FIT5196-S2-2023 Assessment 1(35%)

This is an **individual assessment** and worth **35%** of your total mark for FIT5196.

Due date & time: Friday 25 Aug, 2023, 16:30PM

Background

Data, especially well-structured data, is the foundation of the current success of the Machine Learning and AI industry. The recent trending tool: ChatGPT, as a generative pre-trained language model, is also built upon training with numerous good-quality data.
However, in real-life scenarios, data is often in an unstructured format, commonly referred to as 'raw data'. Examples of raw data include text files obtained from system sales reports, PDF files downloaded from government databases, and images extracted from journal papers. The level of 'intelligence' attained by future or target models depends on the variety, accuracy, and level of structure present in the data they are fed.

Objectives

Parsing text files and performing text pre-processing are critical steps in preparing text data for analysis. These steps help ensure that the resulting insights are accurate, reliable, and meaningful. In this assessment, you are required to apply the knowledge and skills learned from Week 2 to Week 5 to parse text data from different raw data files, extract specific information, and perform slight pre-processing. The ultimate goal is to output the data in a structured, machine-readable format suitable for further data analysis and modelling tasks. This assessment is an individual task and accounts for 35% of your total mark for FIT5196.

Task 1: Parsing Text Files (15/35)

Parsing data touches on the very first step of data analysis, i.e., extracting data from different formats of data sources. You are provided with 10 files that contain information about reviews from a retail company users on its product. You can only use the data file with your STUDENT_ID, i.e. ***<student_id>.txt*** from the shared Google Drive folder (student_data). Using the wrong data will result in a ZERO mark as every student has a unique dataset to produce unique results. Three output files need to be named following the rules in the table below.

Input Files	Output Files (submission)
<student id>_task1_input#.txt	*<student id>.xml* *task1_<student id>.ipynb* *task1_<student id>.py*

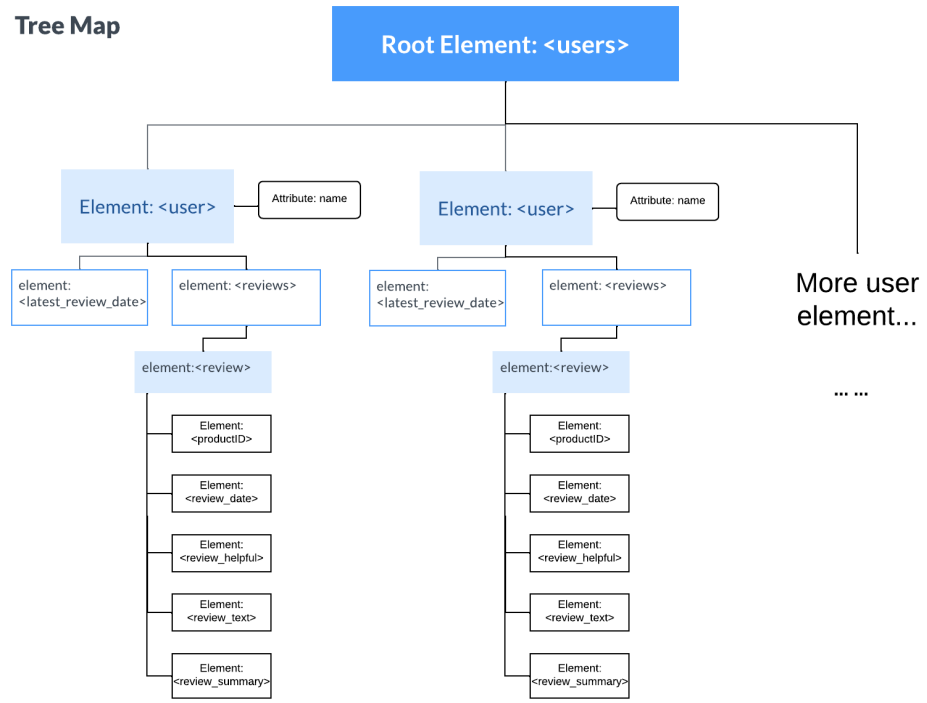
Your input data contains information about reviews, i.e., **"reviewerID", "productID", "reviewer.NAME", "No. helps", "review_date", "REVIEW",** and the **"SUMMARY"**. Your task is to use **regular expressions** to extract all information regarding reviews from the text

file, transform and represent the extracted data into a **XML** format with the following elements:

