

**Faculty of Engineering & Technology**

**Electrical & Computer Engineering Department**

**ENCS 313-Linux Laboratory**

**Project One - Shell Scripting**

**[Text Summarization using Sentence Centrality]**

**Prepared by:**

**Yasmin Jwabreh-1180815**

**Hedaya Mustafa-1182126**

**Instructor: Dr. Muhammad Jubran.**

**Teaching Assistant: Eng. Shahd Muhareb.**

**Section: 3**

**Date: 20-12-2020**

# **Abstract**

The aim of this project is to be more familiar with shell scripting programming by producing a text summarization using sentence centrality.

Contents

[Abstract 2](#_Toc59394038)

[Introduction 4](#_Toc59394039)

[Text Summarization using Sentence centrality: 4](#_Toc59394040)

[Procedure & Discussion 5](#_Toc59394041)

[ ReadFile function: 5](#_Toc59394042)

[ Splitpunc function: 7](#_Toc59394043)

[ ToLower function 8](#_Toc59394044)

[ RemoveStopWords function: 8](#_Toc59394045)

[ Centrality Function Description: 10](#_Toc59394046)

[Testing Examples: 14](#_Toc59394047)

[1. Test Case 14](#_Toc59394048)

[2. Test Case 17](#_Toc59394049)

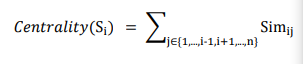
[Conclusion 19](#_Toc59394050)

[References 20](#_Toc59394051)

# **Introduction**

## **Text Summarization using Sentence centrality:**

Extractive summarization works by choosing a subset of sentences from the original document that contains the main contents. Several techniques presented in the literature to handle extractive text summarization. Centrality concept is one of the most used technique. In this approach the document is tokenized into sentences based on (. ! ? ) punctuation marks, then the similarity of each pair of sentences is computed. Finally, top scored sentences are selected based on the summary ratio. Formally, let D denote a document consisting of a sequence of sentences {s1, s2, ..., sn}, and Simij is the similarity score for each pair (si, sj). The degree centrality for sentence si can be defined as:

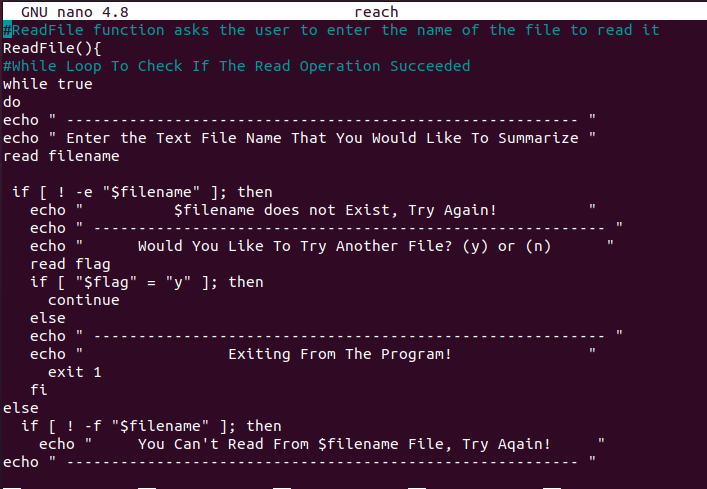


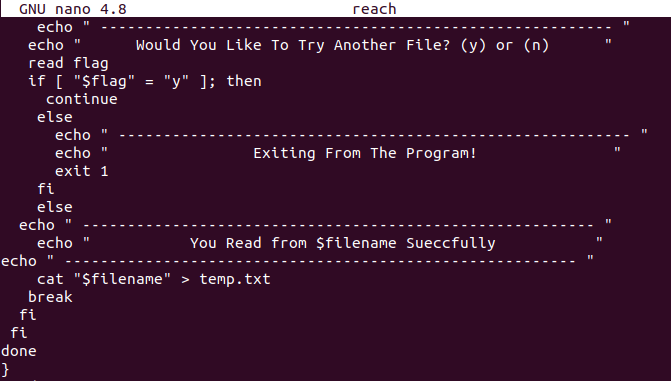
After obtaining the centrality score for each sentence, sentences are sorted in reverse order and the top ranked ones are included in the summary.

