AP17110010129-Rakush Rimal
AP17110010130-Cyrus Maharjan
AP17110010131-Ramesh K. Yadav

Plagiarism Detector

Abstract:

As the technology is growing very fast and usage of computer systems are increasing as compared to the old times, plagiarism is the phenomenon which is increasing day by day. Wrongful appropriation of someone else's work is known as plagiarism. Detection of plagiarism manually is really very difficult so this process should be automated. We can use various tools and techniques which helps in plagiarism detection. Some work on intrinsic plagiarism while others work on extrinsic plagiarism. Data mining techniques can be used for the detection of plagiarism as well as can help to improve the efficiency of the process.

Introduction:

Due to easy availability of computer devices and the advancement of the internet has made it easy to access others' work and copy them which leads to plagiarism. So, Plagiarism can be defined as the act of using someone else's work without the information of the author or without giving acknowledgement to that corresponding person.

Plagiarism can be found growing very vastly everywhere mostly in schools, institutes and industries. We can find that students nowadays mostly submit their assignments and works in electronic forms because e-form is easy and suitable for both teachers and students but leads towards plagiarism. With the advancement of technology and the internet, it has become easy to copy the data from different sources through the internet on the basis of various online papers, books over the internet and simply paste them in a single work and submit it. This action hampers the learning of the students.

Not only in the educational field, plagiarism can occur in other fields as well like literature eg: novels, source codes, research paper etc. People sometimes do it intentionally but mostly they do it unintentionally due to lack of awareness about the usage of resources according to their own need.

Therefore, plagiarism can be classified into various forms. Some are easily detectable and some are complex. Some of the forms are:
1)Coping & pasting: Copying and pasting is the type of plagiarism in which a single sentence, a whole paragraph or a complete page of any written text is copied without any reference.
2)Re-using existing work: It is the type of plagiarism in which the existing work or already written e-data is used again and again.
3)Manipulating the text: In this type of plagiarism the text is modified and its appearance is changed.

4) Translating the text: When data is translated from one language to another without giving any reference of the source.
5) Plagiarizing the idea: One the major form in which someone else's idea is used without acknowledging the owner.
6) Incorrect citation: Citation of unread sources and without giving acknowledge to the other sources from where the data has been read.
7) Self-plagiarism: The type in which the author uses his own previously done work and presents as new one with any reference to prior work.

Problem Survey:

So, what can we do to detect the plagiarism between the respective works or projects? It is really very difficult to detect the plagiarism manually. So it must be automated so that it can be done efficiently. For this purpose, there are different data mining techniques and ways to implement this. Let's take an example:

• Algorithms to compare documents.
• Crawler to search data from the websites
• Methods using the language-specific structures and much more.

Data mining is one of the fields which can be used for this purpose through which relations in existing data can be mined.

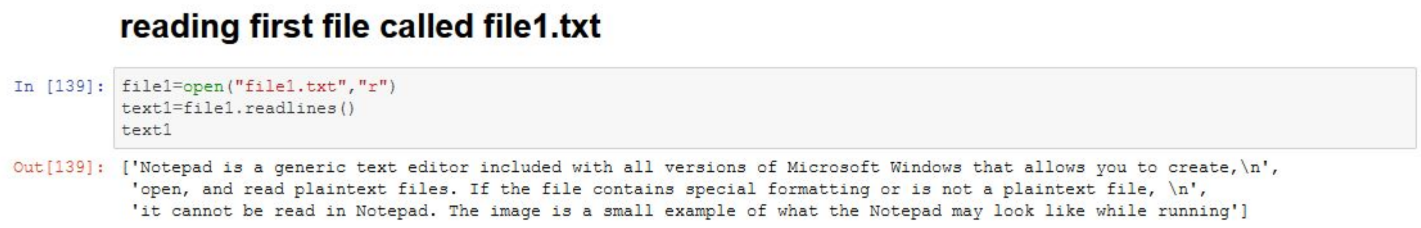
Algorithm:

We can use different data mining techniques to detect plagiarism easily and effectively. We have proposed a methodology for this purpose which is explained below.

