Name – Devshekhar Pattnaik

Reg No – 19BCE1292

Slot – L19-20

Web Mining Lab-1

1. Explore Unix command GREP

The grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression. Grep stands for globally search for regular expression and print out.

Syntax:

grep [options] pattern [files]

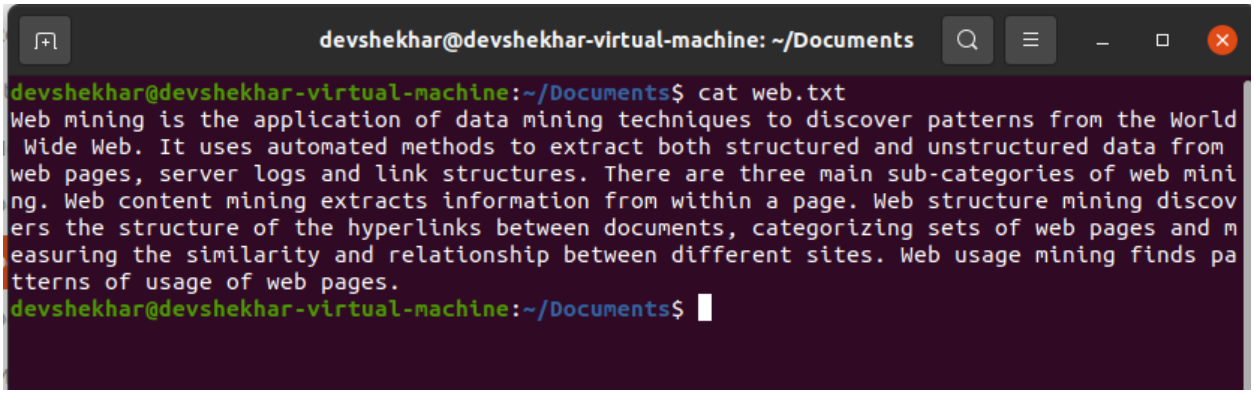
Options Description

- -c : This prints only a count of the lines that match a pattern
- -h : Display the matched lines, but do not display the filenames.
- -i : Ignores, case for matching
- -l : Displays list of a filenames only.
- -n : Display the matched lines and their line numbers.
- -v : This prints out all the lines that do not matches the pattern
- -e exp : Specifies expression with this option. Can use multiple times.
- -f file : Takes patterns from file, one per line.
- -E : Treats pattern as an extended regular expression (ERE)
- -w : Match whole word
- -o : Print only the matched parts of a matching line,
- with each such part on a separate output line.
- -A n : Prints searched line and nlines after the result.

- -B n : Prints searched line and n line before the result.
- -C n : Prints searched line and n lines after before the result.

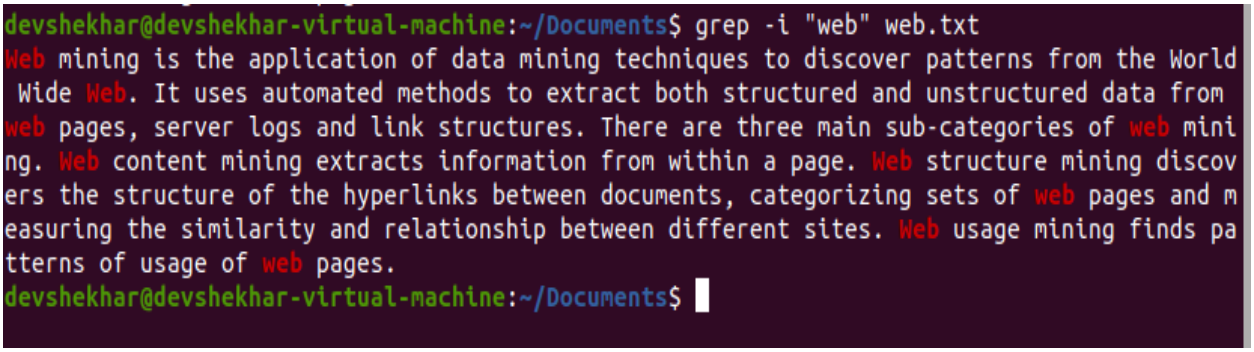
Sample Commands

Input file



1. Case insensitive search

   The -i option enables to search for a string case insensitively in the give file.