# **Procedure & Discussion**

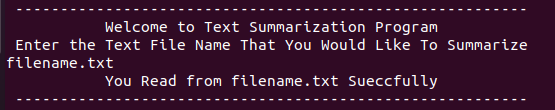
## **ReadFile function:**

This function will ask the user to enter the name of the file to read it, then checks if the operation succeeded or not. If the user enters a wrong file name it will ask him to choose between either continuing the program or exiting from it. When the user enters a valid name then it will read it and put it in temp.txt file.

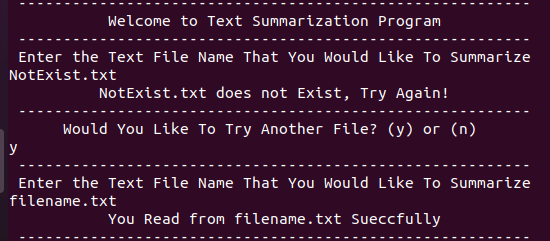




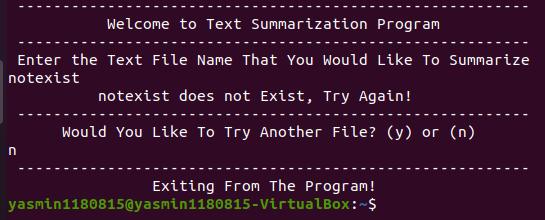
* **Testing ReadFile Function:**



* **Trying to read not existing file:**

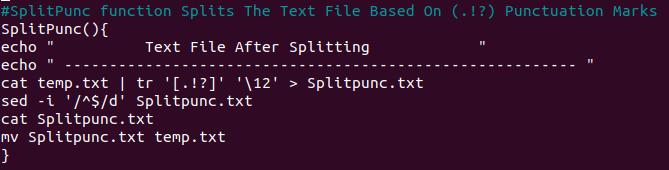
****

* **Trying to read unreadable file:**

****

## **Splitpunc function:**

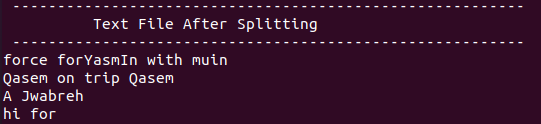
Sentence tokenization based on (. ! ?) punctuation marks.



* **Testing Splitpunc function on this data:**

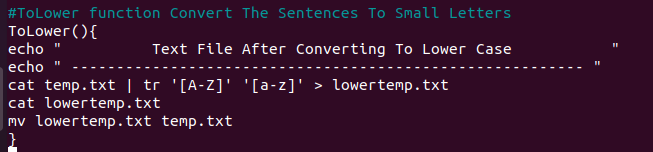
C:\Users\HP\Downloads\132025569_387257085715970_7665974981660597333_n.png

* **Output of Splitpunc function:**

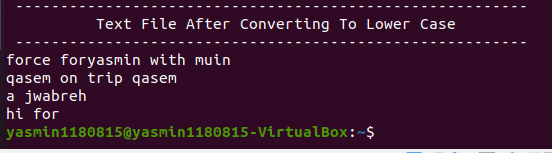


* **ToLower function:**

This function will convert the sentences into small letters.

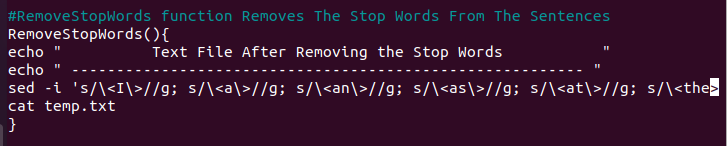
****

* **ToLower function output:**

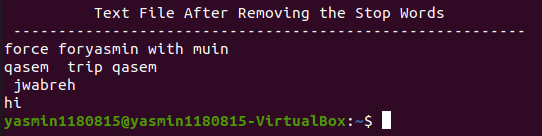


* **RemoveStopWords function:**

In this function, words which do not contain important information are removed from both sentences.