Collection of assignments: As all the documents are collected first in electronic format so that plagiarism can be detected.
Pre-processing: Pre-processing is a major step in the process in which all the assignments are converted into an appropriate format. The collected assignments should be in the same format.The documents should be scanned and exclude the things such as numbers, figure values, pictures and all those things which are not from a-z group.
Classification: So, first we took two text files and imported them into the program. Then to compare them we converted them to string and later splitted them into sentences in order to verify the plagiarism or get the similarity score.
Text analysis: Further, the data of both text files will be compared and passed through the text analyzing step.
Processing: If the contents of the files are matched with each other then the matched data is marked and will be saved to another new text file which is mentioned in the below implementation part.

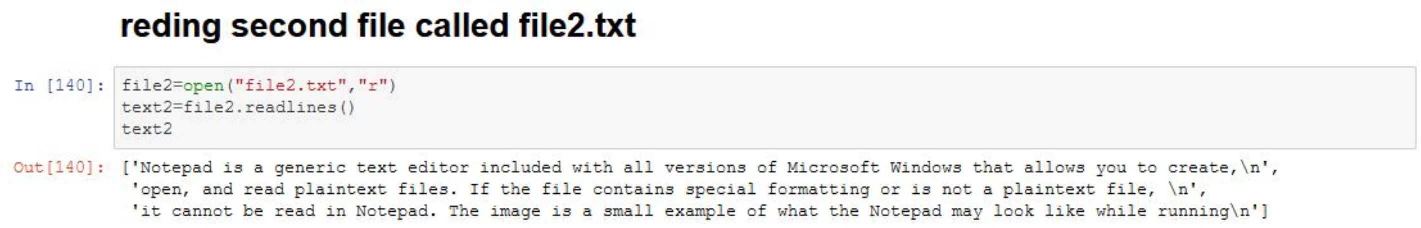
Similarity score: Similarity score is then calculated by the sequence matcher technique. It is calculated in the form of ratio and later can be determined in percentage. Higher the value of percentage defines the high similarity of data between the compared text files.

Implementation:



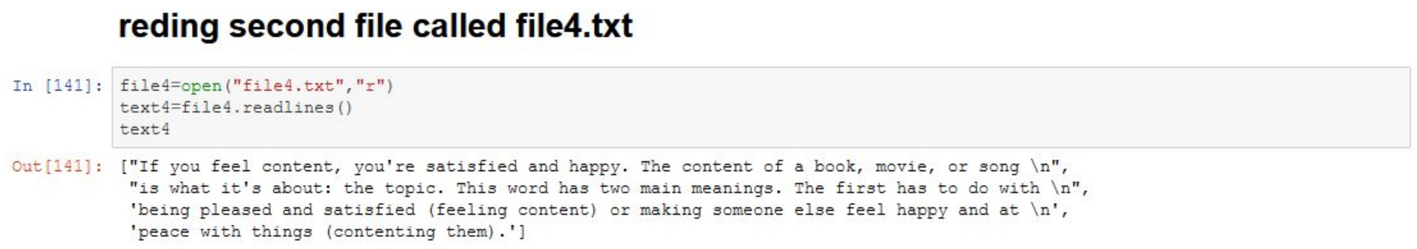
reading first file called file1.txt

```
In [139]: file1=open("file1.txt","r")
          text1=file1.readlines()
          text1

Out[139]: ['Notepad is a generic text editor included with all versions of Microsoft Windows that allows you to create,\n',
           'open, and read plaintext files. If the file contains special formatting or is not a plaintext file, \n',
           'it cannot be read in Notepad. The image is a small example of what the Notepad may look like while running']
```



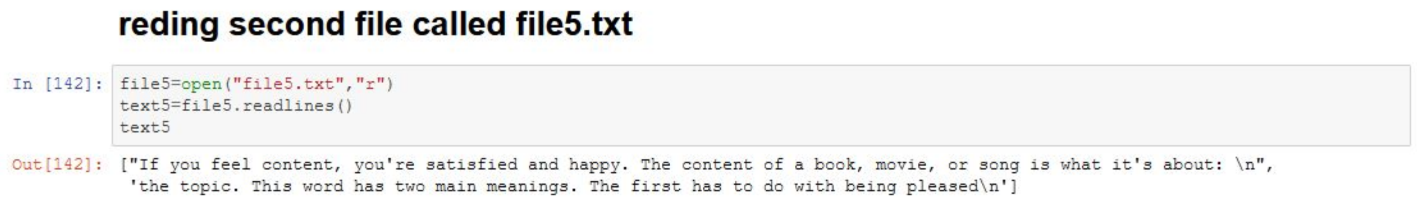
reding second file called file2.txt

```
In [140]: file2=open("file2.txt","r")
          text2=file2.readlines()
          text2

Out[140]: ['Notepad is a generic text editor included with all versions of Microsoft Windows that allows you to create,\n',
           'open, and read plaintext files. If the file contains special formatting or is not a plaintext file, \n',
           'it cannot be read in Notepad. The image is a small example of what the Notepad may look like while running\n']
```



