Text File Analyzer Using Perl

Dhruv Shah
Department of Electronics and Communication Engineering,
Nirma University, Ahmedabad
21bec030@nirmauni.ac.in

Dhruv Gobbi
Department of Electronics and Communication Engineering,
Nirma University, Ahmedabad
21bec035@nirmauni.ac.in

Abstract— **This Perl-based text file analysis tool aims to provide comprehensive insight into text data in a variety of file formats.This tool performs various operations such as statistical analysis, keyword extraction, content modification and export functionality. The main features of this tool include word, line, character counting, keyword extraction, search and replace functionality, keyword frequency analysis and export of analysis results to a text file.**

Key Words— **Word, Line, Character Counting, Keyword Extraction, Search and Replace, Frequency Analysis.**

Introduction

Text File Analyzer is an intelligent script that makes assessments of any given text file. The Perl language was chosen because of its amazing text processing and manipulation capabilities.This tool can calculate word count, line count , top 5 keywords ,can perform search and replace function and exports this valuable information into a text file. Email feature was added using a bash shell script which readily converts the text file into pdf format and sends it to the specified email address easily and quickly. It is a useful tool for data analysts, content writers and editors. Useful in documenting and reporting of large documents.

Key Packages Utilized

1. perl:

This package was installed in order to execute perl language based scripts properly in the Linux Ubuntu environment. In simple terms, it is a Perl Interpreter which is the core component of Perl language that is responsible for interacting with Perl scripts and it also includes some basic libraries with Perl Documentation.

2. mailutils:

This package provides a set of various utilities for managing emails from the command line. It includes the 'mail' command which is used in the bash shell script for sending mail to a specific email id.

3. smtp (Simple Mail Transfer Protocol):

This package was installed to utilize its capabilities of sending emails efficiently from the command line. It follows a client-server model. The SMTP servers exchange acknowledgment messages to confirm successful email transmission.

Pseudo Code

1. Reading any given text file. For example: the name of the text file is test3.txt in this case.

2. SMTP Configuration: Set up SMTP server configuration including server address, port, username, and password.

3. Subroutine to analyze Text File (analyze_text):
Check if the file exists; if not, display an error and terminate.
It returns the following results: word count, character count, line count.

4. Subroutine to search and replace (search_and_replace) :
Calls analyze_text subroutine to get initial data. Performs a global search and replace on each and every line. Returns the original and modified contents upon replacement.

5. Subroutine to extract the top keywords (extract_keywords):
Again calls analyze_text subroutine to get initial data. Sorts word_count which is a hash data structure in perl used here by frequency. Extracts the top 5 keywords and returns them.

6. Subroutine to analyze and export to Text File (analyze_and_export_to_text): It calls for another subroutine export_results_to_text_file and stores the resulting text file name then displays a message for completed assessment and giving a confirmation of successful transfer of results into a text file called analysis_results.txt.

