FIT5196-S1-2022 Assessment 1

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

Due date: 11:55 PM, Friday, April 8.

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 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 a 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-s1-2022/A1/Task1/input_data/<stdno>/"). Each text file contains information about the reviews, i.e., **"reviewer ID", "product ID", "reviewer name", "number of helpful flags", "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 **XML** format with the following elements:

1. users: this tag wraps all the 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 product ID, review text, review date, review_helpful info, and review summary of the user review

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

The XML 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 XML structure (i.e. task1_sample_output.xml) such as wrong key names which can be caused by different spelling, different upper/lower case, etc., wrong hierarchy, not handling the XML special characters 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. We also provided you with this output checker in which you can verify the structure of your output files. **You can also use other online web applications such as xmlviewer to better understand the structure of the output.**

VERY IMPORTANT NOTE: The provided output checker and 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.xml 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 XML file, **<your_student_number>.xml** following the format of **sample.xml**
- Explaining your code and your methodology in **task1_<your_student_number>.ipynb with all the cells' outputs.**
- A pdf file, "**task1_<your_student_number>.pdf** ". You can first clean all the output in the jupyter notebook **task1_<your_student_number>.ipynb** and then export it as a pdf file then run all the codes again to make sure your ipynb file has all the outputs. This pdf will be passed to Turnitin for plagiarism check.

Methodology (25%)

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

Documentation (25%)

The solution to get 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 Excel file which contains many user reviews on different products for the past few years. Please find your Excel 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-s1-2022/A1/Task2/input_data/<stdno>.xlsx”). Your task is to extract and transform the information of the Excel file performing the following task:

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 unique date (articles come with a date at their title), generate the sparse representation (i.e., doc-term matrix) of the Excel file according to the structure of the **sample_countVec.txt. The review texts and the review summaries of the same date must be concatenated before converting to the vector representation.** The order of concatenation is not important for us (e.g., assuming “review1” and “review2” are both written on the same day, then you can either do review1+review2 or review2+review1).

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 tutorials.** If in doubt, you are encouraged to ask questions and discuss with the teaching team.

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_days / 2**).
3. Tokens should be stemmed using the **Porter** stemmer.
4. Rare tokens (with the threshold set to less than **10 days (i.e. 10 unique dates)**) 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 pdf file, "**task2_<your_student_number>.pdf** ". You can first clean all the output in the jupyter notebook **task2_<your_student_number>.ipynb** and then export it as a pdf file. This pdf will be passed to Turnitin for plagiarism check.
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 day** 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. You can use this output checker to verify the correctness of your output structure.

VERY IMPORTANT NOTE: The output checker 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 get 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.