MATLAB Programming NYCU Fall 2022 LAB#6 2022/10/25

For this lab, you will try some string processing in MATLAB, as well as a bit of file I/O. You should write a function that takes a path to a text file as its only input. The function should find the individual words in the string, and determine the unique words and count their times of occurrence. For this lab, you can use any string processing function provided by MATLAB.

- You function header should look like

  ```
  [word, count] = my_word_count(fn)
  ```

  Here `fn` is the path to the file. There are two output arguments: `word` is a cell array of character vectors containing the unique words, and `count` is a vector containing their counts of occurrence.

  First, use a text editor to save the following text (which is an English tongue-twister) to a text file, and use its path as the input to your function:

  ```
  She sells seashells by the seashore.
  The shells she sells are surely seashells.
  So if she sells shells on the seashore,
  I am sure she sells seashore shells.
  ```

- Within your function, you need to open the file first. There are several different ways to retrieve the words from the file. The most convenient approach here is to use `textscan`:

  ```
  s = textscan(fid, '%s' ,'delimiter', '., ');
  A = s{1};
  ```

  Here `fid` is the file ID returned by `fopen`. Remember to close the file after this.

  The function `textscan` returns a cell array where each element corresponds to one item in the format specification. That is why we have to take just its first element, which is itself a cell array that contains all the individual words. The use of the `delimiter` attribute is a convenient way to directly skip the white spaces and punctuations when reading the file.

- The next step is to convert all the words to lower case using the function `lower`. While you can use a loop to convert the individual words, you can actually apply it to the whole cell array of character vectors in one call. (Tip: Many MATLAB string processing functions can be directly applied to all the elements in a cell array of character vectors or a string array in a single call. This reduces the need to use loops and leads to more compact and efficient programs.)

- You can apply `unique` to a cell array of character vectors or a string array. Use this technique to get the unique words. The result should be assigned to `word`. Note: The output will be sorted in dictionary order.

- Make a loop over the elements in `word` to count their occurrences. For each word in `word`, use `strcmp` to compare it with `A`, the list of all the words (with duplicates) in the original text. The output of `strcmp` here is a logical vector indicating the matches in `A` with the particular word. Apply `sum` to this vector to get the number of matches. (Tip: This is a convenient way to count the "**true**" elements in a logical array.)

- Next, sort `word` and `count` in the descending order of `count`, so that the more frequent words will appear first. (Note: sort `count`, and use the indices returned by `sort` to rearrange the element ordering of `word`.)

- Finally, if your function is called with no output argument (`nargout==0`), let the function print out the words and their counts. Let the output be like the following:

  ```
  sells       4
  she         4
  seashore    3
  ```

```
shells      3
the         3
seashells   2
am          1
are         1
by          1
i           1
if          1
on          1
so          1
sure        1
surely      1
```