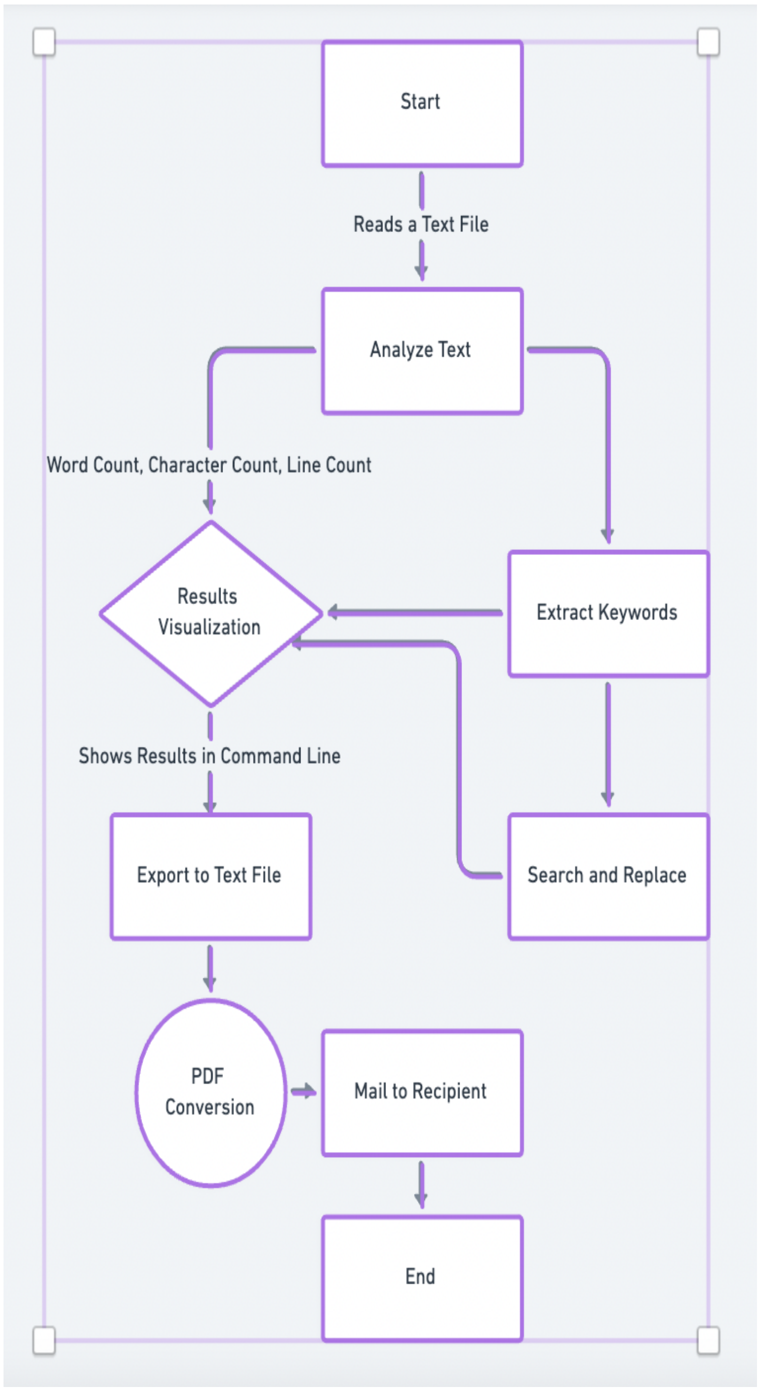
7. Subroutine to export results (export_results_to_text_file):
It calls analyze_text which gives word count, character count, line count and next calls search_and_replace to get the original and modified contents . It then obtains the top 5 keywords from the text file using extract_keywords subroutine. It writes all information gathered in a new text file called analysis_results.txt and then sends it back to analyze_and_export_to_text subroutine.

8. All the above subroutines are executed based on the Main subroutine wherein input file is taken from the user and based upon the contents present in the text file results are obtained and then those results are transferred to a text file as mentioned above.

9. Conversion of text file to pdf format using bash commands 'enscript' and 'ps2pdf' which are built in part of the Ubuntu environment. A separate bash shell script is created for this conversion and then sending emails to the specified email address.

10. Sending Email: Use SMTP to send mail to a user given email address with the analysis_results.pdf attachment created above using bash commands through 'mail' command with any subject necessary.

