FIT5196-S1-2024 assessment 1

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

Due date: 11:55 PM, Friday, April 19, 2024

Text documents, such as crawled web data, are usually composed of topically coherent text data, which within each topically coherent data, one would expect that the word usage demonstrates more consistent lexical distributions than that across the dataset. A linear partition of texts into topic segments can be used for text analysis tasks, such as passage retrieval in IR (information retrieval), document summarization, recommender systems, and learning-to-rank methods.

Task 1: Parsing Raw Text Files(17/35)

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

Allowed libraries: re,json,pandas

Input Files	Output Files (submission)
Group<group_number>.txt	task1_<group_number>.json task1_<group_number>.ipynb task1_<group_number>.py

Your group is provided with a unique dataset containing youtube comments. Please use the data file with your *group_number*, i.e. **Group<group_number>.txt** 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 on trademark assignments (**please find your group file on the Google drive**, i.e., Group<group_number>.txt). The trademark assignments are recorded with a set of attributes, e.g., reel no, frame no, assignors, assignees etc. Please check with the sample input file(**sample_input_task1.txt**) for all the available attributes.

Your task is to extract the data and transform it into the JSON format with the following elements:

- **rf-id:** a unique numerical ID for a trademark assignment entry. It is a combination of reel-no and frame-no
- **last-update-date:** An date identifies when the assignment record was last modified (output format YYYY-MM-DD in digits)

- **conveyance-text:** Contains textual description of the interest conveyed or transaction recorded.
- **correspondent-party:** The name of a person or organisation to whom correspondence related to the assignment. Don't need to reformat the name for this entity.
- **assignors-info:** a root element with one or more assignors, contains fields:
    - **party-name:** Identifies the party (person or organisation name) conveying an interest or transaction. If it is a person and has a title, remove the title.
    - **date-acknowledged:** Date on which the supporting legal documentation was acknowledged. (output format YYYY-MM-DD in digits).
    - **execution-date:** Date on which the supporting legal documentation was executed. (output format YYYY-MM-DD in digits).
    - **country:** The party's country location. Put "USA" as the value if the country name is a variant of USA and fill the value as "UK" if the country is a variant of the UK. When there is no explicit country-name, you can infer the country-name by state, when there is no explicit state, you can assume the value for nationality is the party's country location. In the case of when a country is indecisive or you get a value of NOT PROVIDED, place "NA" as the value.
    - **legal-entity-text**: A textual description describing the party's legal entity status.
- **assignees-info:** root element with one or more assignees, contains fields:
    - **party-name:** Identifies the party (person or organisation name) receiving an interest or transaction. If it is a person and has a title, remove the title.
    - **country:** The party's country location. Put "USA" as the value if the country name is a variant of USA and fill the value as "UK" if the country is a variant of the UK. When there is no explicit country-name, you can infer the country-name by state, when there is no explicit state, you can assume the value for nationality is the party's country location. In the case of when a country is indecisive or you get a value of NOT PROVIDED, place "NA" as the value.
    - **legal-entity-text:** A textual description describing the party's legal entity status
- **property-count:** number of properties for a a trademark assignment entry

Note:
1. If any field is empty, put 'NA' as the value
2. All the tag names are case-sensitive in the output XML file. You can refer to the sample **sample_output_task1.json** for the correct XML file structure.
3. The output, methodology, and documentation will be marked separately for this task.

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

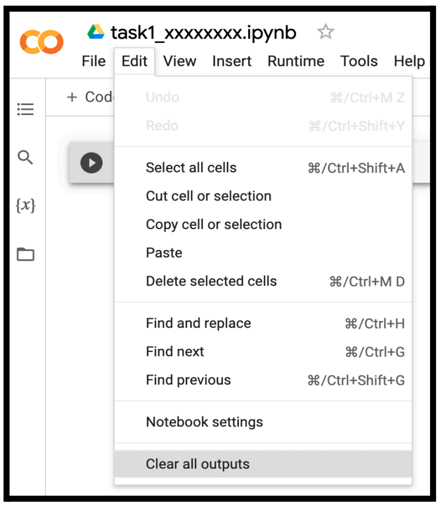
Step 2: Further process the extracted text from Step 1
- Convert the XML special characters (eg, &amp; to &)
- Save the data into a proper data format e.g. dataframe, dictionary...

