Bearach Byrne

C15379616@mytudublin.ie

OOP Assignment 2 – Explaining Document

21/02/2021

Intro

This assignment introduced basic document retrieval methods using a supplied text file "ap_docs.txt". The 3 basic functions of the program are to:

1. **Allow user to search by keyword(s)** – This will return document numbers for any articles that contain their search term.
   ' july ' Was found in articles: {1, 2, 99, 36, 5, 132, 162, 41, 43, 44, 143, 144, 145, 179, 21, 87}
   In the case of multiple words, the words are all searched for individually and they will only show as a hit if all the words show up in the same article. So for example if the user searches for "find the", both words are searched for individually and then compared against each other to see if they both appear in any documents.
   ' find ' was found in articles: {129, 223, 133, 6, 199, 74, 140, 77, 110, 143, 175, 115, 88, 57, 221, 127}
   ' the ' was found in articles: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226}

   Your words were found in the following articles: {129, 223, 133, 6, 199, 74, 140, 77, 110, 143, 175, 115, 88, 57, 221, 127}

2. **Allow users to print specific documents** – This will allow the user to enter a document number (from 1-226) and it will print to the run window the article that they have selected.
3. **Exit** – This will allow the user to exit. If they do not exit, the program will keep prompting them to enter another search/print an article.

This program was completed using 5 custom built functions. Each one is detailed below.

The driver code for this project simply calls on the main_menu function and the program takes care of the rest.

```
main_menu()
```

main_menu

This function is the one that ties together all the code written for this assignment. It takes no input argument and returns nothing (only prints to screen for the user). There is quite a lot of print statements printing dashes and new lines in this function that don't impact the functionality of the function/program but were included to make it easier for the user to read the results.

This function prints some introductory messages with some time delays between them as I think it looks nicer to see the lines of text print sequentially. It then presents the users with their 3 options (1 to search by keyword, 2 to print specific article or 3 to quit). The user is then asked to input a number 1, 2 or 3 to make their selection. The function checks which value they entered, if they enter an invalid number or an invalid character, they are given a message explaining this. This was done using a try/except block for ValueError to check for a string/float/Bool input, checking for an invalid number was done using an else clause on the section that checks the user input.

If the user enters a valid input (1, 2 or 3), the relevant function is called (print_article or search).

create_list()

This function reads in the text file (ap_docs.txt) and populates a list with each document taking up a single list item (this is needed for the next function print_article). It takes no input arguments and returns a list "paragraph_list".

The line separating each document "<NEW DOCUMENT>" was used with the split function to divide the text file into the correct documents. This however created a blank element in the list in the first position. This was because the first line in the text file was "<NEW DOCUMENT>", so it split it based on that being the separator, but because there was no text in front of the <NEW DOCUMENT>, the element was null. This element was simply removed using:

```
paragraph_list.remove(paragraph_list[0])
```

print_article

This function will take care of the 2nd required user function (allowing them to print a specific article). It takes 1 input argument, which is the list created in create_list(), and it prints out whatever article number the user asks for by printing the list index for the document in question (this is done by subtracting 1 from whatever the user enters). This function checks for incorrect inputs using a try/except block and checks for IndexError for when the user enters a number outside of 1-226, and a ValueError for if the user enters a string that cannot be converted to an integer.

create_dict

This function enables the primary functionality of searching documents by keyword(s), as it creates the dictionary that will be searched. It takes no input and handles the text file and then stores each word in a dictionary as a key, with the values for each key being the articles that the word appears in.

Originally, I had tried to use the list of articles created in the create_list function as an input for this function and create the dictionary from there, however I could not figure out how to properly add the values for the keys. Every time I added a new value it would overwrite the previous values. So for instance if the word “test” was added, if it was in articles 1,2,3; the only value to be saved would be the final one ({“test”: {3}}). I spent a lot of time trying to fix this however I couldn’t come up with a solution that worked, so as a result I decided to ignore the already created list and read in the file again and create the dictionary from there.

This function iterates through the lines in the file and strips the punctuation and line breaks, then converts them to lower case. It then adds 1 to the document number counter if it comes to a “<NEW DOCUMENT>” line, otherwise it will iterate through the words in the line and add them to the dictionary if they are not already in it, and it will then add the document number as a value. On the other hand, if the word is already in the dictionary as a key it will only add the document number to a set (this is done to avoid duplicates in the case of a word showing in a document multiple times). This set is then added as the values for that key.

search

This function adds the primary function of the entire assignment, allowing the user to search for articles by keyword(s). It the entire function is in a while loop so that it will keep the user here searching for keywords until they choose to exit. It takes the dictionary created in the previous function as an input and returns nothing (it simply prints to the console for the user). The function takes in the user input of search terms (i.e. a string), then treats it by removing punctuation and converting to lower case.

It will then print out the number of search terms the user has specified (“stock market” will show as 2 terms).

If the user has only specified 1 term if will run a simple check to see if the word is in the dictionary. If it is it will then print the document numbers for that word. If not, it will tell the user that there is no match.

If the user has entered more than 1 search term, then the function will iterate through their search words one at a time, this was done by breaking their input into a list with each word taking up a list element. The function then checks if each word is present and if so it will print the relevant document numbers and store them in a list with each set of document numbers being an element of the overall list.

```
(e.g. word_match = [[1,6,8,10],[2,6,10]] )
```

If not, it will print a message saying it was not found. This is done for each word in the search term. Once the words have been searched for individually, the list is converted to a set and the intersection of the elements is found. This intersection will show the articles where all of the words occur. This is then printed for the user (I left the curly brackets on as I think it made it more legible).