

FIT5196-S2-2024 Assessment 1 (35%)

This is a group assessment and worth 35% of your total mark for FIT5196.

Due date: 23:55 PM, Friday, August 30, 2024

Text documents, such as those derived from web crawling, typically consist of topically coherent content. Within each segment of topically coherent data, word usage exhibits more consistent lexical distributions compared to the entire dataset. For text analysis tasks, such as passage retrieval in information retrieval (IR), document summarization, recommender systems, and learning-to-rank methods, a linear partitioning of texts into topic segments is effective. In this assessment, your group is required to successfully complete all five tasks listed below to achieve full marks.

Task 1: Parsing Raw Files(8/35)

This task touches on the very first step of analysing textual data, i.e., extracting data from semi-structured text files.

Allowed libraries: re, json, pandas, datetime, os

Input Files	Output Files (submission)
group<group_number>.txt group<group_number>.xlsx	task1_<group_number>.json task1_<group_number>.csv task1_<group_number>.ipynb task1_<group_number>.py (the <group_number> has 0 paddings ie.001,010...)

Your group is provided with review information on Google map from California (ratings, text, images, etc.). Please use the input data files with your *group_number*, i.e.

group<group_number>_x.txt and **group<group_number>.xlsx** in the Google drive folder ([student_data](#)).

Note: Using a wrong input dataset will result in ZERO marks for 'Output' in marking rubric.

Your dataset contains a subset of records of the Google map reviews. The google map reviews are recorded with a set of attributes:

- **user_id** - ID of the reviewer
- **name** - name of the reviewer
- **time** - time of the review (unix time)
- **rating** - rating of the business

- **text** - text of the review
- **pics** - pictures of the review
- **resp** - business response to the review including unix time and text of the response
- **gmap_id** - ID of the business

Please check with the sample input files ([sample input](#)) for all the available attributes.

Your task is to extract the data from all of your input files (including an excel file and 15 .txt files following a **mis-structured** xml format). You are asked to extract and transform the data into a csv file and a JSON file.

For the **csv file**, you are required to produce an output file with following columns:

- **gmap_id**: the ID of the business.
- **review_count**: the number of total reviews for a business.
- **review_text_count**: the number of reviews that contains a text.
- **response_count**: the number of responses from a business.

For the **JSON file**, you are required to produce an output file with the following fields:

- **gmap_id**: the ID of the business
- **reviews**: a root element with one or more reviews, contains fields:
 - **user_id**: ID of the reviewer.
 - **time**: the time of the review. (output format: UTC time in YYYY-MM-DD tt:hh:ss)
 - **review_rating**: rating of the business.
 - **review_text**: the english review text, if a review is in other languages, only extract the english translation version. Need to remove emojis before output based on the list and the output text should be normalised to lowercase. Place "None" as the value, if there is no review.
 - **If_pic**: if the reviewer include pictures. (output format: Y/N)
 - **pic_dim**: the dimension of pictures in a list of tuples.eg [[h,w],[h,w]...]. Place [] as the value, if there is no picture.
 - **If_response**: if the review has a response (output format: Y/N)
- **earliest_review_date**: the earliest review date for a given business in the given data subset. (output format: UTC time in YYYY-MM-DD tt:hh:ss)
- **latest_review_date**: the latest review date for a given business in the given data subset. (output format: UTC time in YYYY-MM-DD tt:hh:ss)

VERY IMPORTANT NOTE:

1. All the tag names are **case-sensitive** in the output json file. You can refer to the [sample output](#) for the correct json file structure.
2. The sample output files are just for you to understand the structure of the required output and the correctness of their content in task 1 is not guaranteed. So please do not try to reverse engineer the outputs as it will fail to generate the correct content.

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 try to find any 'potential interesting' patterns for different data elements

Step 1: Txt file parsing and excel file parsing

- Load the input file
- Use Regex to extract the required attributes and their values as listed from the txt file
- Extract necessary data from the excel file
- Combine all data together

Step 2: Further process the extracted text from Step 1

- Remove any duplicates if needed
- Further process the extracted data
- Note for review texts: they should be transformed into lower case and with no emojis
 - To remove emojis, make sure your text data is in utf-8 format
 - The list of emojis to remove are:

```
"["  
    "\U0001F600-\U0001F64F"  
    "\U0001F300-\U0001F5FF"  
    "\U0001F680-\U0001F6FF"  
    "\U0001F1E0-\U0001F1FF"  
    "\U00002702-\U000027B0"  
    "\U000024C2-\U0001F251"  
"]+"
```

Step 3: file output

- Output the required files based on the specified structures provided above, make sure your data is utf-8 encoded.