Flowchart



Commands Used In Linux To Import Libraries

1.) sudo apt-get install perl
2.) sudo apt-get install ssmtp
3.) sudo apt-get install mailutils
4.) sudo apt-get install mailx

Code

Code to analyze any given text file and to obtain the results in a new text file

```
#!/usr/bin/perl

use strict;
use warnings;

sub analyze_text {
   my ($file_name) = @_;

   unless (-e $file_name) {
      die "File '$file_name' does not exist.\n";
   }

   open my $fh, '<', $file_name or die "Could not open
'$file_name' for reading: $!\n";

   my %word_count;
   my $char_count = 0;
   my $line_count = 0;
   my $word_total = 0;  # Add word count variable
   my @lines;

   while (my $line = <$fh>) {
      $line_count++;
      $char_count += length($line);
      my @words = split(/\s+/, $line);
      $word_total += scalar @words;  # Increment word
count
      foreach my $word (@words) {
         $word_count{$word}++;
      }
      push @lines, $line;
   }

   close $fh;

   return (\%word_count, $char_count, $line_count,
$word_total, \@lines);  # Include word_total in the return
}
```

```
sub search_and_replace {
    my ($file_name, $search, $replace) = @_;

    my ($word_count, $char_count, $line_count, $word_total,
  $lines_ref) = analyze_text($file_name);
    my @lines = @$lines_ref;

    my $original_file_name = $file_name . '.original';
    my $modified_file_name = $file_name . '.modified';

    open my $original_fh, '>', $original_file_name or die
  "Could not create '$original_file_name' for writing: $!\n";
    open my $modified_fh, '>', $modified_file_name or die
  "Could not create '$modified_file_name' for writing: $!\n";

    foreach my $line (@lines) {
        my $modified_line = $line;
        $line =~ s/$search/$replace/g;  # Perform global search
  and replace
        print $original_fh $line;
        print $modified_fh $modified_line;
    }

    close $original_fh;
    close $modified_fh;

    my $original_contents = do {
        local $/;
        open my $fh, '<', $original_file_name;
        <$fh>;
    };

    my $modified_contents = do {
        local $/;
        open my $fh, '<', $modified_file_name;
        <$fh>;
    };

    return ($original_contents, $modified_contents);
  }

  sub extract_keywords {
    my ($file_name, $num_keywords) = @_;

    my ($word_count) = analyze_text($file_name);

    my @keywords = sort { $word_count->{$b} <=>
  $word_count->{$a} } keys %$word_count;

    return @keywords[0..$num_keywords - 1];
  }
```

```
sub analyze_and_export_to_text{
   my ($file_name, $search, $replace) = @_;

# Export results to a text file
   my $text_file_name =
export_results_to_text_file($file_name,$search, $replace);
   print "Analysis results exported to '$text_file_name'\n";
}

sub export_results_to_text_file {

   my ($file_name, $search, $replace) = @_;
   my $text_file_name = 'analysis_results.txt';

   my ($word_count, $char_count, $line_count, $word_total,
$lines_ref) = analyze_text($file_name);
   my ($modified_contents, $original_contents) =
search_and_replace($file_name, $search, $replace);
   my $num_keywords = 5;
   my @keys = extract_keywords($file_name,
$num_keywords);
   my $keywords = \@keys;

   open my $fh, '>', $text_file_name or die "Could not create
$text_file_name: $!\n";

   print $fh "File Name: $file_name\n";
   print $fh "Character Count: $char_count\n";
   print $fh "Word Total: $word_total\n";
   print $fh "Line Count: $line_count\n\n";

   print $fh "Top $num_keywords Keywords of $file_name
are as follows :" . join(', ', @$keywords) . "\n";

   print $fh "\nOriginal Contents:\n";
   print $fh $original_contents;

   print $fh "\nModified Contents:\n";
   print $fh $modified_contents;

   close $fh;

   return $text_file_name;
}

sub main {
   print("Enter File Name :");
   my $file_name = <STDIN>;
   chomp($file_name);

   my ($word_count, $char_count, $line_count, $word_total,
$lines_ref) = analyze_text($file_name);
```

