**Plagiarism Checker**

|  |  |  |
| --- | --- | --- |
| Collaborators | Platform | Language |
| 2018-CS-96 | Web Based | Python |
| 2018-CS-94 |  |  |

**Milestone: ( 2 )**

**Working:**

**1)** Check English Text **2)** Check C++ Code

There are two pseudo codes one for English text and one for C++ files checking.

**Inputs:**

The input is multiple files in a zip folder. And the folder will be uploaded to “media” directory in server. From there we will retrieve data.

**Outputs:**

Our main code will generate a list of all matching words in all files. And the percentage in a variable.

Then an additional code will convert this data into meaning full information. Like the whole files will be converted to “pdf” file and highlights the words that are in above list. And the total percentage will be provided. User can download this file or view on browser.

**-----------------------------------------------------------------------------------------------------------**

**Pseudo Code for English Text Checker**

**-----------------------------------------------------------------------------------------------------------**

def unzip(): # unzip the provided zip folder

with ZipFile(“input.zip”,’r’) as zipobj: # “ZipFile” is external library

zipobj.extract(“data”) # Extraction folder

names=zipobj.nameslist() # All names of files include

files=[]

def storefiles(): # store the pointers of files in list

for x in enumerate(names):

f=open(“data/”+str(x),’r’)

files.append(f)

def Reverse(tuples): # reverse the order of tuples

new\_tup=()

for k in reversed(tuples):

new\_tup=new\_tup+(k,)

return new\_tup

combo1=[]

combo=[]

def combination(): # create combinations of files to

l=len(files) # be matched and store in combo-

index=list(range(l)) # list

perm=permutations(index,2) # pair of 2 files to be checked

for i in list(perm):

combo1.append(i)

for i in range(len(combo1)):

for j in range(i):

if Reverse(combo1[i])==combo1[j]:

combo.append(Reverse(combo1[i]))

if \_\_name==”\_\_main\_\_”:

unzip()

storefiles()

combination()

final \_words=[]

final\_words\_count=[]

matched=[]

percent\_list=[]

all\_data=[]

files\_checked=[]

for i in combo:

matching=[]

matching\_count1=[]

checking\_count2=[]

total\_words=0

total\_match\_words=0

percent=0

f1=files[i[0]].read()

f2=files[i[1]].read()

f3=split f1 on basis of newline

f4=split f2 on basis of newline

pop last element of list f3

pop last element of list f4

join all elements of f3 in a string f5

join all elementts of f4 in a string f6

split the string f5 on basis of ‘.’ and store in a list l1 #List contains whole

split the string f6 on basis of ‘.’ and store in a list l2 #Sentences separated by ‘.’

pop last element of list l1

pop last element of list l2

join all elements of list l1 in a string l3

join all elements of list l2 in a string l4

split the string l3 on basis of ‘ ‘ space and store in l5 # 2 Lists of all words in both

split the string l4 on basis of ‘ ‘ apace and store in l6 # files

if i[0] and i[1] not in files\_checked:

if i[0] not in files\_checked:

all\_data.append(l5)

else:

all\_data.append(l6)

for p in l5:

count=1

if p in matching:

# ind is index where ‘p’ is stored in matching list

matching\_count1[ind]=matching\_count[ind]+1

else:

for q in l6:

if p.lower==q.lower:

if count==1:

matching.append(p)

matching\_count1.append(1)

matching\_count2.append(1)

else:

# ind is index where ‘p’ is stored in matching list

matching\_count2[ind]=matching\_count2[ind]+1

count=2

total\_words=len(l5)+len(l6)

w1=0

w2=0

for w in matching\_count1:

w1=w1+w

for w in matching\_count2:

w2=w2+w

total\_match\_words=w1+w2

percent=(total\_match\_words/total\_words)\*100

matched.append(matching)

percent\_list.append(percent)

“ here we have a “**matching**”, “**matching\_count1**” and “**matching\_count2**” these contain **matching** **words** of individual combination of files and **count value** of matching words in “**file1**” and “**file2**”.

We will call **additional function** which prints all data into **pdf page** which prints a page contains **matching words of current file combination and their count values and the percentage of combination**. “

stat=0

pstat=1

for it in matched[0]:

for ids in range(1,len(matched)):

for its in matched[ids]:

if it==its:

stat=1

else:

stat=0

pstat=0

if stat==1 and pstat==1:

final\_words.append(it)

“ “**final\_words**” this list have all words **common in all files**. Now we will find these words **occurs** in all files **how many times**? “

for i in final\_words:

value=0

for j in range(len(files)):

for k in all\_data[j]:

value=value+k.count(i)

final\_words\_count.append(value)

all\_len=0

for i in all\_data:

all\_len=all\_len+len(i)

Sum=sum(final\_data\_count)

total\_percent=(Sum/all\_len)\*100

“ now we have 2 lists “**final\_words**” and “**final\_words\_count**” which contain all words common in all files and there count value. We call **additional function** to print **pdf** of these two files. The common words and their count. We can also find the percentage of matching words over all words in all files and the **final percentage**. Then **merge all pdf into single file but final result will be on top.**”

**----------------------------------------------------------------------------------------------------------------------------**

**Pseudo Code for C++ Checker**

**----------------------------------------------------------------------------------------------------------**

**Difference between English text and C++ code checker:**

**1)** In this checker we will not split lines on basis of period ‘.’ here lines are separated on basis of new line ‘\n’.