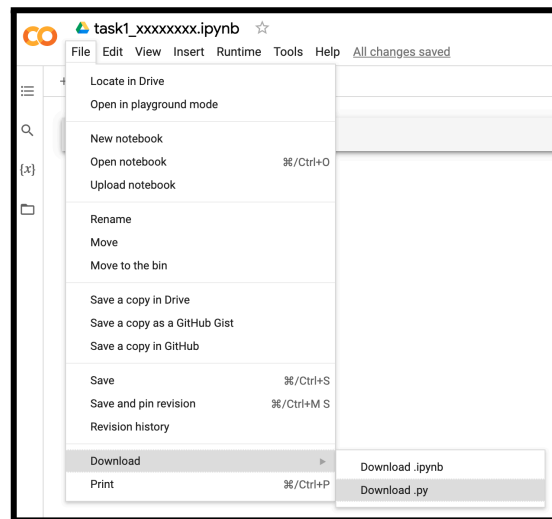
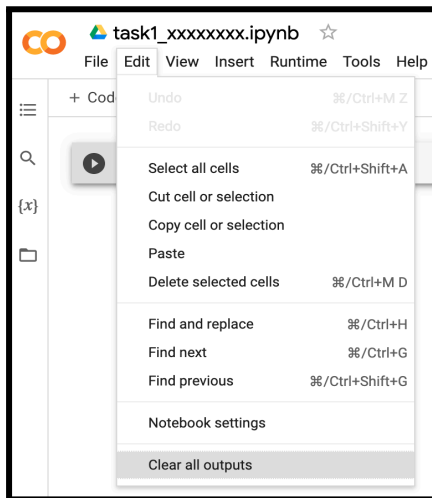
Submission Requirements

You need to submit 4 files:

- A **task1_<group_number>.json** file contains the correct review information with all the elements listed above.
- A **task1_<group_number>.csv** file contains the correct review information with all the elements listed above.
- A Python notebook named **task1_<group_number>.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 you PRINT OUT your cell output).

- A **task1_<group_number>.py** file. This file will be used for plagiarism check (make sure you clear your cell output before exporting).

In Google colab:



Requirements on the Python notebook (report)

- Methodology - 35%
 - 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 a penalty on grade.
 - 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 - 15%
 - 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 (12/35)

This task involves the next step in textual data analysis: converting extracted text into a numerical representation for downstream modelling tasks. You are required to write Python code to preprocess Google Reviews text from Task 1 and transform it into numerical representations. These numerical representations are the standard format for text data, suitable for input into NLP systems such as recommender systems, information retrieval algorithms, and machine translation. The most fundamental step in natural language processing (NLP) tasks is converting words into numbers to enable machines to understand and decode patterns within a language. This step, although iterative, is crucial in determining the features for your machine learning models and algorithms.

Allowed libraries: ALL

Input Files	Output Files (submission)
<i>task1_<group_number>.json</i> OR <i>task1_<group_number>.csv</i>	<i><group_number>_vocab.txt</i> <i><group_number>_countvec.txt</i> <i>task2_<group_number>.ipynb</i> <i>task2_<group_number>.py</i>

In this task you are required to continue working with the data from task1.

You are asked to use the **review** fields in all the reviews from all the businesses that have at least 70 text reviews. 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:

- **<group_number>_vocab.txt** comprises unique stemmed tokens sorted alphabetically, presented in the format of **token:token_index**
- **<group_number>_countvec.txt** includes numerical representations of all tokens, organised by channels_id and token index, following the format **channel_id, token_index:frequency**.

Carefully examine the sample output 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: Text extraction

- You are required to extract the review text from the output of task 1.

- Validate your review text data: the text data should be all in English and in lower case with no emojis.
- You are only required to extract the vocab and countvec lists for review from businesses that **have at least 70 text reviews**

Step 2: Generate the unigram and bigram lists and output as vocab.txt

- The following steps must be performed (**not necessarily in the same order**) to complete the assessment. **Please note that the order of preprocessing matters and will result in different vocabulary and hence different count vectors. It is part of the assessment to figure out the correct order of preprocessing which makes the most sense as we learned in the unit.** You are encouraged to ask questions and discuss them with the teaching team if in doubt.
 - a. The word tokenization must use the following regular expression, "[a-zA-Z]+"
 - b. The context-independent and context-dependent stopwords must be removed from the vocabulary.
 - For context-independent, The provided context-independent stop words list (i.e, [stopwords_en.txt](#)) must be used.
 - For context-dependent stopwords, you must set the threshold to words that appear in more than **95% of the businesses that have at least 70 text reviews.**
 - c. Tokens should be stemmed using the **Porter** stemmer.
 - d. Rare tokens must be removed from the vocab (with the threshold set to be words that appear in less than **5% of the businesses that have at least 70 text reviews.**
 - e. Tokens with a length less than 3 should be removed from the vocab.
 - f. **First 200 meaningful bigrams** (i.e., collocations) must be included in the vocab using **PMI** measure, then makes sure the collocations can be collocated within the same review.
 - g. Calculate the vocabulary containing both unigrams and bigrams.
- Combine the unigrams and bigrams, sort the list alphabetically in an ascending order and output as vocab.txt

