Airi Chow (#40003396)
COMP 479
Fall '20

Project #1

Using Python3
~~Yellow Highlight: Issue with running.~~
Green Highlight: Given names for outputs/inputs.

Notes:
- "Root Folder" consists of a copy of how my directory looks like after running all the commands below. This has been tested for pipelined version and file/path input version which results in the same output.
- "Individuals" consists of the collection segmented into individual files/articles/documents...
- "Step1 and Step 2" consists of the entire collection.
- "Step 3 to Step 6" consists of using the first 5 documents which is to be processed. This can be changed by modifying the '5' parameter to the desired number of documents in "Step #3" though it would take a while to run on the entire collection :3

Step #1 – Reading the Reuter's collection (block-1-reader.py)

Tested Commands

- **python3 ./block-1-reader.py --path reuters21578**
- **python3 ./block-1-reader.py --path reuters21578 -o collection.json**

==========
Input: reuters21578 (Corpus' path)
Output: Creates a JSON file of the entire collection of SGM 000 to 021 data as a string.
==========

- Changing directory to /reuters21578 to scan for .sgm files.
- open(file_name,'r', errors='ignore'): Opening files but set errors parameter to ignore due to unicode issue in sgm17.
Error:
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xfc in position 1519554: invalid start byte

Step #2 – Extracting the raw text of each article from the corpus (block-2-document-segmenter.py)

Tested Commands

- **python3 ./block-2-document-segmenter.py -i collection.json**
- **python3 ./block-2-document-segmenter.py -i collection.json -o documents.json**
- **python3 ./block-1-reader.py --path reuters21578 | python3 ./block-2-document-segmenter.py**
- **python3 ./block-1-reader.py --path reuters21578 | python3 ./block-2-document-segmenter.py -o documents.json**

==========

- Input: collection.json contains all the documents or pipelined stdin data from
- Output: Creates a JSON file with 5 documents.

==========

- **BeautifulSoup must be installed to parse the documents.
- **JSON is needed to parse the pipelined data
- replace("reuters", "REUTERS") is used to satisfied the assert <REUTERS> since html parser lower-cases the tags.
- find_all('reuters',limit=5): Parse 5 documents as for the output required.
- sys.stdin.isatty(): Check if stdin filestream is being used.
- sys.stdin.read(): Read the pipelined data
- str(document): Change the “tag” object to “string”
- enumerate(documents): Used to iterate but keep an index variable

Step #3 – Extraction
(block-3-extractor.py)

Tested Commands

- **python3 ./block-3-extractor.py -i documents.json**
- **python3 ./block-3-extractor.py -i documents.json -o articles.json**
- **python3 ./block-1-reader.py --path reuters21578 | python3 ./block-2-document-segmenter.py | python3 ./block-3-extractor.py**
- **python3 ./block-1-reader.py --path reuters21578 | python3 ./block-2-document-segmenter.py | python3 ./block-3-extractor.py -o articles.json**

==========

- Input: documents.json contains 5 documents
- Output: 5 dictionaries with ID and Text

==========

- **BeautifulSoup must be installed to parse the documents.**
- document_id: Consists of ID with whitespace and escape characters

- document_id_arr: Consists of an array with one element (the ID #). Python's regex library is used to parse the document_id without the escaped characters and whitespace.
- document.body.contents[0]: Used to get the body tag's text
- str(text): Cast the iterable string to a normal string.
- re.sub(r"[\\]+","",text): Remove escaped backslash lines

Step #4 – Tokenisation (block-4-tokeniser.py)

Tested Commands

- **python3 block-4-tokenizer.py -i articles.json**
- **python3 block-4-tokenizer.py -i articles.json -o tokens.json**
- ***python3 ./block-1-reader.py --path reuters21578 | python3 ./block-2-document-segmenter.py | python3 ./block-3-extractor.py | python3 ./block-4-tokenizer.py***
- ***python3 ./block-1-reader.py --path reuters21578 | python3 ./block-2-document-segmenter.py | python3 ./block-3-extractor.py | python3 ./block-4-tokenizer.py -o tokens.json***

==========
- Input: articles.json contains dictionaries with ID and Text
- Output: Lists of Tokens [ID, Token]
==========

- **NLTK must be installed to tokenise the texts.
- **JSON is needed to parse the pipelined data
- word_tokenize(full_text): Use NLTK to tokenise the text.
- Since the pipelined document needs to be properly formatted into JSON, some replacements are needed.
    - data.replace("}","},")
    - data= re.sub(r"{","[{",data, count=1) #For start of index
    - data= re.sub(r"},$","}]",data, count=1)

Step #5 – Apply Porter Stemmer (block-5-stemmer.py)

Tested Commands

- **python3 ./block-5-stemmer.py -i tokens.json**
- **python3 ./block-5-stemmer.py -i tokens.json -o stems.json**
- ***python3 ./block-1-reader.py --path reuters21578 | python3 ./block-2-document-segmenter.py | python3 ./block-3-extractor.py | python3 ./block-4-tokenizer.py | python3 ./block-5-stemmer.py***

- ***python3 ./block-1-reader.py --path reuters21578 | python3 ./block-2-document-segmenter.py | python3 ./block-3-extractor.py | python3 ./block-4-tokenizer.py | python3 ./block-5-stemmer.py -o stems.json***

==========

- Input: tokens.json contains lists with ID and Text
- Output: Lists of Stemmed Tokens [ID, Token]

==========

- **NLTK must be installed to get the PorterStemmer
- **JSON is needed to parse the pipelined data
- sys.stdin.readlines(): Read line by line to parse the token in JSON array
- stemmer.stem(1): Stems the token which is 2nd element of each pair of list.
- (int(token[0]), token_stem): Casts the ID which is of type string into an int.

Step #6 – Given a list of stop words, remove those stop words from text. (block-6-stopwords-removal.py)

Tested Commands

- **python3 ./block-6-stopwords-removal.py -i stems.json**
- **python3 ./block-6-stopwords-removal.py -i stems.json -s stopwords-sample.txt -o results.json**
- ***python3 ./block-1-reader.py --path reuters21578 | python3 ./block-2-document-segmenter.py | python3 ./block-3-extractor.py | python3 ./block-4-tokenizer.py | python3 ./block-5-stemmer.py | python3 ./block-6-stopwords-removal.py***
- ***python3 ./block-1-reader.py --path reuters21578 | python3 ./block-2-document-segmenter.py | python3 ./block-3-extractor.py | python3 ./block-4-tokenizer.py | python3 ./block-5-stemmer.py | python3 ./block-6-stopwords-removal.py -o results.json***

==========

- Input: stems.json contains lists with ID and stemmed text and optionally the path to a list of stopwords. "stopwords-sample.txt" consists of a subset of NLTK stopwords.
- Output: Lists of Tokens [ID, Token] without the stopwords.

==========

- **NLTK must be installed to get a list of stopwords to remove.
- **JSON is needed to parse the pipelined data
- stop_words = set(stopwords.words("english")): Retrieves a set of stopwords in english from NLTK library. This has to be downloaded!
- if stopwords_list: Checks if the path of the "stopwords" is valid and not empty. If it's not empty, it would iterate and adds these elements to an array.

- if stem_word not in stopwords_arr: Checks and adds the tokens if they are not in the previous array.
- if not stopwords_arr: Default if the path is invalid or the file is empty.
- if stem_word not in stop_words: Iterates through NLTK's full stopwords set.