1. **users**: this tag wraps all the users, i.e. multiple <user> tag under <users>
2. **user**: this tag wraps all the reviews from a particular user and keeps the meta data for each user such as the latest review date and its username.
3. **reviews**: wraps all the reviews of a specific user
4. **review**: for each user, this tag wraps the **"productID"**, **"review_date"**, "**review_helpful**", and **"review_text"**, "**review_summary**" of the user tweet

Note: All the tag names are **case-sensitive** in the output XML file. You can refer to the sample **here** for the correct XML file structure.

Tree Map for Target XML:



Example: Sample Tree Map for XML output:

Task 1 Guidelines

To complete the above task, please follow the steps below:

Step 0: Study the sample files

- Open and check your input txt file and find patterns for different data elements
- Use other online web applications such as xmlviewer to better understand the structure of the XML sample output.

Step 1: Txt file parsing

- Use python library to parse txt file
- Use Regex to extract the required attributes and their values as listed above

Step 2: Further process the extracted text from Step 1

- Remove the XML special characters from raw text (or replace with ' ', a white space)
- Save the data into a proper data format e.g. dataframe, dictionary...

Step 3: XML file output

- Use python library to transfer your data in step 2 into proper xml format (make sure you check the spelling, upper/lower case, key names and name hierarchy of your XML data)

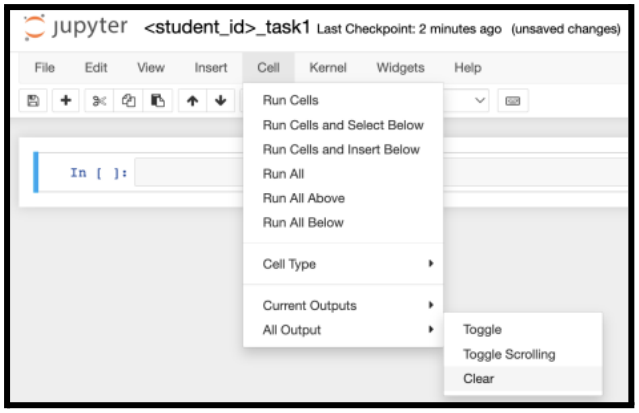
Submission Requirements

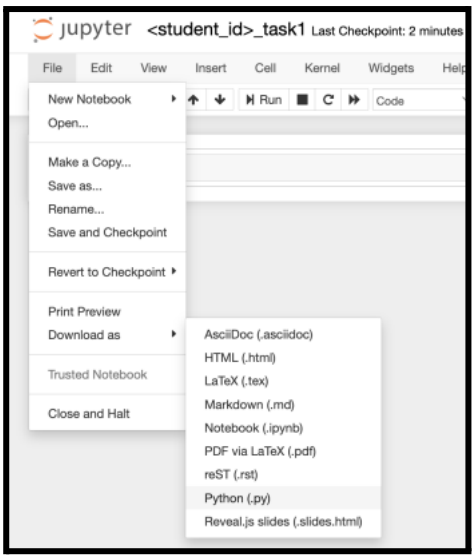
You need to submit 3 files:

- A **task1_<student_id>.xml** file contains the correct review information with all the elements listed above.
- A Python notebook named **task1_<student_id>.ipynb** contains a well-documented report that demonstrates your solution to Task 1. You need to clearly present the methodology, that is, the entire step-by-step process of your solution with appropriate comments and explanations. You can follow the suggested steps in the guideline above. Please keep this notebook easy-to-read, as you will lose marks if we cannot understand it. (make sure the cell outputs are NOT cleared)
- A **task1_<student_id>.py** file. This file will be used for plagiarism check. (make sure the cell outputs are cleared before exporting)