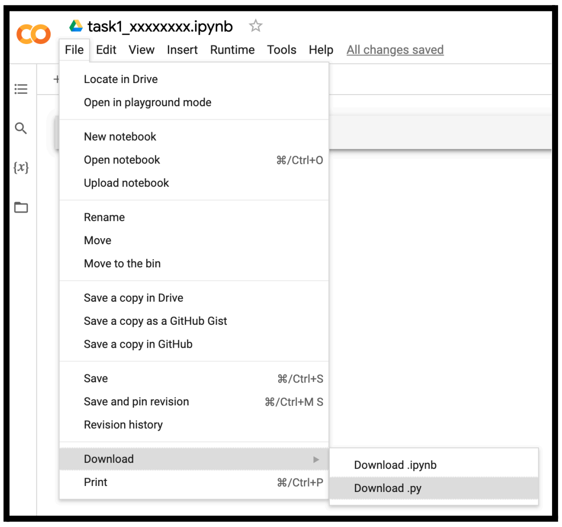
Step 3: JSON file output
- Use python library to transfer your data in step 2 into proper JSON format (make sure you check the spelling, upper/lower case, key names and name hierarchy of your JSON data)

Submission Requirements

You need to submit 3 files:
- A **task1_<group_number>.json** 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 - 25%
    - 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 - 25%
    - 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 (16/35)

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 youtube comments (in an Excel file) 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 machine to understand & decode patterns within a language. This step, though iterative, plays a significant role in deciding features for your machine learning model/algorithm.

Allowed libraries: ALL

Input Files	Output Files (submission)
Group<group_number>.xlsx	*<group_number>_channel_list.csv* *<group_number>_vocab.txt* *<group_number>_countvec.txt* *task2_<group_number>.ipynb* *task2_<group_number>.py*

Your group is provided with a unique dataset containing youtube comments (see **sample_input_task2.xlsx**). Please use the data file with your group_number, i.e. ***<group_number>.xlsx*** in the Google drive folder (student_data). The excel file contains worksheets with many youtube comments data. The excel tables have two columns:

- id: Unique comment identifier
- Snippet: a JSON array that contains information about one top level comment for a particular youtube video, such as the comment text, the channel and video the comment is under.

You are asked to extract the **'textOriginal'** fields in all top level comments for all youtube video lists that we provide to you. 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>*_channel_list.csv** contains unique channel ids along with the counts of top level comments(all language, and English only).
- ***<group_number>*_vocab.txt** comprises unique stemmed tokens sorted alphabetically, presented in the format of **token_index:token**, as outlined in Guideline step 4.
- ***<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**, as outlined in Guideline step 5.

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: Data import

- Each excel file contains multiple worksheets. Data are positioned differently in each worksheet.
- You are required to combine all data together and remove any duplicates to perform the next step

Step 2: Text extraction and cleaning

- You are required to extract the 'textOriginal' fields in all top level comments.
- Since the comment data contain emojis, you will need to remove the emojis and normalise the text into lower case for further analysis.
    - To remove emojis, make sure your text data is in utf-8 format.
    - The list of emojis to remove are in emoji.txt.
- You only require to extract the vocab and countvec lists for english comments from Channels that **have at least 15 english comments** by using the langdetect library with DetectorFactory.seed = 0. **Note you are deciding the language on a comment level, not a sentence level.**

Step 3: Generate csv file
- Generate a csv file that contains unique channel ids along with the counts of top level comments(all language, and english).
- The column names are: channel_id, all_comment_count, eng_comment_count.

Step 4: 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 **99% of channels_ids that have at least 15 english comments**.
    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 appear in less than **1% channels_ids that have at least 15 english comments** .
    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 comment.
    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 5: Generate the sparse numerical representation and output as countvec.txt
1. Re-tokenize your text based on the bigram list generated in step 4 if necessary.

2. Generate **sparse representation** by using the countvectorizer() function OR directly count the frequency using FreqDist().
3. Mapping the generated token with the vocabs in step 4 if need
4. Output the sparse numerical representation into txt file with the following format:
   channel_id1,token1_index:token1_frequency,
   token2_index:token2_frequency, token3_index:token3_frequency, …
   channel_id2,token2_index:token2_frequency,
   token5_index:token5_frequency, token7_index:token7_frequency, …
   channel_id3,token6_index:token6_frequency,
   token9_index:token9_frequency, token12_index:token12_frequency, …

Submission Requirements

You need to submit 5 files:

1. A **<group_number>_channel_list.csv** file contains information about the top level comment count (all language, and english) for each channel id
2. 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.
3. A **<group_number>_countvec.txt** file, in which each line contains the sparse representations of one of the channel in the following format:

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

Requirements on the Python notebook (report)

- Methodology - 25%
  - 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 - 25%
  - 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: Development History (2/35)

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)
<group_number>_development_history_task1.pdf
<group_number>_development_history_task2.pdf

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.

