Concordia University
Engineering and Computer Science

COMP 479/6791
Information Retrieval and Web Search

Assignment 1

Report

Under the guidance of Dr. Sabine Bergler

Vasu Ratanpara 40135264

October 2021

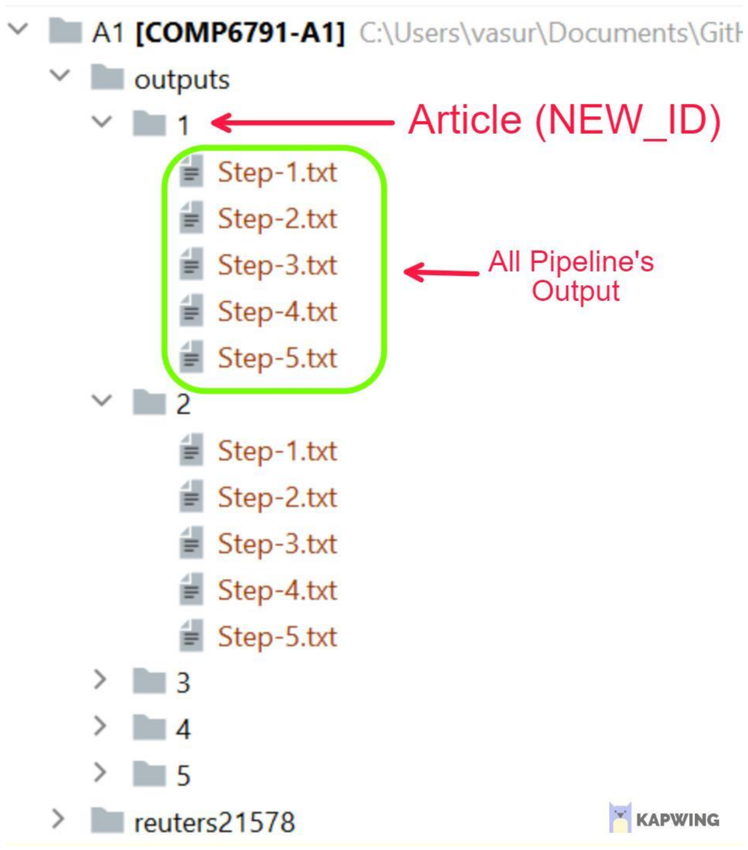
Introduction

In this project, We had to use the “NLTK” library for Stemming the tokens from Reuter’s Corpus (Reuters-21578). To achieve this task, we had to divide the whole process into separate stages. Each stage is considered here as a “Pipeline”. The whole project is consisting of a total of 5 pipelines where each has a particular task.

I have used the “NLTK” library just for Stemming the tokens (by using **Porter stemmer**) and tokenizing the extracted meaningful text (via **word tokenize)**. All other processes such as extracting the meaningful and usable text from Reuter’s Corpus collection, writing them to file etc. are done using Python’s standard libraries.

Pipelines

The in-depth details of each pipeline are mentioned below. Each of the pipelines is dependent on the output files which are generated by the previous pipeline (except pipeline 1). Every step can be executed in a stand-alone fashion with the appropriate input, and it will generate output suitable as input for the next pipeline.



Pipeline 1 (Extract raw text)

This is the first pipeline that is mainly responsible for reading the raw text from Reuter's Corpus collection. First, I created a list of all files which contain the articles and are from Reuter's Corpus collection. Afterwards, I am opening each file one by one and started extracting useful content.

To parse each article, I have divided each article with the **"</REUTERS>"** tag and stored it in the list. By taking leverage of regular expressions from Python 3. I extracted the **title** ("<TITLE>...</TITLE>"), **body** ("<BODY>...</BODY>") and **NEWID** of each article.

The content of each file will be stored under the output folder (which will be dynamically generated). Each article will be stored under with folder name which is the same as its **NEWID** attribute. Pipeline 1 will generate a "Step-1.txt" for each article under its own folder.

Pipeline 2 (Tokenization)

This pipeline mainly handles the task of tokenizing the extracted text from articles. In this pipeline, I am reading the raw text from input files that were generated in the previous pipeline. I have used the **word_tokenize** from the NLTK library to convert text into tokens. Before tokenizing, I have decided to clean the raw text. The reason behind cleaning the raw text is, it's easy to clean text as strings as compared to various tokens.

I have created a function called **text_cleaner()**. Which uses **replace()** and regular expressions to clean the text. I removed all various non-characters and full stops. Also, converted multiple whitespaces to single whitespaces. All numbers are also removed since it's not useful for our project.

During the tokenization, I have used **set()** to only store unique tokens from the output of **word_tokenize()** which removes redundant tokens from each article. At last, I am storing all tokens under their own folders as "Step-2.txt".

Pipeline 3 (Convert to lowercase)

This pipeline takes care of converting all tokens to lowercase so we can process over text easily without bise input (Due to the mixture of upper and lower text). The pipeline will read “Step-2.txt” for each article and store tokens in lowercase under the “Step-3.txt” name. Using **set()** again to avoid duplicate tokens (i.e. if there is the same token with upper and lowercase).

Pipeline 4 (Apply Porter stemmer)

The given pipeline handles the Text Normalization job. Text Normalization will convert each word into a single canonical form by using the stemming process. I have read all files (Step-3.txt) from the above pipeline for each article one by one and stemmed each token by using the object of **PorterStemmer()** from the NLTK library. The results were stored under the name “Step-4.txt” for each article individually.

Pipeline 5 (stop words remover)

In the last pipeline, it will ask the user to enter stop words separated by space. Once the user is done entering the words press enter key. It will read all the input files for each article. It will remove all the stop words from token lists and the new results will be stored as “Step-5.txt” for every article.

References

For this project, I have used various online resources but unfortunately, I forget to note down each of them since I didn’t think to add this section when I was working on the project. Hence, I accept that I have used various online resources (i.e., online articles, code snippets, books, etc.) for a better understanding of the NLTK library and Python 3 to do this project. But I assure you that I didn’t copy code from any resources. All work that I submitted is done by myself.