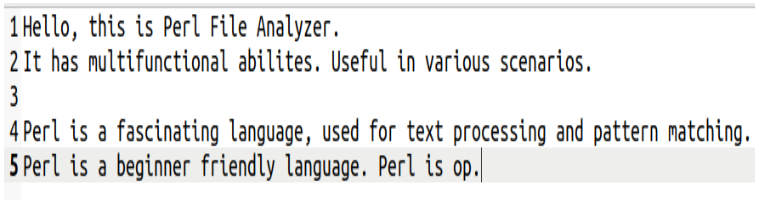
```
    print "Character Count of Original File : $char_count\n";
    print "Line Count of Original File: $line_count\n";
    print "Word Count of Original File: $word_total\n";
#Display word count

    my $num_keywords = 5;
    my @keys = extract_keywords($file_name,
$num_keywords);
    my $keywords = \@keys;
    print "Top $num_keywords Keywords of $file_name are
as follows :" . join(', ', @$keywords) . "\n";

    open my $fh1, '<', $file_name or die "Could not open
'$file_name' for reading: $!\n";

    print("Enter the word you want to search:");
    my $search = <STDIN>;
    chomp($search);

    print("Enter the substitute:");
    my $replace = <STDIN>;
    chomp($replace);

    my ($modified_contents, $original_contents) =
search_and_replace($file_name, $search, $replace);

    print "\n\n\n";
    # Display the original file contents
    print "Original Contents:\n\n";
    print $original_contents;

    print "\n\n\n";
    # Display the modified file contents
    print "Modified Contents:\n\n";
    print $modified_contents;
    print "\n";

    analyze_and_export_to_text($file_name, $search,
$replace);
}

main();
```

Code to convert the text file into a pdf and to send to the designated email address

```
#!/bin/bash

read -p "Enter the file name to be converted :" file
enscript -p temp.ps $file
ps2pdf temp.ps analysis_results.pdf
echo
echo "Text File successfully converted to pdf."
echo
read -p "Enter Email ID :" email_id
mail -s "PDF_Output:" -A analysis_results.pdf $email_id
```

Input

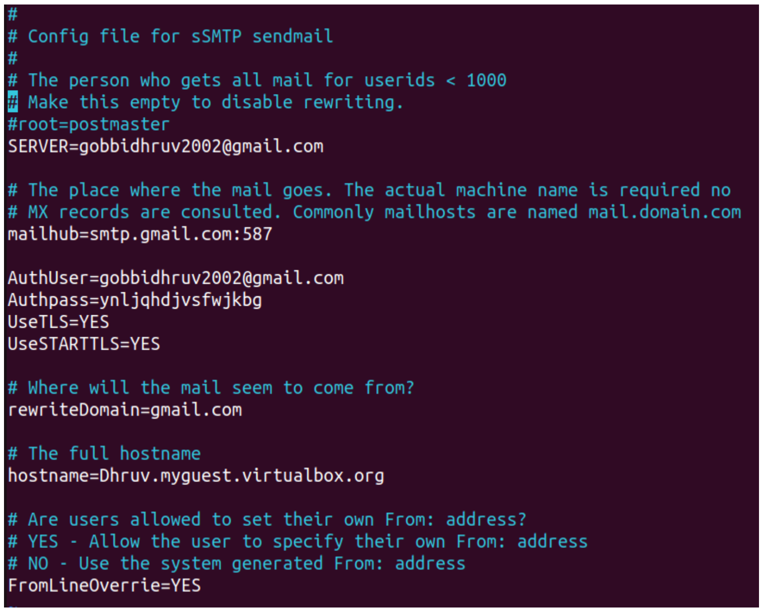
Consider a text file (test3.txt) wherein below text is present in it.



```
Hello, this is Perl File Analyzer.
It has multifunctional abilites. Useful in various scenarios.

Perl is a fascinating language, used for text processing and pattern matching.
Perl is a beginner friendly language. Perl is op.
```

Fig 1: test3.txt

Changes made in the configuration file for SSMTP to send mail from the main server (gobbidhruv2002@gmail.com) as shown below to any given email address.