2. Displaying the count of number of matches

   We can find the number of lines that matches the given string/pattern.

```
devshekhar@devshekhar-virtual-machine:~/Documents$ grep -c "web" web.txt
1
devshekhar@devshekhar-virtual-machine:~/Documents$
```

3. Display the file names that matches the pattern

   We can just display the files that contains the given string/pattern.

```
devshekhar@devshekhar-virtual-machine:~/Documents$ grep -l "web" *
web.txt
devshekhar@devshekhar-virtual-machine:~/Documents$
```

4. Checking for the whole words in a file

   By default, grep matches the given string/pattern even if it found as a substring in a file. The -w option to grep makes it match only the whole words.

```
devshekhar@devshekhar-virtual-machine:~/Documents$ grep -w "web" web.txt
Web mining is the application of data mining techniques to discover patterns from the Worl
d Wide Web. It uses automated methods to extract both structured and unstructured data fro
m web pages, server logs and link structures. There are three main sub-categories of web m
ining. Web content mining extracts information from within a page. Web structure mining di
scovers the structure of the hyperlinks between documents, categorizing sets of web pages
and measuring the similarity and relationship between different sites. Web usage mining fi
nds patterns of usage of web pages.
devshekhar@devshekhar-virtual-machine:~/Documents$
```

5. Displaying only the matched pattern

   By default, grep displays the entire line which has the matched string. We can make the grep to display only the matched string by using the -o option.

```
devshekhar@devshekhar-virtual-machine:~/Documents$ grep -o "web" web.txt
web
web
web
web
devshekhar@devshekhar-virtual-machine:~/Documents$
```

6. Show line number while displaying the output using grep -n

   To show the line number of file with the line matched.

```
devshekhar@devshekhar-virtual-machine:~/Documents$ grep -n "web" web.txt
1:Web mining is the application of data mining techniques to discover patterns from the Wo
rld Wide Web. It uses automated methods to extract both structured and unstructured data f
rom web pages, server logs and link structures. There are three main sub-categories of web
 mining. Web content mining extracts information from within a page. Web structure mining
discovers the structure of the hyperlinks between documents, categorizing sets of web page
s and measuring the similarity and relationship between different sites. Web usage mining
finds patterns of usage of web pages.
devshekhar@devshekhar-virtual-machine:~/Documents$
```

7. Inverting the pattern match

   You can display the lines that are not matched with the specified search sting pattern using the -v option.

```
devshekhar@devshekhar-virtual-machine:~/Documents$ grep -v "Mining" web.txt
Web mining is the application of data mining techniques to discover patterns from the Worl
d Wide Web. It uses automated methods to extract both structured and unstructured data fro
m web pages, server logs and link structures. There are three main sub-categories of web m
ining. Web content mining extracts information from within a page. Web structure mining di
scovers the structure of the hyperlinks between documents, categorizing sets of web pages
and measuring the similarity and relationship between different sites. Web usage mining fi
nds patterns of usage of web pages.
devshekhar@devshekhar-virtual-machine:~/Documents$
```

8. Matching the lines that start with a string

The ^ regular expression pattern specifies the start of a line. This can be used in grep to match the lines which start with the given string or pattern.

```
devshekhar@devshekhar-virtual-machine:~/Documents$ grep "^Web" web.txt
Web mining is the application of data mining techniques to discover patterns from the Worl
d Wide Web. It uses automated methods to extract both structured and unstructured data fro
m web pages, server logs and link structures. There are three main sub-categories of web m
ining. Web content mining extracts information from within a page. Web structure mining di
scovers the structure of the hyperlinks between documents, categorizing sets of web pages
and measuring the similarity and relationship between different sites. Web usage mining fi
nds patterns of usage of web pages.
devshekhar@devshekhar-virtual-machine:~/Documents$
```

9. Matching the lines that end with a string

The $ regular expression pattern specifies the end of a line. This can be used in grep to match the lines which end with the given string or pattern.

```
devshekhar@devshekhar-virtual-machine:~/Documents$ grep "Web$" web.txt
devshekhar@devshekhar-virtual-machine:~/Documents$
```

2. Write a program to create the inverted index and execute for the following document collections. (See Figure 1.3 for an example.)

a)

Doc 1 new home sales top forecasts

Doc 2 home sales rise in july

Doc 3 increase in home sales in july

Doc 4 july new home sales rise

b)

Doc 1 breakthrough drug for schizophrenia

Doc 2 new schizophrenia drug

Doc 3 new approach for treatment of schizophrenia

Doc 4 new hopes for schizophrenia patients

Algorithm

1. Append all the terms in each document into list term and their respective document no. to list docID
2. Sort the term list and sort the docID list wrt the term list and store them in sorted_term and sorted_docID.
3. From sorted_term and sorted_docID delete the term and docID if the same term is repeated more than once in the same document.
4. From the sorted_term find the unique terms and store it in the unique list.
5. For each term in the unique list find the frequency count by comparing with the sorted_term list and append term frequency and the list of documents to freq and l respectively.
6. Store the terms and frequencies in the unique and freq list as key value pairs in the term_dic dictionary
7. Print each key value pair in term_dic dictionary and its respective document list from l list.

