Name: Prathusha JS Naidu
Campus ID: DK07815
Email: prat2@umbc.edu

CMSC 676: Information Retrieval
Programming project: Phase 1

1. Introduction:

The objective of this assignment is to compare two approaches to tokenize and down case all words in a collection of HTML documents. Programming language used for this project is Python.
To run the program, type the following at a terminal :
$python python_tokenizer.py input output
($python <name of the program> <input directory> <output directory>)

Libraries necessary for the program to execute: NLTK and BeautifulSoup.

i) Input: A directory with 503 html documents. Name of this directory is given as the first parameter to the program while executing.
I have used the path.join function from the "os" library to get the accurate path to read files from the directory.

ii) Output: A directory that contains all the output documents. Each input files' tokenized output is written into it's respective output file whose name is same as the input file but with a ".txt" extension.
In addition to this there are two more text files:
token_sort = a file of all tokens and their frequencies sorted by token
freq_sort = a file of all tokens and their frequencies sorted by frequency

iii) Methodology:
Once the input and output directories are passed as parameters, each file is read from in the input directory. A library called BeautifulSoup is used to clean up the HTML contents and extract just the data. The function get_text() from this library does this. "Re" library is used to handle special characters and only alphanumeric characters are tokenized. Since numeric data can also be useful at times, I have tokenized them as words too.
NLTK library is used to tokenize the words in a document with the help of the function word_tokenize(). The result of this is a list data structure with all the split words. These words are written into each of their respective output files.

A global dictionary is updated with words from every new file .
The counter function from the collections library helps in obtaining the frequency of each tokenized word. The results from this function is used to sort the data by tokens and by frequency.

2) Evaluation:

The following graph shows the running time of the tokenizer in terms of system time as well as elapsed time. The running time increases with the increase in number of files. The graph has been plotted for six different number of input files – 50 , 100 ,200, 300,400 and 503

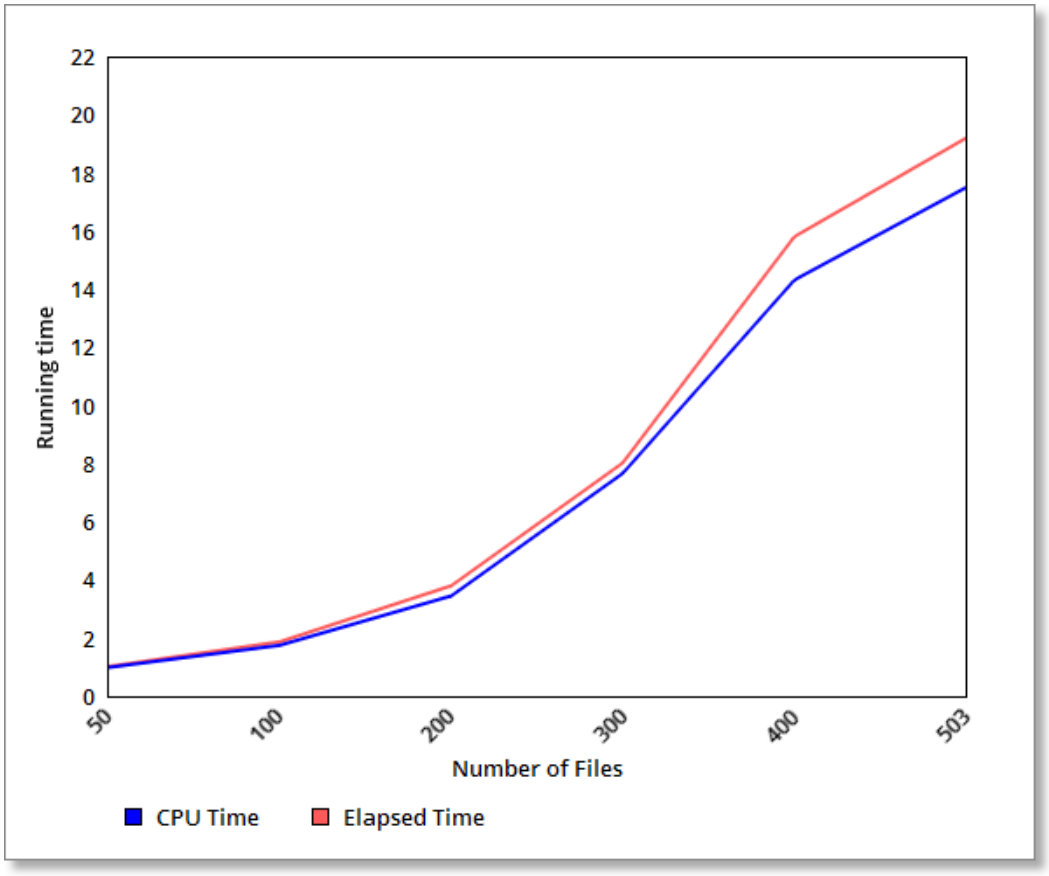

Fig 1: Number of files vs Running time

3) Comparison

The results were compared with results of programs by Tejus Chandrashekar (Tokenizer B), Akshay (Tokenizer C) and Madhura Belli (Tokenizer D)

