CLI For Big Data Analysis

SHYAM MOHAN K.
shyam.mk@outlook.com

1. Introduction

Bash is a Unix command processor that allows a user to type in commands in a text window, that performs certain defined actions. It can read and execute commands from a file, called a shell script. Like all other Unix shells, it supports wildcard matching, piping, document processing, command substitution, variables, and control structures for condition-testing and iteration.

The objective of this project to understand how Bash (CLI) is used for big data analysis. We explore the different bash commands and shell scripts used extensively in file/text processing. This report explains each use case in scope, along with its corresponding command and output.

2. CLI for Big Data Analysis

2.1. Overview

We use a variety of commands like *grep, sed, awk, tr, sort, uniq* and *wc*, to find the outputs for the given use cases. These commands would be executed through a master shell script *"BashExecutionPackage.sh"* which is developed as an interactive Unix application. Through this, the user can choose which use-case to execute and the corresponding output would be displayed on the console. Certain property variables used in this script are defined inside the file *"LoadBashProperties.sh"* and are loaded at the beginning of the execution.

2.2. Commands & Explanations

Command Parameters	Description
sed /^[[:space:]]*$/d	To remove all the blank/empty lines in the files.
sed s/[[:space:]]*$//	To replace all the trailing whitespaces in the files, with blank.
sed s/--/ /g;s/[—‘’“”•´-]//g	To remove the special characters appearing in the input files (in scope).
sed s/^\xEF\xBB\xBF//g	To remove all the BOM characters appearing in the input files (in scope).
${INPUT_COMMAND_PARAM_STRING}	../../inputFiles/*.txt
tr -s '[:punct:][:blank:]\r\n' '\n'	To replace all the punctuations, horizontal whitespaces and \r\n (DOS character) with a new line. This ensures that each word is in a separate line.
tr '[:upper:]' '[:lower:]'	To convert all uppercase letters to lowercase (normalizing the file contents).
sort -f	To sort the words ignoring the case differences (if any exist).
uniq -ci	To get the unique words (uniq) and their counts (-c), ignoring any case differences (-i) that may exist.
wc -l	To count the no. of lines in the output file.

For all the use-cases mentioned, the first step is to clean the file by removing all the whitespace characters, all special characters and punctuations from the input files in scope. Then, normalize the text by converting all the words to lower case. Hence, the above-mentioned commands are common to all the commands corresponding to the use cases.

[1]. (a). Total No. of Distinct Words

Commands:

- sed '/^[[:space:]]*$/d;s/[[:space:]]*$//;s/--/ /g;s/[—“”“•´-]//g;s/^\xEF\xBB\xBF//g' ${INPUT_COMMAND_PARAM_STRING} | tr -s '[:punct:][:blank:]\r\n' '\n'| tr '[:upper:]' '[:lower:]' | sort -f | uniq -ci > $Q1_A_OUTPUT_FILE_NAME
- cat $Q1_A_OUTPUT_FILE_NAME | wc -l

  where **$Q1_A_OUTPUT_FILE_NAME=' ../outputFiles/Q1_A_UniqueWords.txt'**

- The first step is text pre-processing using the commands in the table, and then the set of unique words and their counts are written to the output file *"Q1_A_UniqueWords.txt"*. Since the expected output is the number (count) of the unique words, the final output would be the count of lines in the output file.
- Note that the generated output file is only for reference. The count of unique words can be obtained by applying the command *wc -l* to the output of the first command instead of transferring it to an output file.

[1]. (b). No. of Words Starting with the letter Z/z

Commands:

- sed '/^[[:space:]]*$/d;s/[[:space:]]*$//;s/--/ /g;s/[—“”“•´-]//g;s/^\xEF\xBB\xBF//g' ${INPUT_COMMAND_PARAM_STRING} | tr -s '[:punct:][:blank:]\r\n' '\n'| tr '[:upper:]' '[:lower:]' | sort -f | uniq | grep "^[z]" > $Q1_B_OUTPUT_FILE_NAME
- cat $Q1_B_OUTPUT_FILE_NAME | wc -l

  where **$** Q1_B_OUTPUT_FILE_NAME**=' ../outputFiles/Q1_B_Z_Words.txt'**

- Once the text pre-processing is completed using the commands mentioned in the table above, the set of words starting with 'z' are written to the output file *"Q1_B_Z_Words.txt"*. As per the question, the expected output is the number (count) of the words that start with z/Z. This can be obtained by counting the no. of lines in the output file, which is done by the second command.
- We use ***grep "^[z]",*** in the first command to find the set of words starting with 'z'.
- Note that the generated output file is only for reference. The count of the words that start with z/Z can be obtained using the first command itself by applying *wc -l* to the output of the first command instead of transferring it to an output file.
- Furthermore, we need not try to find the words that start with 'Z', because we have already converted all the uppercase words to lower case to normalize the text.

[1]. (c). No. of Words Appearing Less Than 4 Times

Commands:

- sed '/^[[:space:]]*$/d;s/[[:space:]]*$//;s/--/ /g;s/[—“”“•´-]//g;s/^\xEF\xBB\xBF//g' ${INPUT_COMMAND_PARAM_STRING} | tr -s '[:punct:][:blank:]\r\n' '\n'| tr '[:upper:]' '[:lower:]' | sort -f | uniq -ci | awk '{if ($1 < 4) printf "%s: %s\n",$2,$1}' > $Q1_C_OUTPUT_FILE_NAME
- cat $Q1_C_OUTPUT_FILE_NAME | wc -l

  where **$** Q1_C_OUTPUT_FILE_NAME**=' ../outputFiles/Q1_C_Words_LessThan4Times.txt'**

- Once the text pre-processing is completed, all the words appearing less than 4 times are written to the output file *"Q1_C_Words_LessThan4Times.txt"*. As per the question, the expected output is the number (count) of the words that appear less than 4 times. This can be obtained by counting the no. of lines in the output file, which is done by the second command.
- The command ***awk '{if ($1 < 4) printf "%s: %s\n", $2, $1}'*** finds the final output in this case. From the unique counts of words obtained (using ***'uniq -ci'*** in the first command), each word's count is checked to find if it is less than 4 and if true, then the word and its count is formatted and written to the output file.
- Note that the generated output file is only for reference. The final output can also be obtained directly by applying the command *wc -l* to the output of the first command instead of transferring it to an output file.