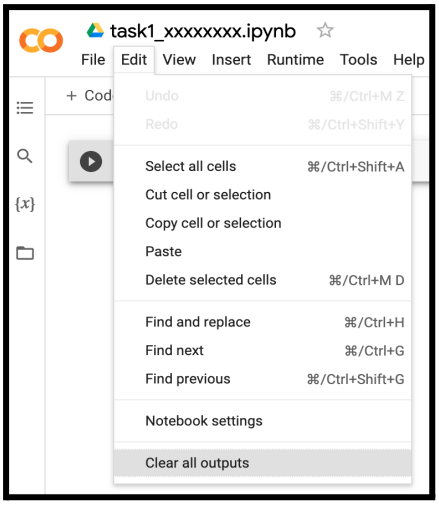
To generate a .py file, you need to clear all the cell outputs, and then download it.

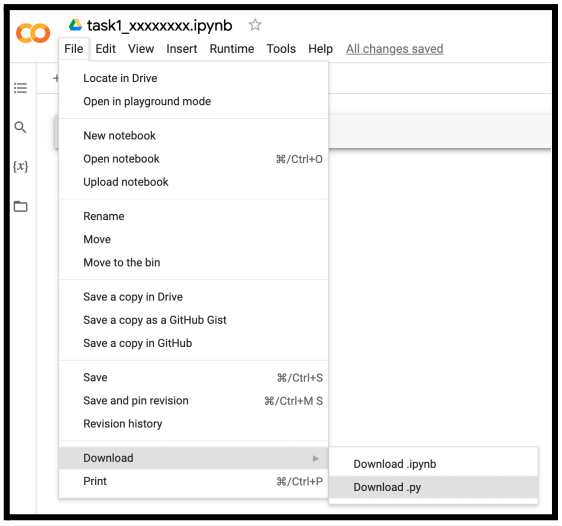
In Jupyter notebook:





In Google colab:





Requirements on the Python notebook (report)

- Methodology - 30%
  - You need to demonstrate your solution using correct regular expressions. Results from each step could help to demonstrate your solution better and be easier to understand.
  - You should present your solution in a proper way including all required steps. Skip any steps will cause
  - You need to select and use the appropriate Python functions for input, process and output.
  - Your solution should be an efficient one without redundant operations and unnecessary reading and writing the data.
- Report organisation and writing - 10%
  - The report should be organised in a proper structure to present your solutions to Task 1 with clear and meaningful titles for sections and subsections or sub-subsection if needed.

- Each step in your solution should be clearly described. For example, you can write to explain your idea of the solution, any specific settings, and the reason for using a particular function, etc.
- Explanation of your results including all intermediate steps is required. This can help the marking team to understand your solution and give partial marks if the final results are not fully correct.
- All your codes need proper (but not excessive) commenting.
- You can refer to the notebook templates provided as a guideline for a properly formatted notebook report.

Task 2: Text Pre-Processing (17/35)

This task touches on the next step of analysing textual data, converting the extracted text data into a numerical representation thus it can be used for a downstream modelling task. In this task, you are required to write Python code to pre-process a set of published papers (pdf) and convert them into numerical representations. The numerical representation is the standard format of text data when (which are suitable for input into NLP systems such as: recommender-systems, information-retrieval algorithms, machine-translation etc.). The most basic step for natural language processing (NLP) tasks is to convert words into numbers for machines to understand & decode patterns within a language. This step, though iterative, plays a significant role in deciding features for your machine learning model/algorithm.

Input Files	**Output Files (submission)**
<student id>_paper_list.pdf	*<student id>_paper_list.csv* *<student id>_vocab.txt* *<student id>_countvec.txt* *task2_<student id>.ipynb* *task2_<student id>.py*

You are provided with a unique dataset containing 80 URLs for papers published in several popular AI conferences. Please use the data file with your STUDENT_ID, i.e. ***<student_id>.txt*** in the Google drive folder (student data). The pdf file (***<student_id>.pdf***) contains a table in which each row contains a paper with a unique ID and a URL where it can be downloaded.

You are asked to parse the table of paper URLs in python, and output the table into a csv file. Then programmatically download all papers, and parse the required abstract section from all papers. Then pre-process the abstract text and generate a vocabulary list and numerical representation for the corresponding text, which will be used in the model training by your colleagues. The information regarding output files is listed below:

- **paper_list.csv** contains the unique paper IDs along with their corresponding URLs.
- **vocab.txt** comprises unique stemmed tokens sorted alphabetically, presented in the format of **token_index:token**, as outlined in Guideline step 3.