Step 3: 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. Output the sparse numerical representation into txt file with the following format:


```
gmap_id1, token1_index:token1_frequency, token2_index:token2_frequency,
token3_index:token3_frequency, ...
gmap_id2, token2_index:token2_frequency, token5_index:token5_frequency,
token7_index:token7_frequency, ...
gmap_id3, token6_index:token6_frequency, token9_index:token9_frequency,
token12_index:token12_frequency, ...
```

Note: the token_index comes from the vocab.txt and make sure you are counting bigrams

Submission Requirements

You need to submit 4 files:

1. A `<group_number>_vocab.txt` that contains the unigrams and bigrams tokens in the following format, `token:token_index`. Words in the vocabulary must be sorted in alphabetical order.
2. A `<group_number>_countvec.txt` file, in which each line contains the sparse representations of one of the channel in the following format:

```
gmap_id1, token1_index:token1_wordcount, token2_index:token2_wordcount, ...
```

Please note: the tokens with zero word count should **NOT** be included in the sparse representation.

3. A `task2_<group_number>.ipynb` file that contains your report explaining the code and the methodology. (make sure you PRINT OUT your cell outputs)
4. A `task2_<group_number>.py` file for plagiarism checks. (make sure you clear your cell outputs)

Requirements on the Python notebook (report)

- Methodology - 35%
 - 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 - 15%
 - 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: Data Exploratory Analysis (12/35)

In this task, you are asked to conduct a comprehensive exploratory data analysis (EDA) on the provided Google Review data. The goal is to uncover interesting insights that can be useful for further analysis or decision-making.

Output Files (submission)
<code>task3_<group_number>.ipynb</code> <code>task3_<group_number>.py</code>

Task 3 Guideline

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

Step 1: Understand the Sample Data:

- Review the data provided from the previous exercise.
- Summarise the key features and variables included in the dataset.
- Identify any initial patterns, trends

Step 2: Understand the Auxiliary Dataset: Metadata(optional):

- Understand the [auxiliary metadata dataset](#).
- Evaluate the usefulness of the metadata in conjunction with the main dataset.
- Decide whether to incorporate the metadata into your analysis.

Note: you need to perform this step for a HD grading.

Step 3: Data Analysis:

- You can choose to use either the main Google Review data, the auxiliary metadata, or both.
- Perform an exploratory data analysis to investigate and uncover interesting insights.
- Insights should be data-driven and substantiated by visualisations and/or statistical summaries
- You are required to Investigate 2-5 interesting insights from the data.
- For a HD grading, you need to have at least 4 **meaningful** insights with in-depth discussions (and at least two of them need to come from the combination of metadata and the review data).

Submission Requirements

You need to submit 2 files:

5. A `task3_<group_number>.ipynb` file that contains your report explaining the code and the methodology. (make sure you PRINT OUT your cell outputs)
6. A `task3_<group_number>.py` file for plagiarism check. (make sure you clear your cell outputs)

Task 4: Video presentation for Task 3 (2/35)

Create a video presentation **(5-8 minutes)** to effectively communicate the findings from your exploratory data analysis (EDA) on the Google Review data. The goal is to present your methodology and insights in a clear, concise, and engaging manner.

Output Files (submission)
<code>task4_<group_number>.mp4</code>

Submission Requirements

Here are the key components you need to include in your submission:

Introduction:

- Briefly introduce yourself and provide context for the analysis.
- Explain the purpose of the EDA and the datasets used (main Google Review data and auxiliary metadata, if applicable).

Methodology:

- Describe the steps taken during the data analysis process.

Insights:

- Present 2-5 interesting insights uncovered from the analysis.
- Use visual aids such as charts, graphs, or tables to support your insights.
- Explain the significance of each insight and how it can be applied or interpreted.

Conclusion:

- Summarise the key findings and their potential implications.
- Discuss any limitations of the analysis and suggest areas for further research.

Task 5: Development History (1/35)

For this task, your group is required to provide a comprehensive development history of your assignment, showcasing incremental progress over **at least three different time points**. The purpose of this task is to demonstrate your ability to manage and document the evolution of your project, including changes made, challenges faced, and collaborative efforts with your group mates.