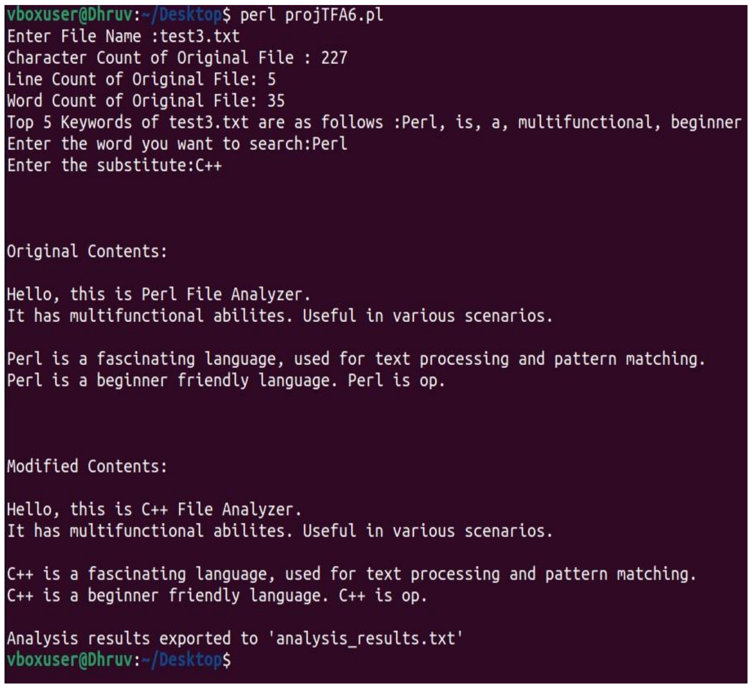


```
#
# Config file for sSMTP sendmail
#
# The person who gets all mail for userids < 1000
# Make this empty to disable rewriting.
#root=postmaster
SERVER=gobbidhruv2002@gmail.com

# The place where the mail goes. The actual machine name is required no
# MX records are consulted. Commonly mailhosts are named mail.domain.com
mailhub=smtp.gmail.com:587

AuthUser=gobbidhruv2002@gmail.com
Authpass=ynljqhdjvsfwjkbg
UseTLS=YES
UseSTARTTLS=YES

# Where will the mail seem to come from?
rewriteDomain=gmail.com

# The full hostname
hostname=Dhruv.myguest.virtualbox.org

# Are users allowed to set their own From: address?
# YES - Allow the user to specify their own From: address
# NO - Use the system generated From: address
FromLineOverrie=YES
```

Fig 2: SSMTP Configuration File

Output

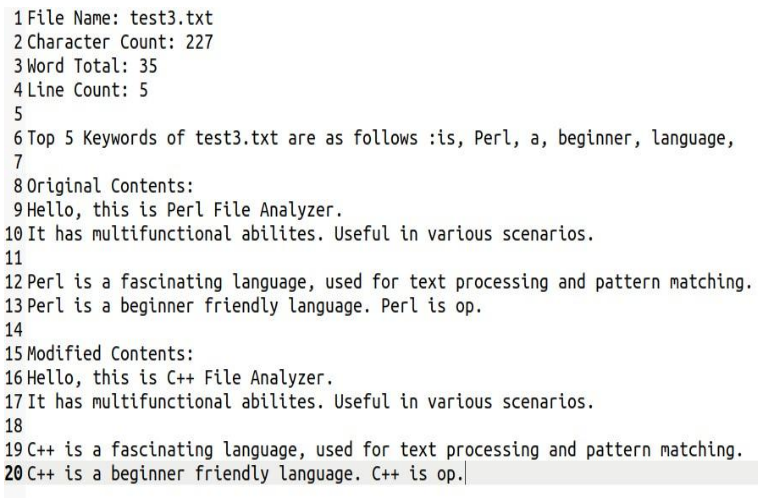
Below shows the analysis of a given text file (test3.txt). Some of the parameters calculated are Character Count, Line Count, Word Count, Top 5 Keywords, also tested the functionality of search and replace function which is shown using an example below.



```
vboxuser@Dhruv:~/Desktop$ perl projTFA6.pl
Enter File Name :test3.txt
Character Count of Original File : 227
Line Count of Original File: 5
Word Count of Original File: 35
Top 5 Keywords of test3.txt are as follows :Perl, is, a, multifunctional, beginner
Enter the word you want to search:Perl
Enter the substitute:C++



Original Contents:

Hello, this is Perl File Analyzer.
It has multifunctional abilites. Useful in various scenarios.

Perl is a fascinating language, used for text processing and pattern matching.
Perl is a beginner friendly language. Perl is op.



Modified Contents:

Hello, this is C++ File Analyzer.
It has multifunctional abilites. Useful in various scenarios.

C++ is a fascinating language, used for text processing and pattern matching.
C++ is a beginner friendly language. C++ is op.

Analysis results exported to 'analysis_results.txt'
vboxuser@Dhruv:~/Desktop$
```