- **countvec.txt** includes numerical representations of all tokens, organised by paper ID and token index, following the format **paper_id, token_index:frequency**, as outlined in Guideline step 4.

Carefully examine the sample files (**here**) for detailed information about the output structure.

VERY IMPORTANT NOTE: The sample outputs are just for you to understand the structure of the required output and the correctness of their content in task 2 is not guaranteed. So please do not try to reverse engineer the outputs as it will fail to generate the correct content.

Task 2 Guideline

To complete the above task, please follow the steps below:

Step 1: Programmatically download the pdfs

- Use the given URLs to programmatically download the PDF files (manual download will be penalised, so please only use it as a last resort).

Step 2: Read the pdf files into text

- Read the PDF files into text and extract the required entities to complete the above task (hint: **pdfminer, MuPDF, pdfplumber, etc** and **re** packages can help you complete this task).
- Replace the ligatures, deal with the characters of special html entities (either using replace function, or solve it through unicode methods), fix the hyphen-separated word (e.g., change 'r-emix' to 'remix' where hyphen generated due to pdf format),

Step 3: Generate the token list and output as vocab.txt.

Before building the sparse representation, you will need to perform text preprocessing on **Abstract**. Please follow the following text preprocessing steps. Do **NOT** change the order as the correct order of operations ensures producing the correct final set of vocabulary.

1. Tokenize using the following regular expression

   r"[A-Za-z]\w+(?:[-'?]\w+)?"

2. Remove context-independent stop words (i.e., **stopwords_en.txt**)
3. Remove context-dependent stop words (unigram and bigram tokens appearing in 95% or more of the files).
4. Remove rare tokens (appearing in less than 3% of the files)
5. Remove tokens with less than 3 characters/symbols.
6. Stem unigram tokens using the Porter stemmer.
7. Generate the vocab.txt output with ascending ordered unigrams, with format:

   token1: token1_index
   token1: token1_index
   …

Step 4: Generate the sparse numerical representation and output as countvec.txt

1. Generate sparse representation by using the countvectorizer() function OR directly count the frequency using FreqDist().
2. Mapping the generated token with the stemmed token in step 3 if need
3. Output the sparse numerical representation into txt file with the format:
   paper_id1, token1_index:token1_frequency, token2_index:token2_frequency, token3_index:token3_frequency, …
   paper_id2, token2_index:token2_frequency, token5_index:token5_frequency, token7_index:token7_frequency, …
   paper_id3, token6_index:token6_frequency, token9_index:token9_frequency, token12_index:token12_frequency, …

Submission Requirements

You need to submit 6 files:

1. A **<student id>_paper_list.csv** file with exactly the same structure from pdf input
2. A **<student_id>_vocab.txt** that contains the unigrams tokens in the following format, token:token_index. Words in the vocabulary must be sorted in alphabetical order.
3. A **<student_id>_countvec.txt** file, in which each line contains the sparse representations of one of the papers in the following format:

   paper_id, token1_index:token1_wordcount, token2_index:token2_wordcount, …

   Please note: the tokens with zero word count should NOT be included in the sparse representation.
4. A **task2_<student_id>.ipynb** file that contains your report explaining the code and the methodology. (make sure the cell outputs are NOT cleared)
5. A **task2_<student_id>.py** file for plagiarism checks. (make sure the cell outputs are cleared before exporting)

Requirements on the Python notebook (report)

- Methodology - 30%
  - You need to demonstrate your solution using correct regular expressions.
  - You should present your solution in a proper way including all required steps.
  - You need to select and use the appropriate Python functions for input, process and output.
  - Your solution should be an efficient one without redundant operations and unnecessary reading and writing the data.
- Report organisation and writing - 10%
  - The report should be organised in a proper structure to present your solutions to Task 2 with clear and meaningful titles for sections and subsections or sub-subsection if needed.
  - Each step in your solution should be clearly described. For example, you can write to explain your idea of the solution, any specific settings, and the reason for using a particular function, etc.

