Assessment Description

Text documents, such as long recordings and meeting transcripts, are usually comprised of topically coherent text segments, each of which contains some number of text passages. Within each topically coherent segment, one would expect that the word usage demonstrates more consistent lexical distributions than that across segments. A linear partition of texts into topic segments can be used for text analysis tasks, such as passage retrieval in IR, document summarization, and discourse analysis. In this assessment, you are required to write Python code to preprocess a set of meeting transcripts and convert them into numerical representations suitable for input into topic segmentation algorithms.

The detailed tasks are as follows:

- **Task 1: Reconstruct meeting transcripts with topical boundaries.** The original meeting transcripts are stored in three different types of XML files, which are ending with ".words.xml", ".topic.xml" and ".segments.xml". (The details about the three types of files can be found in Section 3 below). The task here is to reconstruct the original meeting transcripts with the corresponding topical and paragraph boundaries from these files. Please note that
  - A meeting transcript must be generated for each of the "*.topic.xml" file. For example, "ES2002a.txt" will be generated from "**ES2002a**.topic.xml".
  - All the generated meeting transcripts with the ".txt" file extension must be saved in the folder "txt_files".
  - The topical boundaries must be denoted with "**********" (i.e., 10 asterisks).
  - All the tokens, including punctuations, must be separated by a white space. For example, "Alright , okay . Okay ."
  - Besides the topical boundaries, the paragraph boundaries must also be reconstructed with the "*.segments.xml" file.
  - The input files to your notebook "task_1.ipynb" must be the three types of XML files. The output must be the meeting transcripts saved in a set of txt files.
  - A sample meeting transcript is provided in the "txt_file" folder.

- **Task 2: Generate sparse representations for the meeting transcripts.** The aim of this task is to build sparse representations for the meeting transcripts generated in task 1, which includes word tokenization, vocabulary generation, and the generation of sparse representations. Please note that
  - The word tokenization must use the following regular expression, **"\w+(?:[-']\w+)?",** and all the words must be converted into the lower case.
  - The stop words list (i.e, **stopwords_en.txt**) provided in the zip file must be used.
  - The words, whose document frequencies are greater than 132, must be removed.
  - Generating multi-word phrases (i.e., collocations) are not needed.
  - The output of this task must contain the following files:

- **vocab.txt**: It contains the **unigram** vocabulary in the following format, **word_string:integer_index.** Words in the vocabulary must be sorted in alphabetic order. For example, "absolute:22" in the following figure means that the 23rd word in the vocabulary is "absolute".

```
a_n_:4
a_p_o_g_e_e_:5
a_s:6
a_s_r_:7
a_v_:8
abandon:9
abandoned:10
abbie:11
abbing:12
abbreviations:13
abdul:14
abigail:15
abilities:16
ability:17
abo:18
abou:19
abrupt:20
abs:21
absolute:22
absolutely:23
```

- **topic_seg.txt:** It contains the topic boundaries encoded in boolean vectors. For example, if a meeting transcript "ES2018d.txt" contains 10 paragraphs in total after being preprocessed, and there are topic boundaries after the 2nd, 5th, and 7th paragraphs, the boolean vector must be "**ES2018d:0,1,0,0,1,0,1,0,0,1**". Every line in *topic_seg.txt* corresponds to one meeting transcript.
- **./sparse_files/*.txt** : Each txt file in the "sparse_files" folder corresponds to one of the meeting transcripts in the "txt_files" folder, and they have the same file name. For example, "*./sparse_files/ES2002a.txt*" corresponds to "*./txt_files/ES2002a.txt*". Each file in "**/sparse_files**" contains the sparse representations for all its paragraphs as

```
6591:1,5763:1,5040:1,7292:1,216:1,7279:1,9362:2
10269:1
5664:1,6883:1,8646:1,3762:1,7153:1
4599:1,9697:1,8078:1,8505:1,2264:1,718:1,9398:1,3931:1
4534:1,2264:1,8150:1,10205:1,4526:1,6835:1,89:1,10195:1,2384:1,718:1,8938:1
10269:1
10269:1
9602:1,5436:1,3200:1,5456:1,2323:1,4062:1,8513:1
3753:1,3894:1,4380:1,6851:1,7153:3,6989:2,7116:1,4526:1,5863:1,2263:2,7487:1,289:1,2560:1
6785:1
6785:1
9607:1,7987:1,8488:1,5299:1,8475:2,5450:1,766:1
9256:1
3782:1,2997:1,8475:1,8457:2,5407:1,1936:1
5407:1
5338:1,5407:1,5409:2
5407:1
5338:1
4:1,2179:1
2180:1
8453:1,8475:1,1936:1
8475:1
8457:1,5393:1,9389:1,5554:1,2997:1
4257:1,5939:1
5407:1,8457:1
```

where 1) each line is a paragraph and the order of the lines must match the

paragraph order in the corresponding meeting transcript. 2) the integer before ":" is the word index in the vocabulary and the one after is the frequency of the word in the corresponding paragraph; 3) empty paragraphs after preprocessing must be excluded.

3. Assessment Resources

Before you start writing your code, you will need to download the file

- meeting_transcripts

Unzipping the file, you will find that

- There are three types of XML files in the given folder :
    1. **./topics/***.topic.xml contains the information about topic segments. Each topic tag directly linked to the root indicates one topic segment that is required in text segmentation task. Each topic segment can contain a number of paragraphs given by different meeting attendees. It can also contain sub-topics.
    2. **./words/***.words.xml contains the word tokens generated with the force alignment technique. Each word is associated with its start time and end time in the meeting transcript.
    3. **./segments/***.segments.xml contains the paragraph boundaries, the start and end of which are denoted by the corresponding word IDs.
- ./spase_files: the file folder used to store the generated sparse representations for all the meeting transcripts.
- ./txt_files: the file folder used to save the reconstructed meeting transcripts.
- ./stopwords_en.txt: the stopword list used in word tokenization.
- ./topic_segs.txt: the file used to save the topical boundaries.
- ./vocab.txt: the file used to save the vocabulary.
- ./task_1.ipynb: the python code you are going to write for task 1
- ./task_2.ipynb: the python code you are going to write for task 2

4. Assessment Criteria

The following outlines the criteria which you will be assessed against.

4.1 Mark allocation and general marking criteria

1. The submitted scripts in the notebook should work **without any errors** and must give **the correct results**. If the submitted notebook cannot be run by the assessor, which will be double-checked by the head tutor and the lecturer, zero marks will then be given to the corresponding task.
    - task 1: 14 out of 30

    - task 2: 14 out of 30
    - task 2 will be assessed **if and only if** task 1 is successfully finished and receives a full mark (i.e., 14).
2. The code should be well structured and properly commented. (1 out of 30)
3. The notebook should be structured in a logical way so that it clearly shows how students finish the tasks in the assessment. (1 out of 30)
4. Criteria 2 and 3 will be assessed **if and only if** the mark for criteria 1 is greater than and equal to 25.

5. How to Submit

Once you have completed your work, take the following steps to submit your work.

1. Only one zip file needs to be submitted: Once you finished the tasks, please zip the folder **only contains the files specified in Section 3**, including the original XML files.