**2)**  In C++ code the space character is widely used ‘ ‘. So we have to remove them from file or ignore.

**3)**  One more thing the curly braces are used to a lot “ { } ” so we also have to ignore them.

4) In this checker we will not check files words by words. We will check them line by line. If a complete line is same in other file then we will highlight that line.

5) In C++ code the comments is widely used ‘// ‘. So we have to remove them from file or ignore.

**Pseudo Code:**

def unzip(): # unzip the provided zip folder

with ZipFile(“input.zip”,’r’) as zipobj: # “ZipFile” is external library

zipobj.extract(“data”) # Extraction folder

names=zipobj.nameslist() # All names of files include

files=[]

def storefiles(): # store the pointers of files in list

for x in enumerate(names):

f=open(“data/”+str(x),’r’)

files.append(f)

def Reverse(tuples): # reverse the order of tuples

new\_tup=()

for k in reversed(tuples):

new\_tup=new\_tup+(k,)

return new\_tup

combo1=[]

combo=[]

def combination(): # create combinations of files to

l=len(files) # be matched and store in combo-

index=list(range(l)) # list

perm=permutations(index,2) # pair of 2 files to be checked

for i in list(perm):

combo1.append(i)

for i in range(len(combo1)):

for j in range(i):

if Reverse(combo1[i])==combo1[j]:

combo.append(Reverse(combo1[i]))

if \_\_name==”\_\_main\_\_”:

unzip()

storefiles()

combination()

final \_words=[]

final\_words\_count=[]

matched=[]

percent\_list=[]

all\_data=[]

files\_checked=[]

for i in combo:

matching=[]

matching\_count1=[]

checking\_count2=[]

total\_words=0

total\_match\_words=0

percent=0

f1=files[i[0]].read()

f2=files[i[1]].read()

f3=split f1 on basis of newline

f4=split f2 on basis of newline

for ii in f3:

k=ii.strip() # Removing all spaces before and end of string

k1=’ ‘.join(k.split()) # Splitting and joining by space no more than one space

f5.append(k1)

for ii in f4:

k=ii.strip() # Removing all spaces before and end of string

k1=’ ‘.join(k.split()) # Splitting and joining by space no more than one space

f6.append(k1)

if i[0] and i[1] not in files\_checked:

if i[0] not in files\_checked:

all\_data.append(l5)

else:

all\_data.append(l6)

for p in f5:

count=1

if p not startswith(“{”):

if p not startswith(“}”):

if p not startswith(“//”):

if p in matching:

# ind is index where ‘p’ is stored in matching list

matching\_count1[ind]=matching\_count[ind]+1

else:

for q in l6:

if p.lower==q.lower:

if count==1:

matching.append(p)

matching\_count1.append(1)

matching\_count2.append(1)

else:

# ind is index where ‘p’ is stored in matching list matching\_count2[ind] = matching\_count2[ind]+1

count=2

total\_words=len(l5)+len(l6)

w1=0

w2=0

for w in matching\_count1:

w1=w1+w

for w in matching\_count2:

w2=w2+w

total\_match\_words=w1+w2

percent=(total\_match\_words/total\_words)\*100

matched.append(matching)

percent\_list.append(percent)

“ here we have a “**matching**”, “**matching\_count1**” and “**matching\_count2**” these contain **matching** **code lines** of individual combination of files and **count value** of matching words in “**file1**” and “**file2**”.

We will call **additional function** which prints all data into **pdf page** which prints a page contains **matching code lines of current file combination and their count values and the percentage of combination**. “

stat=0

pstat=1

for it in matched[0]:

for ids in range(1,len(matched)):

for its in matched[ids]:

if it==its:

stat=1

else:

stat=0

pstat=0

if stat==1 and pstat==1:

final\_words.append(it)

“ “**final\_words**” this list have all code lines **common in all files**. Now we will find these code lines **occurs** in all files **how many times**? “

for i in final\_words:

value=0

for j in range(len(files)):

for k in all\_data[j]:

value=value+k.count(i)

final\_words\_count.append(value)

all\_len=0

for i in all\_data:

all\_len=all\_len+len(i)

Sum=sum(final\_data\_count)

total\_percent=(Sum/all\_len)\*100

“ now we have 2 lists “**final\_words**” and “**final\_words\_count**” which contain all words common in all files and there count value. We call **additional function** to print **pdf** of these two files. The common words and their count. We can also find the percentage of matching words over all words in all files and the **final percentage**. Then **merge all pdf into single file but final result will be on top.**”

**----------------------------------------------------------------------------------------------------------------------------**