Fig 3: Command Window

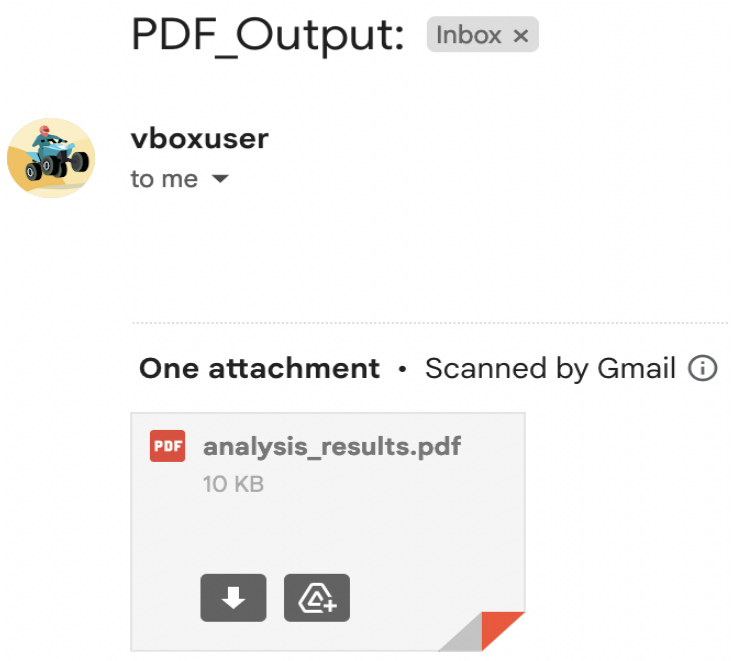
Below shows the results of the Text File Analyzer program in a new text file called analysis_results.txt in a more readable format.



```
File Name: test3.txt
Character Count: 227
Word Total: 35
Line Count: 5

Top 5 Keywords of test3.txt are as follows :is, Perl, a, beginner, language,

Original Contents:
Hello, this is Perl File Analyzer.
It has multifunctional abilites. Useful in various scenarios.

Perl is a fascinating language, used for text processing and pattern matching.
Perl is a beginner friendly language. Perl is op.

Modified Contents:
Hello, this is C++ File Analyzer.
It has multifunctional abilites. Useful in various scenarios.

C++ is a fascinating language, used for text processing and pattern matching.
C++ is a beginner friendly language. C++ is op.
```

Fig 4: analysis_results.txt

Here as you can see below the analysis_results.pdf was sent via email from the main server to any given email address, this was done through a shell script which converted the text file to pdf format and sent the email.