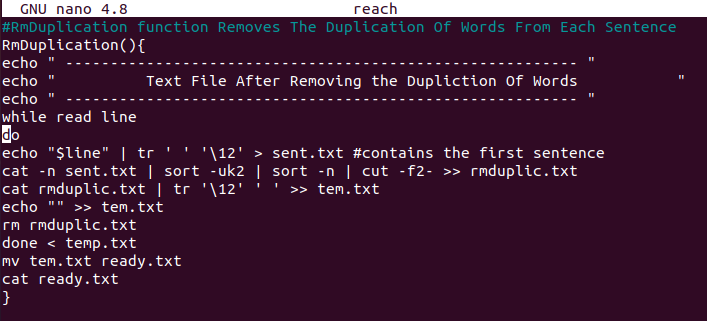


* **RemoveStopWords function output:**

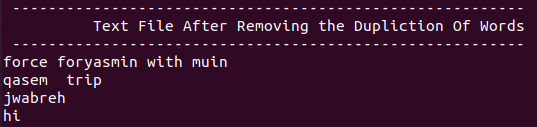


* **RemoveDuplication function:**

In this function, the duplication of words removed from both sentences. In other words, each word appears once per sentence.



* **RemoveDuplication function output:**



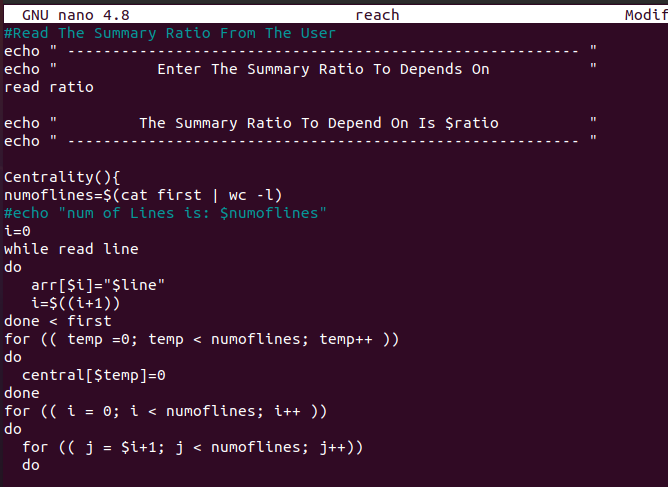
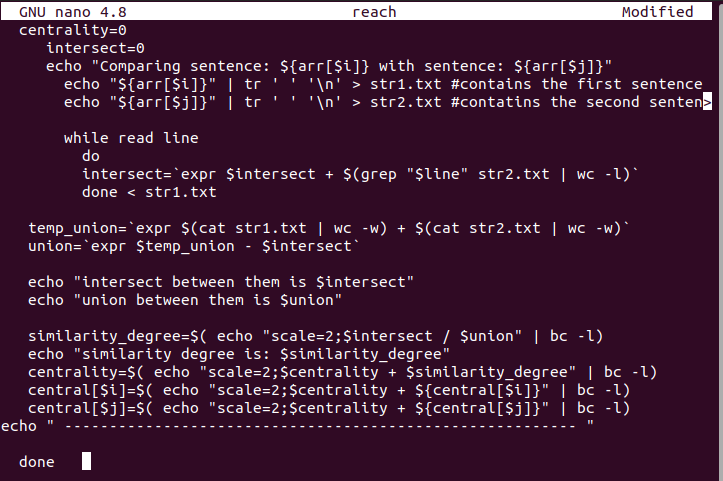
## **Centrality Function Description:**

In this Step, we Calculated the number of words that each two sentences intersect, and union, the method we used was moving each sentence of the two to compare to two text files, splitting each sentence with ‘ ‘ so we can have each word on a line, and then with nested looping we compared each sentence with the other, and counted the number of words where each sentence intersected with the other. After computing the intersect for each sentences with the other, we computed the union of them, that’s by considering that it’s the sum of the #of words for the both sentences minus the intersected words number. Finally, we got the similarity ratio for each pair of sentences in the file, saving the values to an array (central[])

Centrality of a sentence is the sum of the similarity ratio for that sentence with the other sentences in the file. To achieve the right answer for each, we counted on the central array with each comparison, For example, when we compare the first sentence with the second, the similarity for each will be added to their counters in the array. After finishing the comparison we’ll get an array with the centrality for each sentence.

We moved the array elements of the centrality result to a file then sorted it reversely that the first sentence is with the highest centrality, we found some challenges in this step, but we solved it by considering the array index as the sentences number, and so, we compared the centrality in reverse order with the original array with unsorted centralities, which that we compared the first centrality array element after sorting with the elements before sorting, by getting the answer index it’ll refer to the needed sentence index.