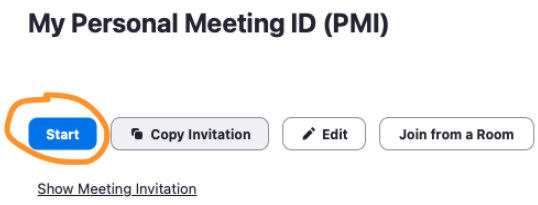
- Explanation of your results including all intermediate steps is required. This can help the marking team to understand your solution and give partial marks if the final results are not fully correct.
- All your codes need proper (but not excessive) commenting.
- You can refer to the notebook templates provided as a guideline for a properly formatted notebook report.

Task 3: Video Presentation (3/35)

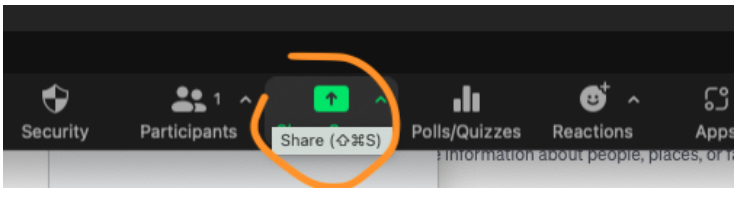
Presenting your methodology to the audience and explaining your logic is a crucial skill for a data analyst. In this task, you are required to **record a video** (less than 5 minutes) to go through the main logic of your codes in **BOTH Task 1 and 2.** You are required to show your notebook file with output while explaining in voice how your code works to generate the expected output. The explanation of methodology needs to be clear and allows the audience to understand how your python code parses the data, extract the data and pre-process the data. You can follow the steps below to record your video.

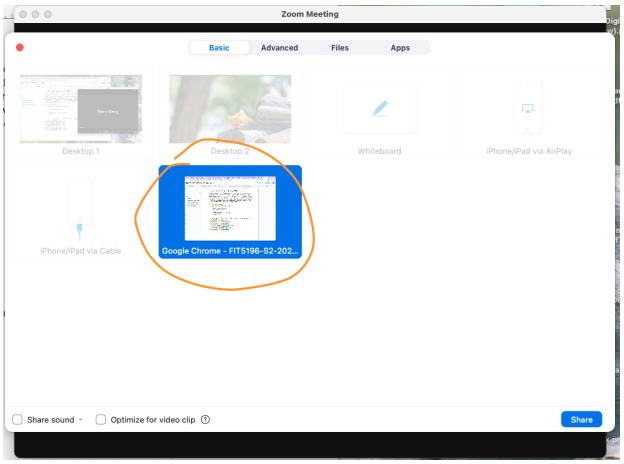
Zoom Recording Guideline:

1. Open the Zoom and enter your meeting room.

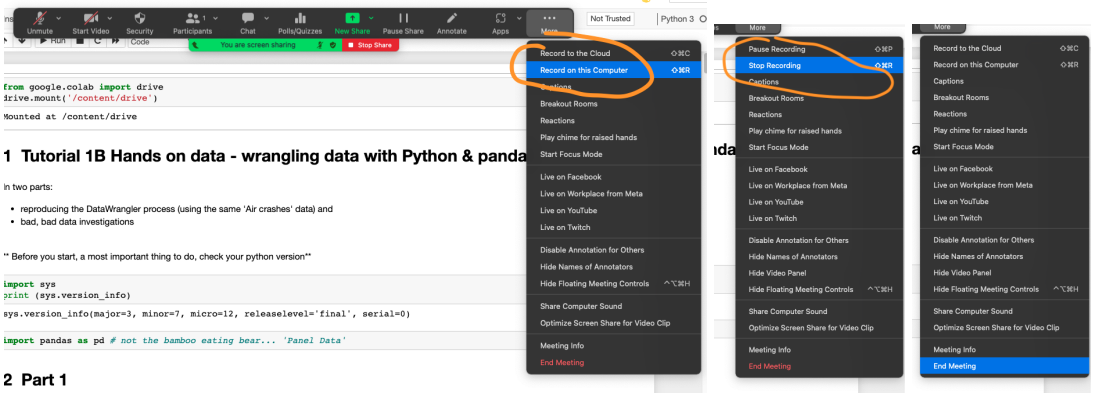


2. **Share your screen** with the notebook opened, make sure your camera is turned on and shown on the screen.