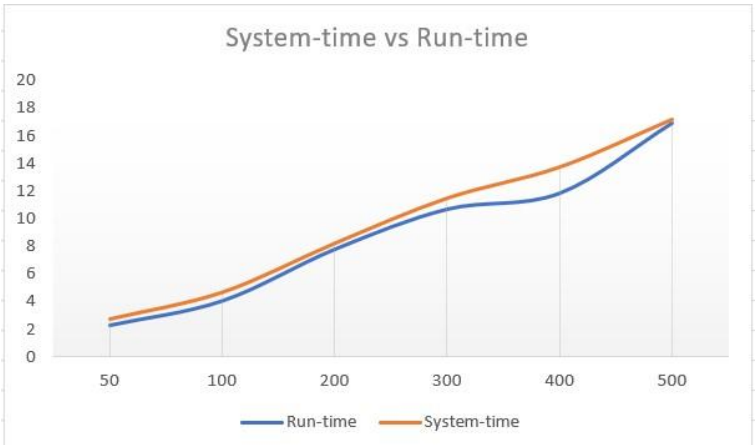

Fig 2: Tokenizer B

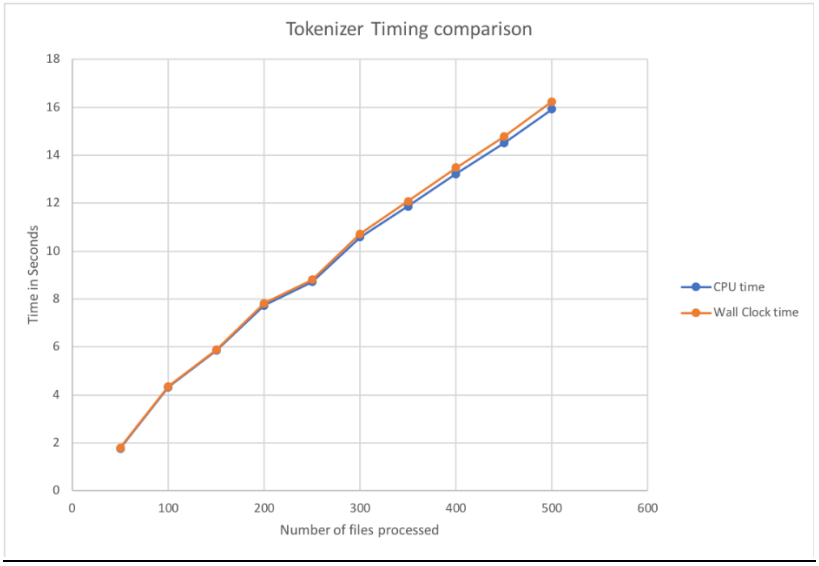

Fig 3: Tokenizer C

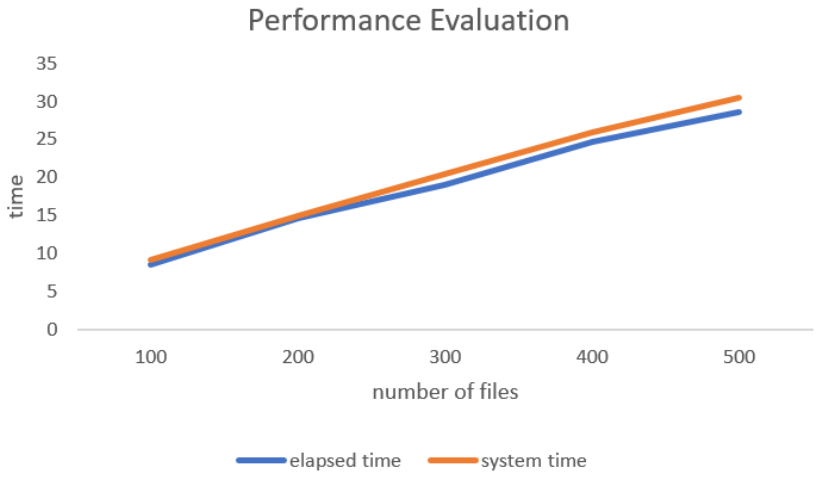

Fig 4 : Tokenizer D

4) Conclusion

The comparison results show that the running time almost always increases linearly with the number of files. Since each of us used different libraries to parse the documents and tokenize the html content, the running time parameters are not exactly same.

Though the documents were successfully processed to an extent, characters belonging to different languages and different encoding formats weren't tokenized accurately.