FIT5196-S2-2020 assessment 1

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

Due date: **Wednesday, 9 September 2020, 11:55 PM**
Text documents, such as crawled web data, are usually comprised 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 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 (%55)

This assessment touches the very first step of analyzing textual data, i.e., extracting data from semi-structured text files. Each student is provided with a data-set that contains information about COVID-19 related tweets (please find your own directory "part1" from here). Each text file contains information about the tweets, i.e., "id", "text", and "created_at" attributes. Your task is to extract the data and transform the data into the **XML** format with the following elements:

1. id: is a 19-digit number.
2. text: is the actual tweet.
3. Created_at: is the date and time that the tweet was created

The XML file must be in the same structure as the sample folder. **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. test_output.xml and test_output.json files) and any deviation from this structure (e.g. wrong key names which can be caused by different spelling, different upper/lower case, etc., wrong hierarchy, not handling the XML special characters,...) will result in receiving zero for the output mark as the marking script would fail to load your file.** (hint: run your code on the provided example and make sure that your code results in the exact same output as the sample output. **You can also use the "xmltodict" package to make sure that your XML is loadable**). Beside the XML structure, the following constraints must also be satisfied:

1. The "id"s must be unique, so if there are multiple instances of the same tweets, you must only keep one of them in your final XML file.
2. The non-english tweets should be filtered out from the dataset and the final XML should only contain the tweets in English language. For the sake of consistency, you **must** use the langid package to classify the language of a tweet.
3. The **re, os,** and the **langid** 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 output and the documentation will be marked separated 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 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**
- 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. 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 (%45)

This assessment touches on the next step of analyzing textual data, i.e., converting the extracted data into a proper format. In this assessment, you are required to write Python code to preprocess a set of tweets and convert them into numerical representations (which are suitable for input into recommender-systems/ information-retrieval algorithms).

The data-set that we provide contains 80+ days of COVID-19 related tweets (from late March to mid July 2020). Please find your ***.xlsx*** file from the folder "part2" from this link. The excel file contains 80+ sheets where each sheet contains 2000 tweets. 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 day (i.e., sheet in your excel file), calculate the top 100 frequent unigram and top-100 frequent bigrams according to the structure of the **sample_100uni.txt** and **sample_100bi.txt**. If you have less than 100 bigrams for a particular day, just include the top-n bigrams for that day (n<100).
3. Generate the sparse representation (i.e., doc-term matrix) of the excel file according to the structure of the **sample_countVec.txt**

Please note that the following steps must be performed (**not necessarily in the same order**) to complete the assessment.

1. Using the **"langid"** package, only keeps the tweets that are in **English** language.
2. The word tokenization must use the following regular expression, **"[a-zA-Z]+(?:[-'][a-zA-Z]+)?"**
3. The context-independent and context-dependent (with the threshold set to more than **60 days**) stop words must be removed from the vocab. The provided context-independent stop words list (i.e, **stopwords_en.txt**) must be used.
4. Tokens should be stemmed using the **Porter** stemmer.
5. Rare tokens (with the threshold set to less than **5 days**) must be removed from the vocab.
6. Creating sparse matrix using countvectorizer.
7. Tokens with the length less than 3 should be removed from the vocab.
8. First 200 meaningful bigrams (i.e., collocations) must be included in the vocab using **PMI** measure.

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 following 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 tweet data in the format of **sample_countVec.txt**
5. **<student_number>_100uni.txt and <student_number>_100bi.txt :** Each line in the txt file contains the top 100 most frequent uni/bigrams of **one day** of the tweet data in the format of **sample_100uni.txt and sample_100bi.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.

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.