FIT5196-S2-2022 Assessment 1

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

Due date: Wednesday, August 31st 23:55

Text documents, such as reviews, 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 data-set. 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 Text Files (60%)

This assessment touches on the very first step of analysing textual data, i.e., extracting data from semi-structured text files. Each student is provided with a data set that contains information about reviews from retail company users on its products (please find your own directory for Task1 from here). Note that if you mount Google Drive in the Google Colab environment (as explained in the tutorials) you can access your own file directly with a unique URL (i.e., "/content/drive/Shareddrives/FIT5196-s2-2022/A1/Task1/input_data/<stdno>/"). Each text file contains information about the reviews, i.e., **"product ID", "review date", "review text",** and the **"review summary"**. Your task is to extract the data from the text file and transform the data into a **CSV** format with the columns:

1. **PID**: the product ID
2. **latest_review_date**: the latest review dates of all the reviews for a particular product
3. **product_review**: list of all the review texts for a product
4. **review_summary**: list of all the review summaries for a product

Please note that the **re, os and the Pandas** packages in **Python** are the only packages that you are allowed to use for task 1 of this assessment. Any other packages that you need to "import" before usage is not allowed.

The CSV file must be in the same structure as the sample solution provided here. **Please note that, as we are dealing with large datasets, the manual checking of outputs is impossible and output files would be processed and marked automatically therefore, any deviation from the CSV structure (i.e. task1_sample_output.csv) such as wrong column names which can be caused by different spelling, different upper/lower case, etc., wrong data types, etc. will result in receiving zero for the output mark.** (hint: run your code on the provided sample input and make sure that your code results in the exact **same structure** (not necessary content) as the sample output.

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

The output and the documentation will be marked separately in this task, and each carries its own mark.

Output (50%)

See sample.csv for detailed information about the output structure. The following must be performed to complete the assessment.

- Designing efficient regular expressions in order to extract the data from your textual dataset
- Storing and submitting the extracted data into an CSV file, **<your_student_number>.csv** following the format of **sample.csv**
- Explaining your code and your methodology in **task1_<your_student_number>.ipynb with all the cells' outputs.**
- A .py file, "**task1_<your_student_number>.py** ". This file will be used for plagiarism check.

Methodology (25%)

The report should demonstrate the methodology (including all steps) to achieve the correct results.

Documentation (25%)

The solution to getting the output must be explained in a well-formatted report (with appropriate sections and subsections). Please remember that the report must explain both the obtained results and the approach to produce those results. You need to explain both the designed regular expression and the approach that you have taken in order to design such an expression.

Task 2: Text Pre-Processing (40%)

This task touches on the next step of analysing textual data, i.e., converting the extracted data into a numeric representation. In this task, you are required to write Python code to preprocess a set of articles about user reviews and convert them into numerical representations (which are suitable for input into recommender-systems/ information-retrieval algorithms).

The data set that we provide is a single PDF file that contains many user reviews on different products. Please find your PDF file from the folder "input_data" from this link. Note that if you mount Google Drive in the Colab environment (as explained in the tutorials) you can access your own file directly with a unique URL (e.g., "/content/drive/Shareddrives/FIT5196-s2-2022/A1/Task2/input_data/<stdno>.pdf"). Your task is to extract and transform the information from the PDF file by performing the following tasks:

1. Generate the corpus vocabulary with the same structure as **sample_vocab.txt**. Please note that the vocabulary must be sorted alphabetically.
2. For each product ID, generate the sparse representation (i.e., doc-term matrix) of the PDF file according to the structure of the **sample_countVec.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 with the teaching team if in doubt.

1. The word tokenization must use the following regular expression, **"[a-zA-Z]+(?:[-'][a-zA-Z]+)?"**
2. 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 more than **ceil(Number_of_PIDs / 2**).
3. Tokens should be stemmed using the **Porter** stemmer.
4. Rare tokens (with the threshold set to less than **10 PIDs (i.e. 10 PIDs)**) must be removed from the vocab.
5. Creating the sparse matrix using countvectorizer.
6. Tokens with a length less than 3 should be removed from the vocab.
7. First 200 meaningful bigrams (i.e., collocations) must be included in the vocab using **PMI** measure.
8. Calculate the vocabulary containing both unigrams and bigrams.

Please note that you are allowed to use any Python packages as you see fit to complete the task 2 of this assessment. The output and the documentation will be marked separately in this task, and each carries its own mark.

Output (50%)

The output of this task must contain the following files:

1. **task2_<your_student_number>.ipynb** which contains your report explaining the code and the methodology
2. A py file, "**task2_<your_student_number>.py** ". This file will be used for plagiarism checks.
3. **<student_number>_vocab.txt**: It contains the **bigrams and unigrams** tokens in the exact format of **sample_vocab.txt.** Words in the vocabulary must be sorted in alphabetical order.
4. **<student_number>_countVec.txt:** Each line in the txt file contains the sparse representations of **one PID** of the review data in the format of **sample_countVec.txt**

Similar to task 1, in task 2, any deviation from the sample output structures may result in receiving zero for the output. So please be careful.

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

Methodology (25%)

The report should demonstrate the methodology (including all steps) to achieve the correct results.

Documentation (25%)

The solution to getting the output must be explained in a well-formatted report (with appropriate sections and subsections). Please remember that the report must explain both the obtained results and the approach to produce those results.

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.