Data structures used

1. List
2. Dictionary

Code

```
term=[]
docID=[]
for i in range(1,5):
    f=open('doc'+str(i)+'.txt','r')
    for x in f:
        for word in x.split():
            term.append(word)
            docID.append(i)
sorted_docID= [x for _,x in sorted(zip(term,docID))]
sorted_term=term.copy()
sorted_term.sort()
n=len(sorted_term)
for i in range(0,n-1):
    for j in range(i+1,n-1):
        if sorted_term[i]==sorted_term[j]:
            if sorted_docID[i]==sorted_docID[j]:
                del(sorted_term[i])
                del(sorted_docID[i])
freq=[]
unique=[]
for i in sorted_term:
    if i not in unique:
        unique.append(i)
l1=[]
for x in range(0,len(unique)):
    count=0
    l=[]
    for i in range(0,len(sorted_term)):
        if unique[x]==sorted_term[i]:
            count=count+1
            l.append(sorted_docID[i])
    l1.append(l)
    freq.append(count)
term_dic={}
for i in range(0,len(unique)):
    term_dic[unique[i]]=freq[i]
```

```
j=0
print("term:","doc_freq->","posting_list")
for key,value in term_dic.items():
    print(key,":",value,"->",l1[j])
    j=j+1
```

Output

a.

```
PS D:\Programs\Python\.vscode> python -u "d:\Programs\Python\.vscode\web_mining_lab1_2.py"
term: doc_freq-> posting_list
forecasts : 1 -> [1]
home : 4 -> [1, 2, 3, 4]
in : 2 -> [2, 3]
increase : 1 -> [3]
july : 3 -> [2, 3, 4]
new : 2 -> [1, 4]
rise : 2 -> [2, 4]
sales : 4 -> [1, 2, 3, 4]
top : 1 -> [1]
PS D:\Programs\Python\.vscode>
```

b.

```
PS D:\Programs\Python\.vscode> python -u "d:\Programs\Python\.vscode\web_mining_lab1_2.py"
term: doc_freq-> posting_list
approach : 1 -> [3]
breakthrough : 1 -> [1]
drug : 2 -> [1, 2]
for : 3 -> [1, 3, 4]
hopes : 1 -> [4]
new : 3 -> [2, 3, 4]
of : 1 -> [3]
patients : 1 -> [4]
schizophrenia : 4 -> [1, 2, 3, 4]
treatment : 1 -> [3]
PS D:\Programs\Python\.vscode>
```

3. Generate term-document incidence matrix for a) and b).

Algorithm

1. Append all the terms in each document in terms list.
2. Append all the unique terms in the terms list in unique list.
3. Sort the unique list.
4. For each Document for each term in unique list if the term is present in the document append 1 to a list p else append 0,Then append the list p to list m after each term.
5. Print the terms in the unique list and their respective values for each document using list m.

Data structures used

1. List

Code

```
terms=[]
for i in range(1,5):
    f=open('doc'+str(i)+'.txt','r')
    for x in f:
        for word in x.split():
            terms.append(word)
unique=[]
for x in terms:
    if x not in unique:
        unique.append(x)
m=[]
unique.sort()
for y in unique:
    p = []
    for i in range(1, 5):
        flag=0
        f = open('doc' + str(i) + '.txt', 'r')
        for x in f:
            for word in x.split():
```

```
                if word==y:
                    flag=1
                    p.append(1)
                    break
        if flag==0:
            p.append(0)
    m.append(p)
for i in range(0,len(m)):
    print(unique[i],":",end=' ')
    for j in range(0,4):
        print(m[i][j],end=' ')
    print("")
```

Output

a.

```
PS D:\Programs\Python\.vscode> python -u "d:\Programs\Python\.vscode\web_mining_lab1_3.py"
forecasts : 1 0 0 0
home : 1 1 1 1
in : 0 1 1 0
increase : 0 0 1 0
july : 0 1 1 1
new : 1 0 0 1
rise : 0 1 0 1
sales : 1 1 1 1
top : 1 0 0 0
PS D:\Programs\Python\.vscode>
```

b.

```
PS D:\Programs\Python\.vscode> python -u "d:\Programs\Python\.vscode\web_mining_lab1_3.py"
approach : 0 0 1 0
breakthrough : 1 0 0 0
drug : 1 1 0 0
for : 1 0 1 1
hopes : 0 0 0 1
new : 0 1 1 1
of : 0 0 1 0
patients : 0 0 0 1
schizophrenia : 1 1 1 1
treatment : 0 0 1 0
PS D:\Programs\Python\.vscode>
```