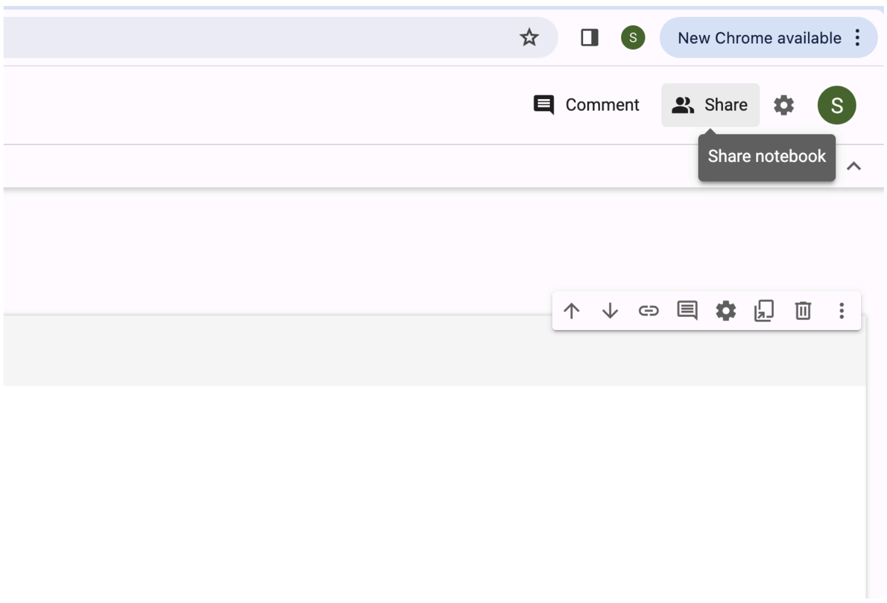
2. Version Screenshots: Include screenshots or snapshots of different versions of your assignment at each time point. This should clearly illustrate the incremental development and any modifications made to the project.

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.

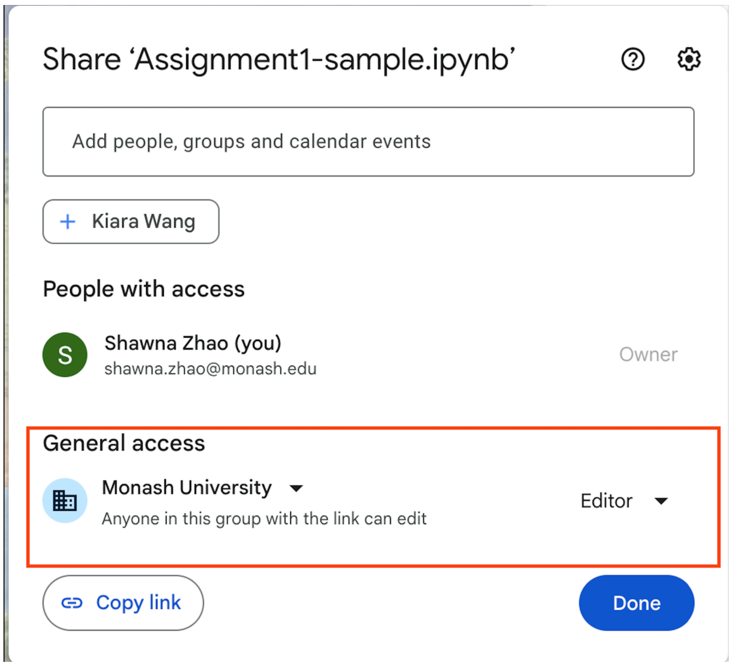
Note: We only require brief descriptions here, they don't have to be long as long as they reflect the key components we listed above. We recommend you to use Google Colab to complete your assignment, as Google Colab notebooks provide a comprehensive version history.

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

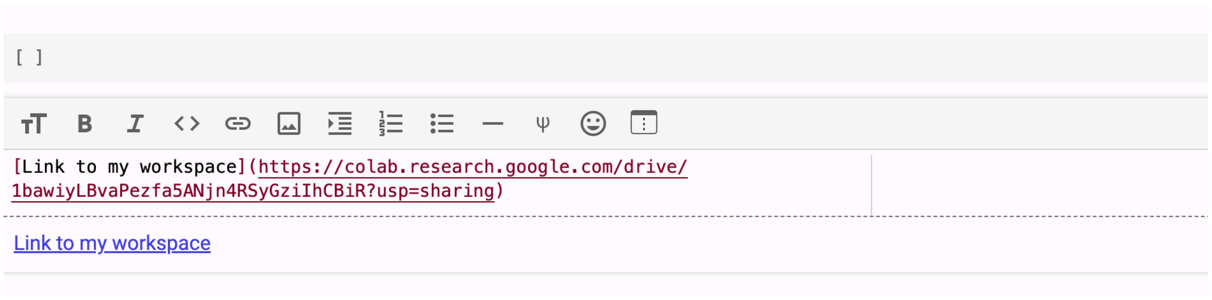
People with access

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

General access

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

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



```
[ ]
```

```
[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 **10 files** in your compressed zip file
- ☐ **<group_number>** should be replaced with **your group id**
- ☐ 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.