PDF_Output: Inbox ×

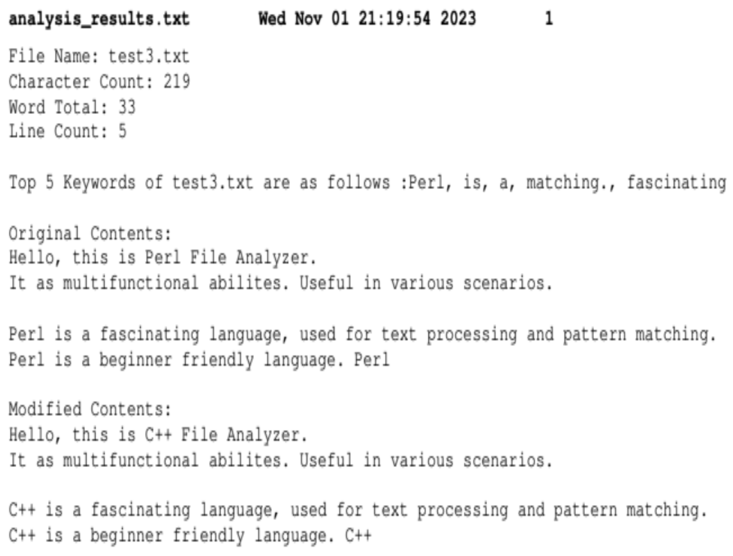
vboxuser
to me

One attachment • Scanned by Gmail

analysis_results.pdf
10 KB

Fig 5: Mail

Below shows the contents of the analysis_results.pdf received from the main email server shown above.



```
analysis_results.txt        Wed Nov 01 21:19:54 2023        1

File Name: test3.txt
Character Count: 219
Word Total: 33
Line Count: 5

Top 5 Keywords of test3.txt are as follows :Perl, is, a, matching., fascinating

Original Contents:
Hello, this is Perl File Analyzer.
It as multifunctional abilites. Useful in various scenarios.

Perl is a fascinating language, used for text processing and pattern matching.
Perl is a beginner friendly language. Perl

Modified Contents:
Hello, this is C++ File Analyzer.
It as multifunctional abilites. Useful in various scenarios.

C++ is a fascinating language, used for text processing and pattern matching.
C++ is a beginner friendly language. C++
```

Fig 6: analysis_results.pdf

Conclusion

Text analysis tools developed in Perl combine core Perl modules and special functions to expertly navigate and handle a variety of text formats. This tool features a wide range of features, including statistical analysis, keyword extraction, content editing, and comprehensive results export. Calculate character counts, word distribution, and line structure using unique file management features. Additionally, text can be replaced while preserving the original content, increasing flexibility and ease of use.This tool interacts seamlessly by prompting for file input, search parameters, and substitutions.A standout feature is the ability to extract important keywords based on relevance and frequency, making it easier to discover insights about important content. Additionally, the tool combines the results into a detailed text file that contains data in addition to the modified content.It also serves as a comprehensive overview, providing practical insights and comprehensive information for researchers and analysts.

Acknowledgment

The successful completion of this report is a testament to the valuable advice and guidance of Prof. Vaishali Dhare in the field of scripting languages. Prof. Dhare's expertise and unwavering support has helped shape and refine the content and methodology presented in this paper. Under Prof Vaishali Dhare's guidance, this report was further developed to provide comprehensive insight into scripting languages. Their in-depth knowledge and encouragement fostered a deeper understanding of the complex concepts and practical applications of this field, making possible the detailed investigation and analysis presented here.

References

[1] Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.

[2] Jurafsky, D., & Martin, J. H. (2019). Speech and Language Processing (3rd ed.). Pearson.

[3] Schwartz, R. L., Phoenix, T., & Foy, B. D. (2012). Learning Perl (6th ed.). O'Reilly Media.

[4] Wall, L., Christiansen, T., & Orwant, J. (2000). Programming Perl (3rd ed.). O'Reilly Media.

[5] Hearst, M. A. (1999). Untangling text data mining. Proceedings of the 37th annual meeting of the Association for Computational Linguistics, 3-10.

[6] Feldman, R., & Sanger, J. (2007). The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data. Cambridge University Press