4. For the document collections shown in a) and b), compute the results for these queries using above matrix as well as inverted index created above :

a. schizophrenia AND drug

Using Inverted Index:

Code

```
term=[]
docID=[]
for i in range(1,5):
    f=open('doc'+str(i)+'.txt','r')
    for x in f:
        for word in x.split():
            term.append(word)
            docID.append(i)
sorted_docID= [x for _,x in sorted(zip(term,docID))]
sorted_term=term.copy()
sorted_term.sort()
n=len(sorted_term)
for i in range(0,n-1):
    for j in range(i+1,n-1):
        if sorted_term[i]==sorted_term[j]:
            if sorted_docID[i]==sorted_docID[j]:
                del(sorted_term[i])
                del(sorted_docID[i])
freq=[]
unique=[]
for i in sorted_term:
    if i not in unique:
        unique.append(i)
l1=[]
for x in range(0,len(unique)):
    count=0
    l=[]
    for i in range(0,len(sorted_term)):
        if unique[x]==sorted_term[i]:
            count=count+1
            l.append(sorted_docID[i])
    l1.append(l)
    freq.append(count)
term_dic={}
```

```
for i in range(0,len(unique)):
    term_dic[unique[i]]=freq[i]
print("Enter query terms: ")
word1 = input()
word2 = input()
l3 = []
l4 = []
j = 0
for key, value in term_dic.items():
    if key == word1:
        l3 = l1[j]
    if key == word2:
        l4 = l1[j]
    j = j + 1
doc = ['doc1', 'doc2', 'doc3', 'doc4']
flag = 0
print("query result: ")
if len(l3) > len(l4):
    for x in l4:
        if x in l3:
            print(doc[x - 1], end=' ')
            flag = 1
    if flag == 0:
        print("Not present in any document")
else:
    for x in l3:
        if x in l4:
            print(doc[x - 1], end=' ')
            flag = 1
    if flag == 0:
        print("Not present in any document")
```

Output

a.

```
PS D:\Programs\Python\.vscode> python -u "d:\Programs\Python\.vscode\web_mining_lab1_4_1.py"
Enter query terms:
schizophrenia
drug
query result:
Not present in any document
PS D:\Programs\Python\.vscode>
```

b.

```
PS D:\Programs\Python\.vscode> python -u "d:\Programs\Python\.vscode\web_mining_lab1_4_1.py"
Enter query terms:
schizophrenia
drug
query result:
doc1 doc2
PS D:\Programs\Python\.vscode>
```

Using term-document incidence matrix

Code

```
terms=[]
for i in range(1,5):
    f=open('doc'+str(i)+'.txt','r')
    for x in f:
        for word in x.split():
            terms.append(word)
unique=[]
for x in terms:
    if x not in unique:
        unique.append(x)
m=[]
unique.sort()
for y in unique:
    p = []
    for i in range(1, 5):
        flag=0
        f = open('doc' + str(i) + '.txt', 'r')
        for x in f:
            for word in x.split():
                if word==y:
                    flag=1
                    p.append(1)
                    break
        if flag==0:
            p.append(0)
    m.append(p)
print("Enter query terms: ")
word1 = input()
word2 = input()
l1 = [0, 0, 0, 0]
l2 = [0, 0, 0, 0]
l3 = []
for i in range(0, len(m)):
    if unique[i] == word1:
        for j in range(0, 4):
            l1[j] = (m[i][j])
    if unique[i] == word2:
        for j in range(0, 4):
            l2[j] = (m[i][j])
for i in range(0, 4):
    if l1[i] == 1 and l2[i] == 1:
```

```
        l3.append(1)
    else:
        l3.append(0)
doc = ['doc1', 'doc2', 'doc3', 'doc4']
flag = 0
print("query result: ")
for i in range(0, 4):
    if l3[i] == 1:
        print(doc[i], end=' ')
        flag = 1
if flag == 0:
    print("Not present in any document")
```

Output

a.

```
PS D:\Programs\Python\.vscode> python -u "d:\Programs\Python\.vscode\web_mining_lab1_4_2.py"
Enter query terms:
schizophrenia
drug
query result:
Not present in any document
PS D:\Programs\Python\.vscode>
```

b.

```
PS D:\Programs\Python\.vscode> python -u "d:\Programs\Python\.vscode\web_mining_lab1_4_2.py"
Enter query terms:
schizophrenia
drug
query result:
doc1 doc2
PS D:\Programs\Python\.vscode>
```

b. for AND NOT(drug OR approach)