reding second file called file4.txt

```
In [141]: file4=open("file4.txt","r")
          text4=file4.readlines()
          text4

Out[141]: ["If you feel content, you're satisfied and happy. The content of a book, movie, or song \n",
           "is what it's about: the topic. This word has two main meanings. The first has to do with \n",
           'being pleased and satisfied (feeling content) or making someone else feel happy and at \n',
           'peace with things (contenting them).']
```



reding second file called file5.txt

```
In [142]: file5=open("file5.txt","r")
          text5=file5.readlines()
          text5

Out[142]: ["If you feel content, you're satisfied and happy. The content of a book, movie, or song is what it's about: \n",
           'the topic. This word has two main meanings. The first has to do with being pleased\n']
```

converting list into string type

```
In [143]: str1
```

```
Out[143]: 'Notepad is a generic text editor included with all versions of Microsoft Windows that allows you to create,\nopen, and rea
          d plaintext files. If the file contains special formatting or is not a plaintext file, \nit cannot be read in Notepad. The
          image is a small example of what the Notepad may look like while running'
```

```
In [144]: str2
```

```
Out[144]: 'Notepad is a generic text editor included with all versions of Microsoft Windows that allows you to create,\nopen, and rea
          d plaintext files. If the file contains special formatting or is not a plaintext file, \nit cannot be read in Notepad. The
          image is a small example of what the Notepad may look like while running\n'
```

```
In [145]: str4
```

```
Out[145]: "If you feel content, you're satisfied and happy. The content of a book, movie, or song \nis what it's about: the topic. Th
          is word has two main meanings. The first has to do with \nbeing pleased and satisfied (feeling content) or making someone e
          lse feel happy and at \npeace with things (contenting them)."
```

```
In [146]: str5
```

```
Out[146]: "If you feel content, you're satisfied and happy. The content of a book, movie, or song is what it's about: \nthe topic. Th
          is word has two main meanings. The first has to do with being pleased\n"
```

splitted strings into sentences

```
In [147]: sent_text1=str1.split('.')
          sent_text2=str2.split('.')
          sent_text4=str4.split('.')
          sent_text5=str5.split('.')
```

```
In [148]: sent_text1
```

```
Out[148]: ['Notepad is a generic text editor included with all versions of Microsoft Windows that allows you to create,\nopen, and re
          ad plaintext files',
           ' If the file contains special formatting or is not a plaintext file, \nit cannot be read in Notepad',
           ' The image is a small example of what the Notepad may look like while running']
```

```
In [149]: sent_text2
```

