**Big Data Programing midterm report**

step o: initialize spark and nltk environment

After initialization of spark and nltk environment, I create RDDs of a text file by using spark Context and by passing the file path to text File ()

Step1: get the word count of the file

After I create the RDDs, I map the RDDs by using a lambda function that creates RDDs for every line then I use flatMap by using lambda function that to creates RDDs for every word. Then I get the word count by using .count() method

Step 2: get the character count

After I create the RDDs, I map the RDDs by using a lambda function that creates RDDs for every line then I use flatMap by using lambda function that to creates RDDs for every character. Then I get the character count by using .count() method

Step 3: Compare the Expected Frequency of English letter to the frequently occurrence letter in the text

To do this, I first find the frequently occurrence letter in the text. First, I map the character RDDs giving each character a value of one and then I reduce them by Key. Then I map the characters by diving each character occurrence to the total character occurrence of the text. Then I filter The characters only to be English letters by using .isalpha() method. Then I sort the created letters based of their total occurrence.

After I sort the letters, I take first element in the RDDs (Since it’s sorted RDD, The frequently occurs letter has to be at the top). I use .first() to return the first RDD as a tuple. In the tuple, the letter found at the first index so I use letter[0] also I assign the letter as string literal.

Then I list English alphabets as stri

Then I create an Array of Most Frequently occur letter in English based a Wikipedia source.

I use compare function that takes the letter(the highest occur letter in the text) and Index of frequency of occurrence of English letters. Then it will return the difference as value.

Step 4: shift the letters by Value

In the last step, I get a value. Now I use shift function that shift every letter in the text by that value. then it will return a new text file. I create a new empty Array data then I append every letter(English alphabets) to the Array list.

Step5: I create a function called gsamp().

That creates a sample of words from the file. In order to create the sample of word, I use a parallelize RDDs that has a partitions of 10 then I filter the RDD to be only letters by using .isalpha method. Then I create an empty arrayOfSamples then I store the samples of word in it.

Step 6: I create tester function

tester function will use wordnet form ntlk.corpus library then it will check weather the sample of words are English words or not. Every time the sample word in an English word, it will increase the tally by one. Finally, it will return the total number of tally

Step 7: Checker function

checker function will take the total amount of tally form the tester function then will make a conditional decision. Like if the total amount of tally greater than 5 then words are English word therefore, the decryption is correct then it save it as text file by asking users the desirable file name.

else If tally is less than 5, the function will recursively call function starting from step 1 to 7. To do this, First it will declare i as global variable and increase the value of i by one then recursively call funcions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name of the file | New (Decrypted) file name | Frequently occur letter in the Encrypted file | source | Decryption of key |
| Encrypted-1 | Decrypted-1 | R | https://studfile.net/preview/393909/page:78/ | 13 |
| Encrypted-2 | Decrypted-2 | X | https://books.google.com/books?id=7x9yw5DQIOUC&pg=PA92&lpg=PA92&dq=times+of+peace,+are+of+every-day+occurrence+between+contending+armies&source=bl&ots=DkxSINdPlO&sig=ACfU3U2CWQf9gBcVwRpBMFrSjgq3QEDIHg&hl=en&sa=X&ved=2ahUKEwiFkbK53-7nAhXySt8KHSXZBa4Q6AEwAHoECAQQAQ#v=onepage&q=times%20of%20peace%2C%20are%20of%20every-day%20occurrence%20between%20contending%20armies&f=falseEncrypted-3 - https://www.imsdb.com/scripts/Great-Gatsby,-The.html | 7 |
| Encrypted-3 | Decrypted-3 | J | https://www.imsdb.com/scripts/Great-Gatsby,-The.html |  |

Decrypted-1 output:

A screenshot of a social media post

Description automatically generated

Decrypted -2 output:

A screenshot of a cell phone

Description automatically generated

Decrypted-3 output:

A screenshot of a cell phone

Description automatically generated

Github Link:

<https://github.com/samson369/Big-Data-programming-/tree/master>