Using Inverted Index:

Code

```
term=[]
docID=[]
for i in range(1,5):
    f=open('doc'+str(i)+'.txt','r')
    for x in f:
        for word in x.split():
            term.append(word)
            docID.append(i)
sorted_docID= [x for _,x in sorted(zip(term,docID))]
sorted_term=term.copy()
sorted_term.sort()
n=len(sorted_term)
for i in range(0,n-1):
    for j in range(i+1,n-1):
        if sorted_term[i]==sorted_term[j]:
            if sorted_docID[i]==sorted_docID[j]:
                del(sorted_term[i])
                del(sorted_docID[i])
freq=[]
unique=[]
for i in sorted_term:
    if i not in unique:
        unique.append(i)
l1=[]
for x in range(0,len(unique)):
    count=0
    l=[]
    for i in range(0,len(sorted_term)):
        if unique[x]==sorted_term[i]:
            count=count+1
            l.append(sorted_docID[i])
    l1.append(l)
    freq.append(count)
term_dic={}
for i in range(0,len(unique)):
    term_dic[unique[i]]=freq[i]
print("Enter query terms: ")
word1 = input()
word2 = input()
```

```
word3 = input()
l3 = []
l4 = []
l5 = []
j = 0
for key, value in term_dic.items():
    if key == word1:
        l3 = l1[j]
    if key== word2:
        l4=l1[j]
    if key==word3:
        l5=l1[j]
    j = j + 1
doc = ['doc1', 'doc2', 'doc3', 'doc4']
flag = 0
print("query result: ")
for x in l3:
    if x not in l4:
        if x not in l5:
            print(doc[x-1],end=' ')
            flag=1
if flag==0:
    print("Not present in any document")
```

Output

a.

```
PS D:\Programs\Python\.vscode> python -u "d:\Programs\Python\.vscode\web_mining_lab1_4_3.py"
Enter query terms:
for
drug
approach
query result:
Not present in any document
PS D:\Programs\Python\.vscode>
```

b.

```
PS D:\Programs\Python\.vscode> python -u "d:\Programs\Python\.vscode\web_mining_lab1_4_3.py"
Enter query terms:
for
drug
approach
query result:
doc4
PS D:\Programs\Python\.vscode>
```

Using term-document incidence matrix

Code

```
terms=[]
for i in range(1,5):
    f=open('doc'+str(i)+'.txt','r')
    for x in f:
        for word in x.split():
            terms.append(word)
unique=[]
for x in terms:
    if x not in unique:
        unique.append(x)
m=[]
unique.sort()
for y in unique:
    p = []
    for i in range(1, 5):
        flag=0
        f = open('doc' + str(i) + '.txt', 'r')
        for x in f:
            for word in x.split():
                if word==y:
                    flag=1
                    p.append(1)
                    break
        if flag==0:
            p.append(0)
    m.append(p)
print("Enter query terms: ")
word1 = input()
word2 = input()
word3 = input()
l1 = [0, 0, 0, 0]
l2 = [0, 0, 0, 0]
l3 = [0, 0, 0, 0]
l4=[]
l5=[]
for i in range(0, len(m)):
    if unique[i] == word1:
        for j in range(0, 4):
            l1[j] = (m[i][j])
    if unique[i] == word2:
        for j in range(0, 4):
```

```
            l2[j] = (m[i][j])
    if unique[i] == word3:
        for j in range(0, 4):
            l3[j] = (m[i][j])
for i in range(0, 4):
    if l2[i] == 0 and l3[i] == 0:
        l4.append(0)
    else:
        l4.append(1)
for i in range(0,4):
    if l4[i]==0:
        l4[i]=1
    else:
        l4[i]=0
for i in range(0, 4):
    if l1[i] == 1 and l4[i] == 1:
        l5.append(1)
    else:
        l5.append(0)
doc = ['doc1', 'doc2', 'doc3', 'doc4']
flag = 0
print("query result: ")
for i in range(0, 4):
    if l5[i] == 1:
        print(doc[i], end=' ')
        flag = 1
if flag == 0:
    print("Not present in any document")
```

Output

a.

```
PS D:\Programs\Python\.vscode> python -u "d:\Programs\Python\.vscode\web_mining_lab1_4_4.py"
Enter query terms:
for
drug
approach
query result:
Not present in any document
PS D:\Programs\Python\.vscode>
```

b.

```
PS D:\Programs\Python\.vscode> python -u "d:\Programs\Python\.vscode\web_mining_lab1_4_4.py"
Enter query terms:
for
drug
approach
query result:
doc4
PS D:\Programs\Python\.vscode>
```