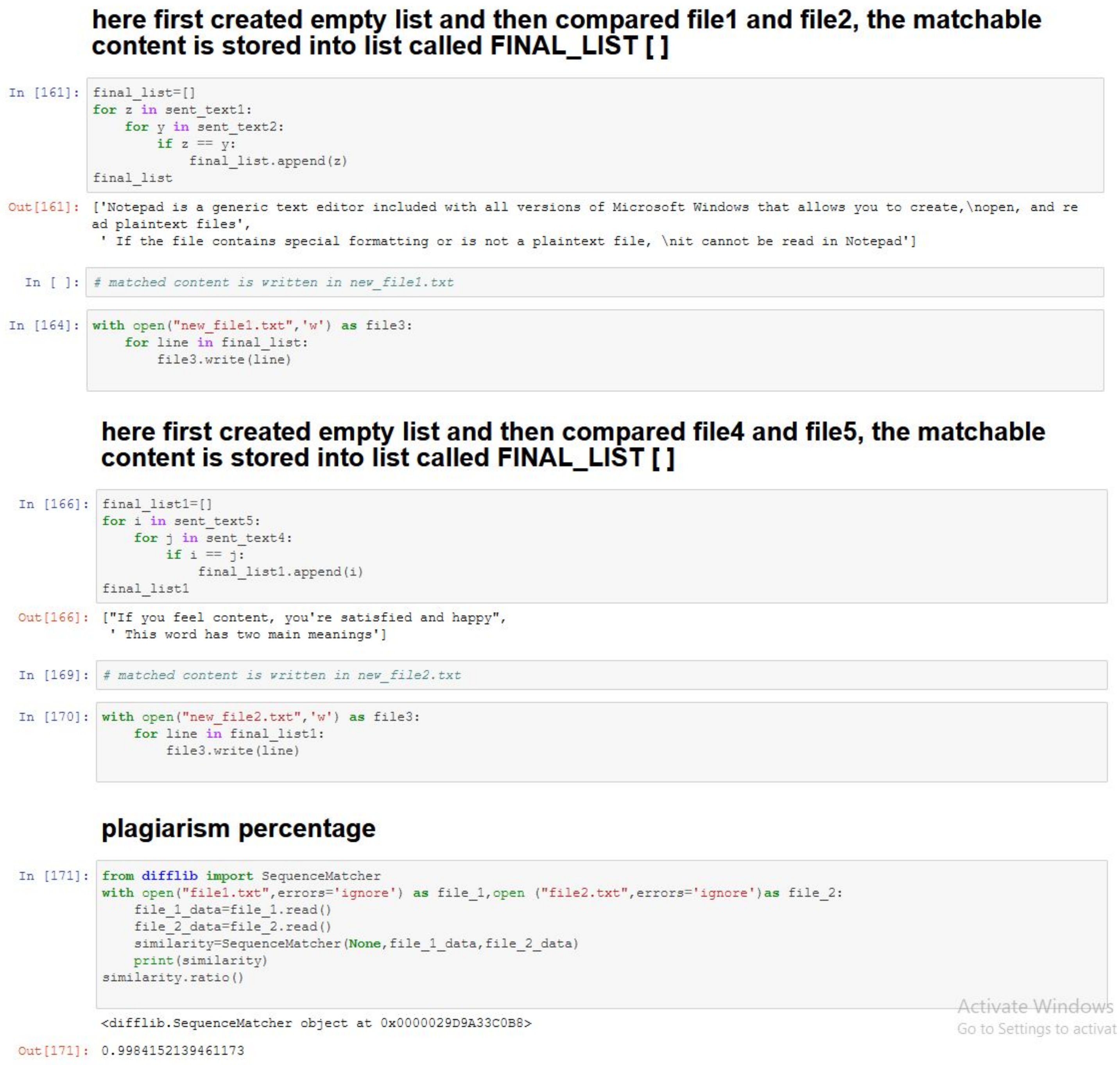
```
Out[149]: ['Notepad is a generic text editor included with all versions of Microsoft Windows that allows you to create,\nopen, and re
          ad plaintext files',
           ' If the file contains special formatting or is not a plaintext file, \nit cannot be read in Notepad',
           ' The image is a small example of what the Notepad may look like while running\n']
```

```
In [150]: sent_text4
```

```
Out[150]: ["If you feel content, you're satisfied and happy",
           " The content of a book, movie, or song \nis what it's about: the topic",
           ' This word has two main meanings',
           ' The first has to do with \nbeing pleased and satisfied (feeling content) or making someone else feel happy and at \npeac
          e with things (contenting them)',
           '']
```

```
In [151]: sent_text5
```

```
Out[151]: ["If you feel content, you're satisfied and happy",
           " The content of a book, movie, or song is what it's about: \nthe topic",
           ' This word has two main meanings',
           ' The first has to do with being pleased\n']
```



here first created empty list and then compared file1 and file2, the matchable content is stored into list called FINAL_LIST [ ]

```
In [161]: final_list=[]
          for z in sent_text1:
              for y in sent_text2:
                  if z == y:
                      final_list.append(z)
          final_list

Out[161]: ['Notepad is a generic text editor included with all versions of Microsoft Windows that allows you to create,\nopen, and re
          ad plaintext files',
           ' If the file contains special formatting or is not a plaintext file, \nit cannot be read in Notepad']
```

```
In [ ]: # matched content is written in new_file1.txt
```

```
In [164]: with open("new_file1.txt",'w') as file3:
              for line in final_list:
                  file3.write(line)
```

here first created empty list and then compared file4 and file5, the matchable content is stored into list called FINAL_LIST [ ]

