Comp 479/6791 # Project 1 **Fall 2020**

Due date: September 30, 2020

Goal: text preprocessing with NLTK, proofreading results

Data: Reuter's Corpus RCV1 `http://www.daviddlewis.com/resources/testcollections/reuters21578/`

Note that you should always retain the original corpus. Speaking of text 'cleaning' and of stop word 'removal' is a sloppy short form for creating a clean second version of your data without stop words. Also, you may find out that you removed items that later turn out to be valuable. Thus for small collections like Reuter's, make a read only copy of the original.

Overview: Use NLTK for this project. Download the Reuter's corpus onto your computer. Use that version of the corpus, not the one available in NLTK.

Develop a pipeline of steps to

1. read the Reuter's collection and extract the raw text of each article from the corpus
2. tokenize
3. make all text lowercase
4. apply Porter stemmer
5. given a list of stop words, remove those stop words from text. Note that your code has to accept the stop word list as a parameter, do not hardcode a particular list

A pipeline means that every step can be executed in stand-alone fashion with the appropriate input and will generate output suitable as input for the next module. Manually proofread every step in your pipeline.

Description: Each step in your pipeline has specific additional requirements in order to be considered satisfactory. It is important that you do not limit yourself to the requirements, but think beyond the minimum requirements for your solution. Requirements are of two different kinds: some help us grade, some specify text processing variants.

To assist the markers in properly assessing your code when there are no in-person demos, you find here a package of Python code that essentially creates a standard pipline shell or template. Your modules have to obey certain formatting constraints and have to be inserted into the templates. The templates call each module a block. Simply add your code in the proper place in the appropriate script.

Each block has it's own objectives to satisfy along with specific input and output structures. Read the documentation part at the beginning of each of the following scripts thoroughly, in order to know what you are expected to observe in your code. **DO NOT MODIFY ANY CODE IN THESE FILES.**

The Python script template stubs for submitting pipeline modules are called

- `block-1-reader.py`
- `block-2-document-segmenter.py`
- `block-3-extractor.py`

- `block-4-tokenizer.py`
- `block-5-stemmer.py`
- `block-6-stopwords-removal.py`

You have to complete the stubs in the file

- `solutions.py`

You may create additional functions and embed them in `solutions.py`.

To familiarize yourself with the specific output structure required for each module, pre-built simple test cases are provided in

- `assert.py`

The `assert` messages will help you understand where you are making a mistake in terms of the input and output values expected by the templates. Read them carefully. Note that you still need to construct your own test cases for the text processing aspect. Again, **DO NOT ATTEMPT TO MODIFY ANY CODE IN THIS FILE.**

Commands to execute each block (python script) can be found below. Each block has a default set of parameters that you can pass in when calling it. Some parameters are mandatory, others are optional, depending on the block.

Parameter options:

`-h, --help` will show description of all parameters and exit

`-i INPUT_FILE, --input_file INPUT_FILE` input from file, default stdin

`-o OUTPUT_FILE, --output_file OUTPUT_FILE` output to file, default stdout

`-p PATH, --path PATH` directory containing input, here Reuter's

`-s STOPWORDS, --stopwords STOPWORDS` path to stopword list from file

See README file for more detail on the script. For all modules, create your own test cases for more general solutions.

Deliverables: a folder named "Deliverables" to be submitted in Moodle before 30 September, 2020 must include

- (6pts) `solution.py` file containing your pipeline
- (2pts) output files returned from each of the six blocks in the pipeline for the first five documents in the collection
- (4pts) Report: a .pdf document of no more than 5 pages that explains your work and submitted modules. Make it readable.

Marking scheme: Grading is based on two components. The first component is adherence to the instructions. This is expected to be 100%. The second component is what you bring to the task. We can only appreciate this, if it is well documented in the report and in the code and illustrated with convincing examples.