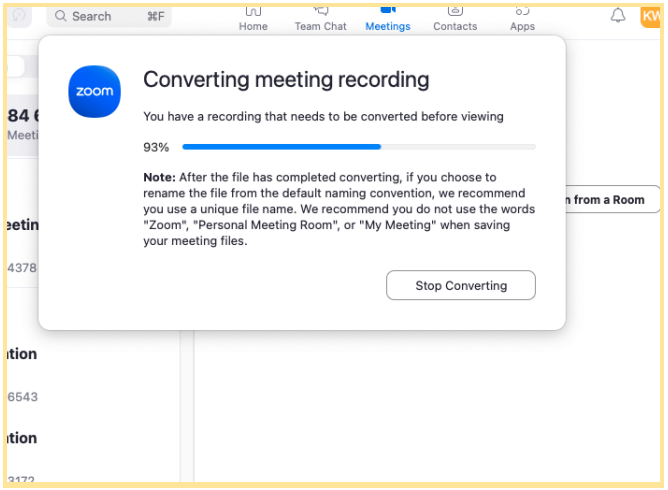


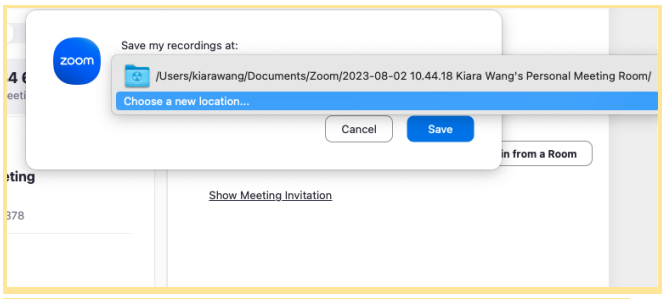


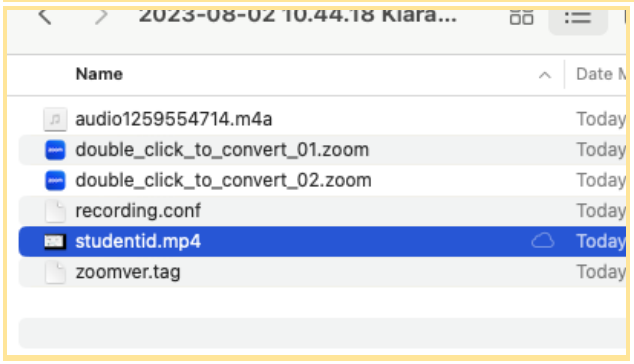
3. **Start recording** and finish everything in 5 minutes, then press **stop recording** and **end meeting**. You can pause and resume multiple times if necessary.



4. Wait for the video conversion and find the file, **rename** it with your **student id**.







5. **Submit it on Moodle** together with your jupyter notebook.

Submission Checklist:

- ☐ Please zip all the submission files for task 1 and 2 into a single file with the name **<student_id>_ass1.zip.** (any other format e.g. rar or 7z will be penalised)
- ☐ There are **8 files** in your compressed zip file
- ☐ Please submit the video with the name ***<student_id>.mp4***
- ☐ **<student_id>** should be replaced with **your monash student ID**.
- ☐ Please strictly follow the file naming standard. Any misnamed file will carry a penalty.
- ☐ Please make sure that your **.ipynb file** contains printed output, while your **.py file** does not include any output.
- ☐ Please ensure that all your files are parsable and readable. You can achieve this by re-reading all your generated files back into python. (e.g. using `read_csv` for CSV files or `ElementTree` module for xml). These checks are only sanity checks and hence should not be added to your final submission.

Note: All submissions will be put through a plagiarism detection software which automatically checks for their similarity with respect to other submissions. Any plagiarism found will trigger the Faculty's relevant procedures and may result in severe penalties, up to and including exclusion from the university.