**Centrality Function**



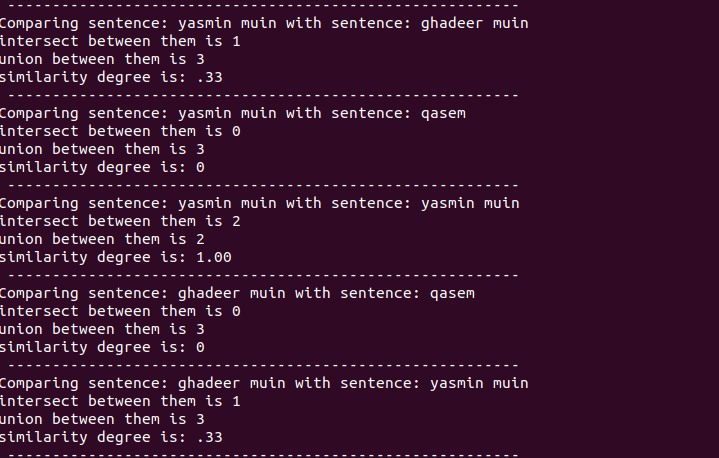


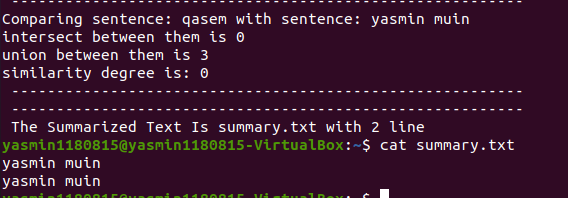
* **Input file to test Centrality function:**

In this file, we entered helpful data to make sure that the centrality computed correctly.

لا يتوفر وصف.

* **Output file:**



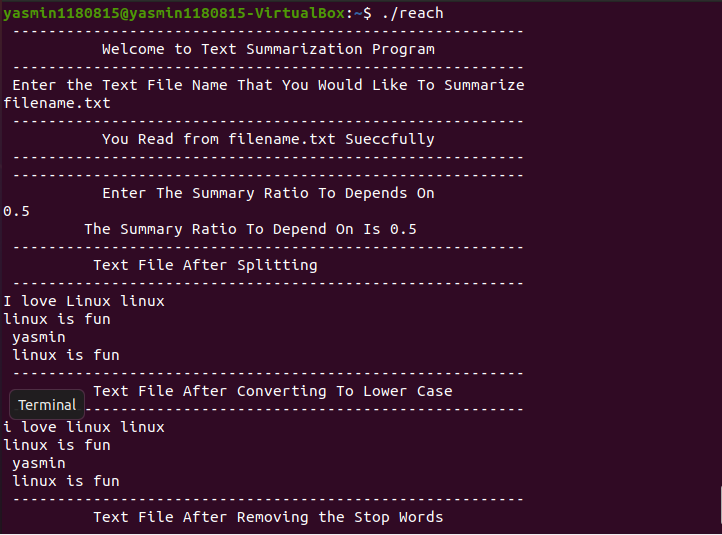


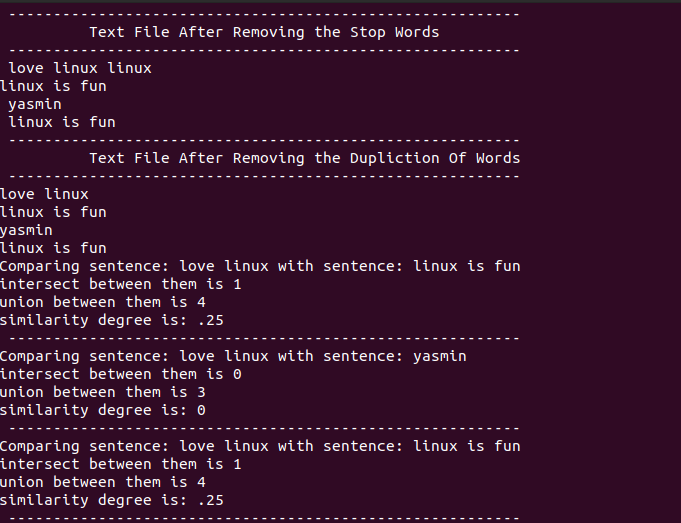
# **Testing Examples:**

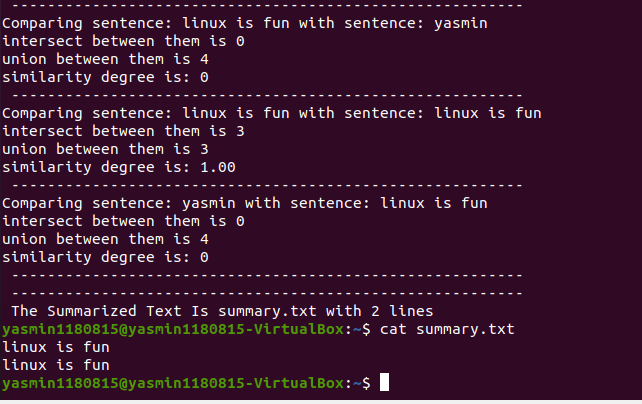
## **Test Case**

* Input file:

