, you will implement a class to visualise the statistics collected in Section 2.3 (Task 2) as some form of graphs. The implementation of this *visualiser* class should make use of the external Python packages, such as *NumPy*, *SciPy*, *Pandas*, and/or *Matplotlib* in order to create the suitable graphs for comparing the statistics collected for the two groups of children transcripts. (We are suggesting bar charts here; you may choose any other types of graph that deemed appropriate.)

The implementation of this visualiser class should include the following methods (you may create additional methods if needed):

- `__init__(self, data)`:
  This is the constructor required for creating instances of this class. This method should take as an argument a list that contains all the statistics to be visualised and store them in an appropriate structure. Your choice of data type or data structure should at least allow you to represent the statistics in a tabular format, where the columns denote the statistic types and the rows denote the statistic counts of each child transcript.

- `compute_averages(self)`:
  This method returns the average or mean of the six statistics for each child group (i.e. both the SLI and TD groups).

- `visualise_statistics(self)`:
  This method should construct the suitable graph(s) to demonstrate the mean difference between the two groups (SLI vs.TD) for each of the six statistics, for comparison purposes. (You may construct more than one graph if you think that is the best way to present the statistics.)

Hint: You would construct at least one object of this visualiser class for each child group, i.e. one object for the SLI group and one object for the TD group.

Note: You should name your program for this final task as "`task3_StudentID.py`".

3 Important Notes

3.1 Documentation

Commenting your code is essential as part of the assessment criteria (refer to Section 3.2). You should also include comments at the beginning of your program file, which specify your name, your Student ID, the start date and the last modified date of the program, as well as with a high-level description of the program. In-line comments within the program are also part of the required documentation.

3.2 Marking Criteria

The assessment of this assignment will be based on the following marking criteria. The same marking criteria will be applied on all the three tasks:

- 60% for working program — functionality (for all three tasks);
- 10% for code architecture — algorithms, data types, control structures, use of internal libraries and/or external packages;
- 10% for coding style — clear logic, clarity in variable/function/class names, code readability;
- 20% for documentation — program comments and user documentation.

Note: Interviews will be conducted during Week 12 tutorial sessions as part of the assessment requirement.

4 Submission

There will be NO hard copy submission required for this assignment. You are required to submit your assignment as a `.zip` file named with your Student ID. For example, if your Student ID is `12345678`, you would submit a zipped file named "`A2_12345678.zip`". Note that marks will be deducted if this requirement is not strictly complied with.

Your submission must be done via the assignment submission link on the FIT9133 S2 2018 Moodle site by the deadline specified in Section 1, i.e. **12 October 2018 (Friday) by 17:00pm**.

4.1 Deliverables

Your submission should contain the following:

- The three Python scripts named as "`task1_StudentID.py`", "`task2_StudentID.py`" and "`task3_StudentID.py`". If you have additional scripts for your implementation, name them with the similar convention.
- An user documentation (at least 4 pages but not more than 8 pages) in the PDF format with clear and complete instructions on how to run your programs and the analytical outputs (tables and graphs). This document should be a well-formatted document with headings and sub-headings, and a table of contents should be included.
- Electronic copies of ALL your files that are needed to run your programs.

Marks will deducted for any of these requirements that are not strictly complied with.

Your programs must at least run on the computers in the University's computer labs. Any submission that does not run accordingly will receive no marks.

4.2 Academic Integrity: Plagiarism and Collusion

Plagiarism means to take and use another person's ideas and or manner of expressing them and to pass them off as your own by failing to give appropriate acknowledgement. This includes materials sourced from the Internet, staff, other students, and from published and unpublished works.

Collusion means unauthorised collaboration on assessable work (written, oral, or practical) with other people. This occurs when you present group work as your own or as the work of another person. Collusion may be with another Monash student or with people or students external to the University. This applies to work assessed by Monash or another university.

It is your responsibility to make yourself familiar with the University's policies and procedures in the event of suspected breaches of academic integrity. (Note: Students will be asked to attend an interview should such a situation is detected.)

The University's policies are available at: `http://www.monash.edu/students/academic/policies/academic-integrity`