[2]. Most Frequent Words that End with '-ing':

Commands:

- sed '/^[[:space:]]*$/d;s/[[:space:]]*$//;s/--/ /g;s/[—‘’“”•´-]//g;s/^\xEF\xBB\xBF//g' ${INPUT_COMMAND_PARAM_STRING} | tr -s '[:punct:][:blank:]\r\n' '\n'| tr '[:upper:]' '[:lower:]' | sort -f | uniq -ci | grep "ing$" | sort -nr | awk '{ printf "%s: %s\n",$2,$1 }' > $Q2_OUTPUT_FILE_NAME
- cat $Q2_OUTPUT_FILE_NAME | wc -l

  where **$** Q2_OUTPUT_FILE_NAME**=' ../outputFiles/ Q2_Words_EndingWith_ing.txt**

- Once the text pre-processing is completed, all the words appearing less than 4 times are written to the output file *"Q2_Words_EndingWith_ing.txt"*. As per the question, the output should be the most frequent words that end with '-ing'. Considering the top 10 such words as the final output, the second command can be used to display the same.
- The extra commands used in this use-case ***grep "ing$" | sort -nr | awk '{ printf "%s: %s\n",$2,$1 }'*** finds the set of words ending with '-ing' (using grep), sorts them in descending order based on their counts (using sort -nr) and then forms the final output in a formatted way (using awk). The identified words are then written into the output file.
- As mentioned in the other cases, the generated output file is only for reference. The final output can be obtained directly by using the command *head -10* along with the first command.

[3]. Most Frequent - me/my/mine/I or us/our/ours/we?

Commands:

- sed '/^[[:space:]]*$/d;s/[[:space:]]*$//;s/--/ /g;s/[—‘’“”•´-]//g;s/^\xEF\xBB\xBF//g' ${INPUT_COMMAND_PARAM_STRING} | tr -s '[:punct:][:blank:]\r\n' '\n'| tr '[:upper:]' '[:lower:]' | sort -f | uniq -ci | awk 'BEGIN{ sum1=0; sum2=0;} {if($2 ~ /^me$|^my$|^mine$|^I$/) sum1=sum1+$1; else if($2 ~ /^us$|^our$|^ours$|^we$/) sum2=sum2+$1} END{ if (sum1 > sum2) printf "%s: %d\n", "Group me/my/mine/I",sum1; else printf "%s: %d\n", "Group us/our/ours/we",sum2}'

- First step is text pre-processing using the commands given in the table above. The expected output is the most frequent group of words among "me/my/mine/I" and "us/our/ours/we". Considering the top 10 such words as the final output, the second command can be used to display the same.
- The command ***awk 'BEGIN{ sum1=0; sum2=0;} {if($2 ~ /^me$|^my$|^mine$|^I$/) sum1=sum1+$1; else if($2 ~ /^us$|^our$|^ours$|^we$/) sum2=sum2+$1} END{ if (sum1 > sum2) printf "%s: %d\n", "Group me/my/mine/I",sum1; else printf "%s: %d\n", "Group us/our/ours/we",sum2}'***, looks for the occurrences of the words **me/my/mine/I** and adds them up.

- Also, the occurrences of the words **us/our/ours/we** would be summed. Finally, the two sums would be compared to find the greater value. The group with greater occurrences would be displayed as output in a formatted way.

[4]. Five Words that appear the most after a specific Stop-word:

Commands:

- read stopWord
- stopWord=`echo ${stopWord} | tr '[:upper:]' '[:lower:]'`
- sed '/^[[:space:]]*$/d;s/[[:space:]]*$//;s/[—‘’“”•´-]//g;s/^\xEF\xBB\xBF//g' ${INPUT_COMMAND_PARAM_STRING} | tr -s '\r\n' '\n' | tr -s '[:punct:][:blank:]' ' ' | tr '[:upper:]' '[:lower:]' | awk '{for(i=1;i<NF;i++) if ($i=="'${stopWord}'") {i++;print $i;} }' | sort | uniq -ci | sort -nr > $Q4_OUTPUT_FILE_NAME
- cat $Q4_OUTPUT_FILE_NAME | head -5

- In this use case, the first step is to obtain a stop-word from the user. The user can enter any stop-word of his choice which would be read by the script and converted to lower-case. This is done by the first two commands.
- Next step is text pre-processing using the same commands mentioned in the table. Once the pre-processing and normalization is completed, we use the command ***awk '{for(i=1;i<NF;i++) if ($i=="'${stopWord}'") {i++;print $i;} }'*** to list all the words that appear after the given stop-word.
- Next, we sort the list, find the unique counts and sort the words based again based on their counts in descending order. The words and their respective counts are written to the output file *"Q4_FiveFrequentWords_AfterStopword.txt".*
- Just as in other use-cases, the generated output file is only for reference. The final output can be obtained selecting the top five words using the command *head -5* along with the third command.