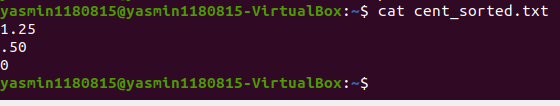


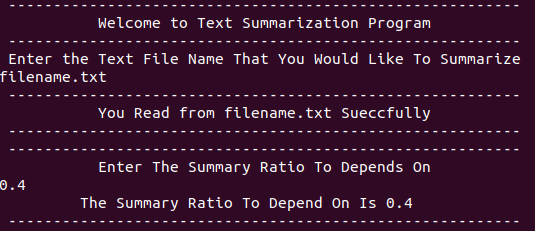
* Running the Program:

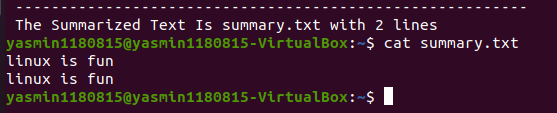




The Centralities For Each Sentence Sorted In Reverse Order

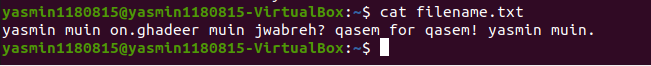


In addition, if we entered 0.4 as summary ratio, the summarized lines answer will be 1.6, in this case, 2 lines will be printed. (Rounded to the biggest number) 



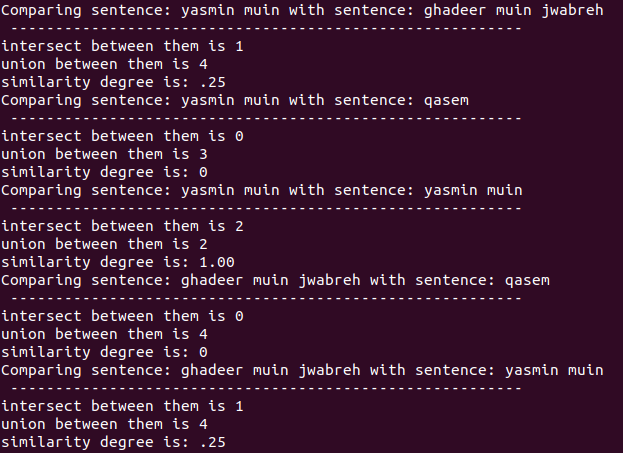
## **Test Case**

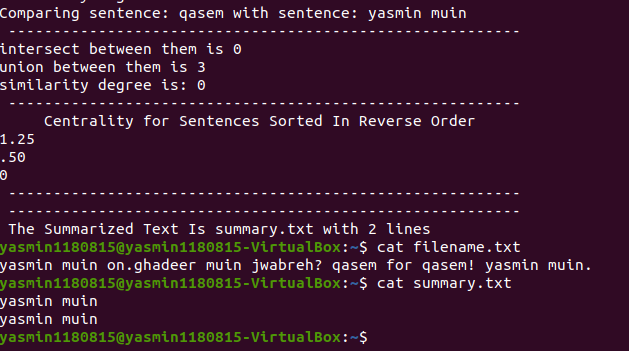
* Input file:



* Running the Program:







# **Conclusion**

In this project we dealt with shell scripting, now we are able to summarize texts by computing the centrality for each sentence.

The objectives of this project were successfully achieved. We have learned and practiced how to produce a text summarization using shell programming.

# **References**

1. <https://www.educba.com/shell-scripting-commands/>

Accessed on 12-12-2020 at 3:15 AM.

1. <https://www.tutorialspoint.com/unix/shell_scripting.htm>

Accessed on 15-12-2020 at 10:45 PM.

1. <https://www.stackoverflow/unix/shell_scripting.htm>

Accessed on 18-12-2020 at 2:45 PM.