Name: Sohag Desai
Homework: 4
Course: W251

==========

Credentials to access GPFS cluster:

==========

```
root@0a08edd7d492:~# slcli vs list
:.........:...........:..............:..............:
:    id   :  hostname  :   primary_ip   :   backend_ip  :
:.........:...........:..............:..............:
: 55254785 :   gpfs1    : 192.155.208.60 :  10.88.9.207  :
: 55254797 :   gpfs2    : 169.54.144.214 : 10.122.121.68 :
: 55254805 :   gpfs3    : 169.54.144.220 : 10.122.121.69 :
:.........:...........:..............:..............:


root@0a08edd7d492:~# slcli vs credentials 55254785
:.........:.........:
: username : password :
:.........:.........:
:   root   : K448xq7v :
:.........:.........:
root@0a08edd7d492:~# slcli vs credentials 55254797
:.........:.........:
: username : password :
:.........:.........:
:   root   : FanX6F52 :
:.........:.........:
root@0a08edd7d492:~# slcli vs credentials 55254805
:.........:.........:
: username : password :
:.........:.........:
:   root   : G8KwZw2P :
:.........:.........:
```

==========

Information on how to run mumbler:

==========

1. Navigate to `/gpfs/gpfsfpo/src` on any node (`gpfs1`, `gpfs2`, `gpf3`).

2. Run `./mumbler.py <starting word> <max number of words>`

==========

Implementation Overview:

==========

Directory listing:

```
[root@gpfs1 src]# ls
constants.py   dump_pickle_file.py     pickle_to_json.py
remote_ssh.py  wget_files.sh.          mumbler.py
preprocess_zipfiles.py                 unzip_files.sh
```

Files and their functions:

`wget_files.sh`
Script to fetch zip files from URL based on node.

`preprocess_zipfiles.py`
Python program that does the following:

1. Reads each zip file in the directory associated with each node.
   - For example, on `gpfs1` it processes files in `/gpfs/gpfsfpo/gpfs1`, on `gpfs2` it processes files in `/gpfs/gpfsfpo/gpfs2`, etc.

2. Creates index files based on first letter of first word of bigram.
   - For example, if it encounters a bigram (“hello” “goodbye”) when run on node `gpfs1`, it will create an index file /`gpfs/gpfsfpo/gpfs1/indexfiles/h.pkl`

3. Adds a dictionary for each bigram with keys `first`, `second` and `count` and values the first element of the bigram (lowercase), second element of the bigram (lowercase) and total count across each year that bigram is found in the corpus.

   - For example, if It encounters entries like:

     ABNORMAL CHANGES 1884 3 3 2
     ABNORMAL CHANGES 1885 1 3 2
     ABNORMAL CHANGES 1886 5 3 2

     it will collapse those it a single dictionary, lower case and serialize it to disk using `cPickle` as below:

     ```
     {“first”: “abnormal”, “second”: “changes”, “count”: 9}
     ```

     At the end of the preprocessing, the program running from each node will have created an `indexfiles` subdirectory within each directory associated with that node e.g.

node1 will have created /gpfs/gpfsfpo/gpfs1/indexfiles/
node2 will have created /gpfs/gpfsfpo/gpfs2/indexfiles/
node3 will have created /gpfs/gpfsfpo/gpfs3/indexfiles/

Each of these directories will contain index files named for each letter of the alphabet, i.e. a.pkl. b.pkl, ... z.pkl. These files will contain dictionary entries for bigrams collected from the files on that node.

mumbler.py

Python program that can be run on any node. It uses the index files generated by preprocessing to generate a series of words based on an initial word and following the second words in a bigram till the max words parameter is exceeded.

Here are its features:

- can be run on any node

- does a remote call to the same program on the next node in series if a word is not found on the node it is initially invoked from. The next node In series is defined from the sequence gpfs1->gpfs2->gpfs3->gpfs1

- picks the next word based on the probability of the second word in the bigram.
  * For example, if the following were all the bigrams starting with “abnormal”:

```
Abnormal Results    2006 12   10   8
Abnormal Results    2007 4    4    4
Abnormal Results    2008 16   16   5
Abnormal barium     1965 1    1    1
Abnormal barium     1972 1    1    1
Abnormal barium     1974 1    1    1
Abnormal barium     1982 1    1    1
```

the probability of selecting “results” as the next word will be

```
(12+4+16)/(12+4+16+1+1+1+1) = 32/36 = 8/9 = 88.89%
```

whereas the probability of selecting “barium” would be

```
(1+1+1+1)/(12+4+16+1+1+1+1) =  4/36 = 1/9 = 11.11%
```

Here are some caveats:

- It discards all “words” that have any non alphabetical characters
- It is case insensitive - it normalizes all words to lower case
- if the next word in the bigram is on the previous node in series it will first make the remote call to the next and then cycle back to the first node

```
constants.py
```
Defines all the constants used by the main programs.

Other files are utilities.

==========

Example runs:

==========

```
[root@gpfs1 src]# ./mumbler.py alice 10
Next word: alice
Next word: feigned
Next word: attacks
Next word: on
Next word: his
Next word: official
Next word: development
Next word: of
Next word: immobility
Next word: and
[root@gpfs1 src]# ./mumbler.py has 10
Next word: has
Next word: appropriated
Next word: to
Next word: serve
Next word: as
Next word: funds
Next word: in
Next word: little
Next word: kids
Next word: on
[root@gpfs1 src]# ./mumbler.py has 10
Next word: has
Next word: not
Next word: asked
Next word: suddenly
Next word: warm
Next word: supporters
Next word: sought
Next word: new
Next word: position
Next word: next
[root@gpfs1 src]# ./mumbler.py alice 10
Next word: alice
Next word: bullock
Next word: committee
```

```
Next word: for
Next word: the
Next word: mandingos
Next word: and
Next word: fluid
Next word: particle
Next word: are
[root@gpfs1 src]# ./mumbler.py alice 10
Next word: alice
Next word: gave
Next word: out
Next word: of
Next word: a
Next word: short
Next word: history
Next word: of
Next word: a
Next word: constant
[root@gpfs1 src]# ./mumbler.py alice 10
Next word: alice
Next word: in
Next word: some
Next word: very
Next word: broad
Next word: territory
Next word: and
Next word: navy
Next word: could
Next word: ever
[root@gpfs1 src]# ./mumbler.py hark 10
Next word: hark
Next word: back
Next word: and
Next word: pierre
Next word: angulaire
Next word: et
Next word: al
Next word: jolson
Next word: was
Next word: prophetic
[root@gpfs1 src]# ./mumbler.py hello 10
Next word: hello
Next word: to
Next word: a
Next word: thousand
Next word: he
Next word: would
```

```
Next word: tend
Next word: not
Next word: fidgety
Next word: nature
[root@gpfs1 src]# ./mumbler.py quinquireme 10
Next word: quinquireme
Next word: and
Next word: comparison
Next word: between
Next word: a
Next word: related
Next word: literature
Next word: that
Next word: this
Next word: medium
[root@gpfs1 src]# ./mumbler.py carpal 10
Next word: carpal
Next word: tunnel
Next word: project
Next word: are
Next word: both
Next word: the
Next word: ringleader
Next word: in
Next word: the
Next word: new
```