Output Files (submission)
<i>task5_<group_number>.pdf</i>

Requirement: This task must be completed (HURDLE)

Consequences of missing development history:

Failure to submit your development history for a particular task, will result in not meeting the hurdle requirements for Assignment A2. Consequently, you will **receive ZERO for the task..**

Submission Requirements

Here are the key components you need to include in your submission ([example](#)):

- 1. Development Timeline:** Provide a detailed timeline highlighting at least three significant time points in the development of your assignment. Each time point should be accompanied by a description of the changes made and the rationale behind those changes.
- 2. Version Screenshots:** Include screenshots or snapshots of different versions of your assignment at each time point. This should clearly illustrate the incremental development and any modifications made to the project.
- 3. Collaborative Effort(If you are doing the assignment with another student):** Document the collaborative effort with your group mates. This can be a description of the contributions made by each team member or screenshots of proof showcasing the collaborative effort with your group mates.

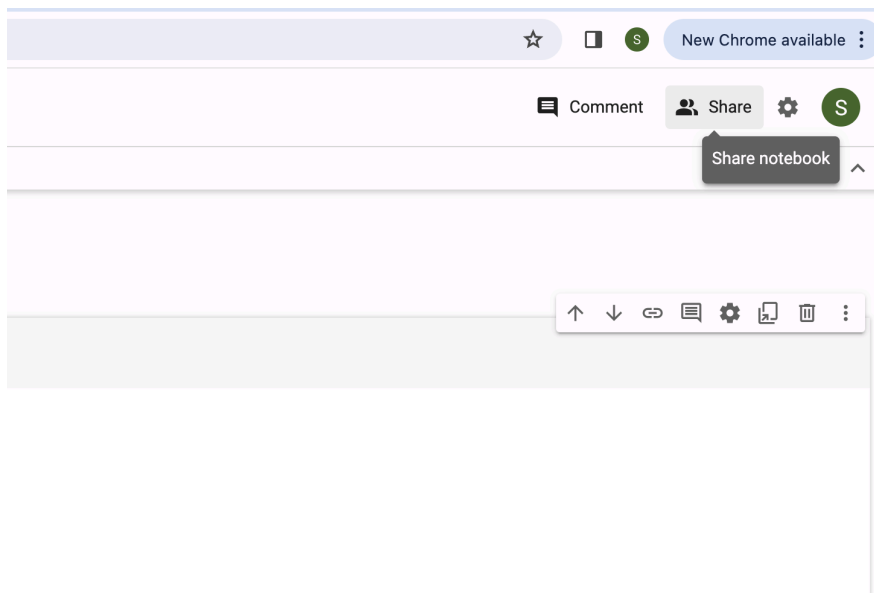
Note: We only require brief descriptions here, they don't have to be long as long as they reflect the key components we listed above. **If you use Google colab, it provides a comprehensive version history.**

Note: to avoid losing important work, you need to pin the version history that we wish to save. This keeps the notebook version saved permanently.

File	Edit	View	Insert	Runtime	Tools	Help
Locate in Drive						
Open in playground mode						
New notebook						
Open notebook					Ctrl+O	
Upload notebook						
Rename						
Move						
Move to trash						
Save a copy in Drive						
Save a copy as a GitHub Gist						
Save a copy in GitHub						
Save					Ctrl+S	
Save and pin revision					Ctrl+M S	
Revision history						

Instructions- Sharing ipynb link

1. Click on the Share button on the top right corner




2. Make sure under General access section, you have the permission sets to 'Monash University' and 'Editor', then click on 'Copy link'

Share 'Assignment1-sample.ipynb' ⓘ ⚙️


Add people, groups and calendar events


+ Kiara Wang

People with access

 Shawna Zhao (you)
shawna.zhao@monash.edu Owner

General access

 Monash University ▼
Anyone in this group with the link can edit Editor ▼

 Copy link Done

3. Create a markdown cell at the end of your assignment. Paste the sharelink/create a hyperlink object.

```
[ ]
```

T **B** *I* <> 🔗 🖼️ 📄 📌 📋 — ψ 😊 📄

[Link to my workspace] (<https://colab.research.google.com/drive/1bawiyLBvaPezfa5ANjn4RSyGziIhCBiR?usp=sharing>)

[Link to my workspace](#)

4. Double check the link to make sure it is working.

Submission Checklist:

- ☐ Please zip all the submission files for task 1 and 2 into a single file with the name **<group_number>_ass1.zip**. (any other format e.g. rar or 7z or other zip file names will be penalised)
- ☐ There are **12 files** in your compressed zip file
- ☐ **<group_number>** should be replaced with **your group id**. (the **<group_number>** has 0 paddings ie.001,010...)
- ☐ Make sure both members of your group click the 'Submit' button on Moodle
- ☐ 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 **json** module for JSON). 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.