Assignment 2

Quan Mai quanmai@uark.edu
010946565

1. Introduction

The objective of this assignment is to build an inverted file (index) for the documents in the collection. The goal is to build a trio of files, dict, post, and map.

1. The dictionary should contain one entry per unique word:
   - the term
   - the number of documents that contain that term
   - the location of the first record in the postings file
2. The postings file should contain one entry per unique word per document:
   - the document id
   - the raw frequency of the word in the document
3. The mappings file should contain one entry per document
   - the document id
   - the filename

2. Implementation

2.1 Preprocessing

I use PLY (Python Lex-Yacc) as a lexer. PLY consists of two separate modules; *lex.py* and *yacc.py*. The *lex.py* module is used to break input text into a collection of tokens specified by a collection of regular expression rules. The preprocessing step is comprised of actions listed in the table 1.

2.2 Program Structure

Code is implemented in Python 3, I use Dictionaries (Python Hash-map datastructure) for the global and per-document hashtables. Python dictionaries use dynamic resizing as a key strategy for handling collisions and maintaining efficient performance. Dynamic resizing involves increasing the size of the dictionary's hash table (often doubling it) when a certain load factor is exceeded. The indexing is the built-in hash() function of python.

Object	Example	Replacement	Comment
email	sgauch@uark.edu	zzzemail	New
url	www.quanmai.github.io	zzzurl	New
html characters	< , >	zzzlessthan, zzzgreaterthan	New
phone number	1-234-567-8910	zzzphonenumber	
number	123112	zzznumber	
comment	/* here is comment */		
html tag	<html lang="en">		
html size	12px, -20px	zzzpixel	
character + number	character20whatmore	zzzcharnum	
color	#FFFF, #3122	zzzcolor	
down case	QuanMai	quanmai	

Table 1: Preprocessing step

3. Results

The program is run using the following command from terminal

```
./run.sh intput_dir/ output_dir/
```

The following files will be created in the output directory:

- dict_file.txt: contains 3 columns corresponding to the term, number of documents in which it appears and its location of the first record in the posting file. This file has 25956 rows corresponding to 25956 **unique** terms in all documents.
- post_file.txt: the posting file containing 2 columns: the document id and the raw frequency of the term in the corresponding document. This file has 289794 rows corresponding to the corpora size in all documents.
- mapping_file.txt: comprises of the document id and its corresponding file name. This file has 1750 rows corresponding the number of input files.

3.1 Complexity Analysis

Preprocessing As the lexer parses all the characters, the time complexity is $\mathcal{O}(\mathcal{N})$ for $\mathcal{N}$ is the total number of characters in all document.

Inverted Indexing Let M be the maximum number of term in a document after the preprocessing step, N the number of input documents.

- Time Complexity: $\mathcal{O}(M \times N)$
- Space Complexity: $\mathcal{O}(M \times N)$

3.2 Timing

The timing is shown in figure 1

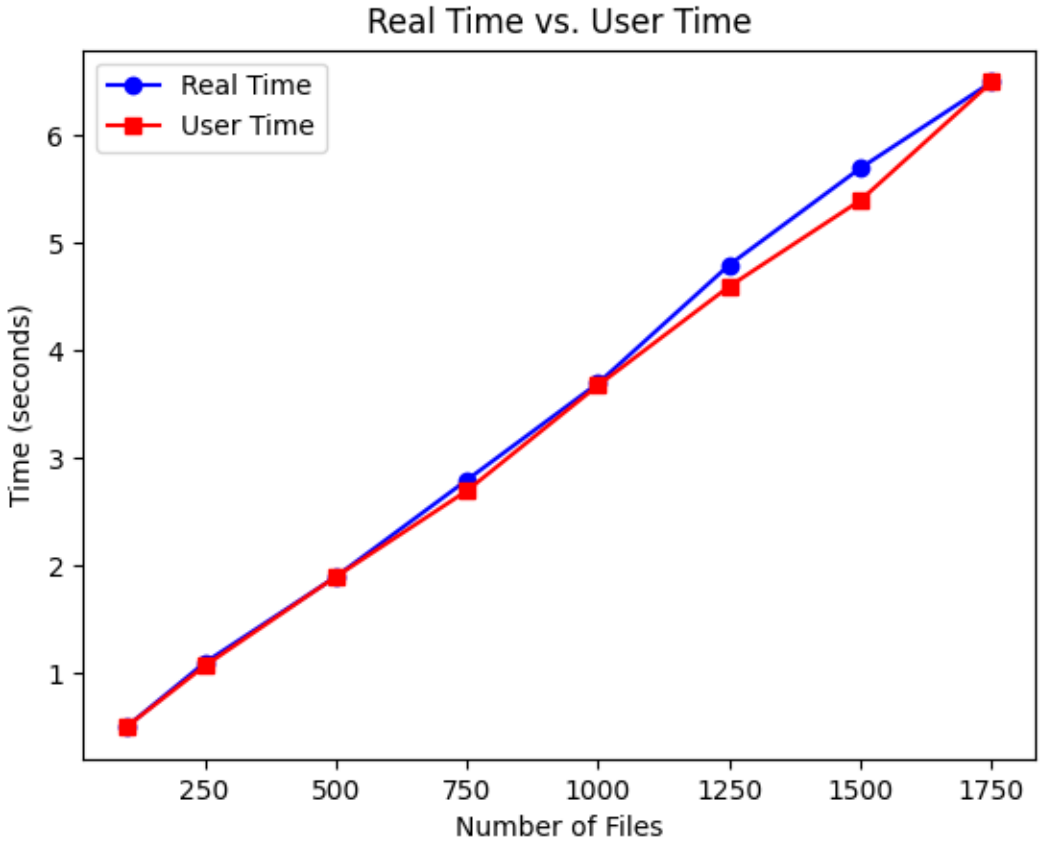

Figure 1: Real time and user time v.s. number of input files

Dictionary File			Posting File		Mapping File	
TERM	NUMDOC	START	DOCID	FREQUENCY	DOCID	FREQUENCY
function	78	78	0	16	0	0.html
w	182	260	26	1	1	1.html
d	263	523	47	1	2	2.html
s	864	1387	59	1	3	3.html
l	121	1508	101	1	4	4.html
i	369	1877	118	3	5	5.html
push	2	1879	124	1	6	6.html
zzzurl	1686	3565	168	1	7	7.html
new	450	4015	234	1	8	8.html
date	1600	5615	237	1	9	9.html

Table 2: First 10 lines of each output file

```
101.html:   by name, in a manner commensurate with influence on the final
1743.html:operating time (with a commensurate decrease in costs or operating
```

Figure 2: Tracing one query

3.3 Testing

Sample Output First 10 lines of each output files are shown in table 2.

Tracing one query Tracing query "commensurate"

1. In dictionary file: commensurate 2 233863
2. In posting file: go to line 233863 and 233864, which are: 96 1 and 1677 1
3. In mapping file: go to docid 96 and 1677, which are 101.html and 1743.html
4. Run command: "grep commensurate *" in input dir, the output is shown in figure 2.