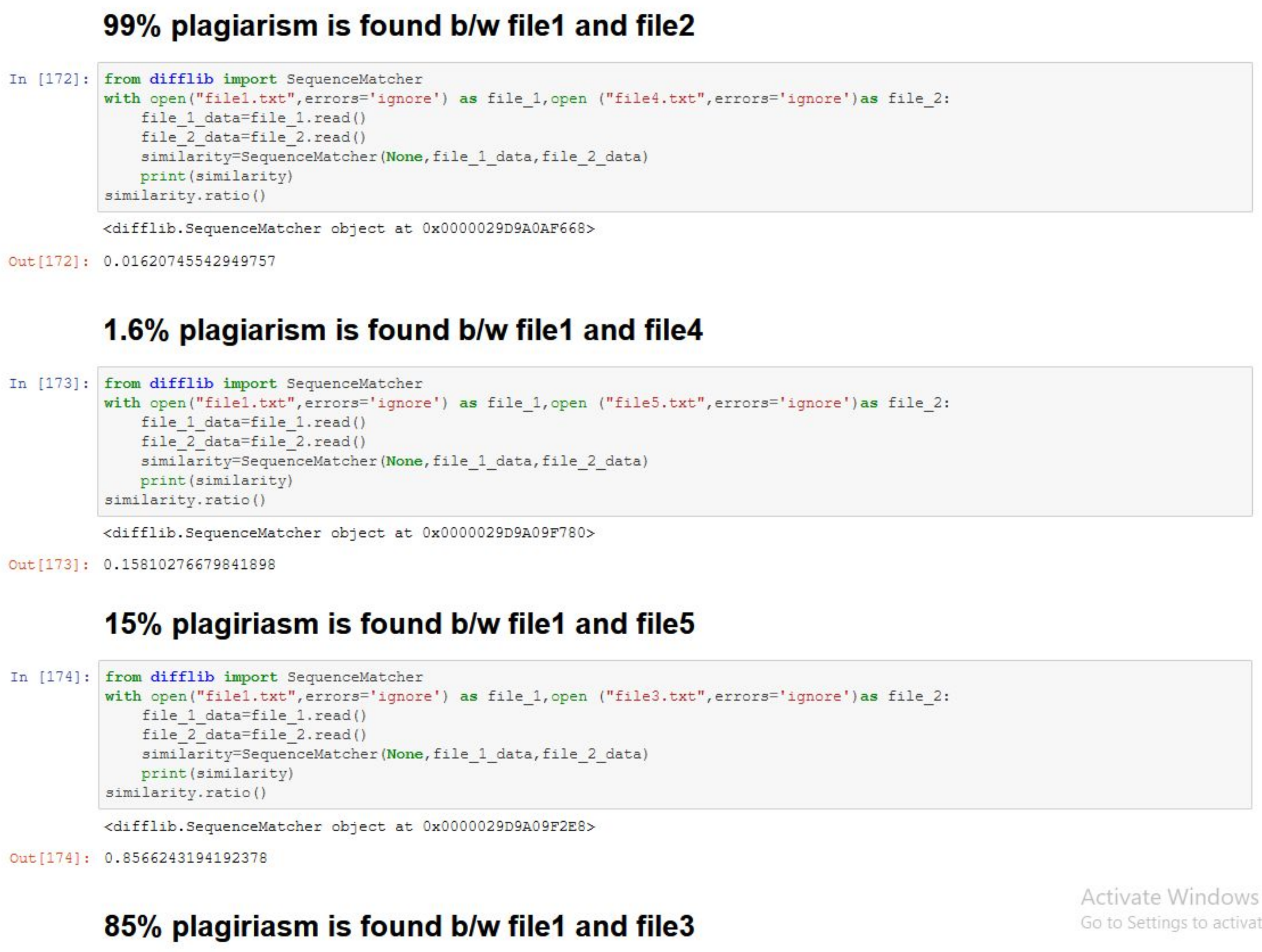
```
In [166]: final_list1=[]
          for i in sent_text5:
              for j in sent_text4:
                  if i == j:
                      final_list1.append(i)
          final_list1

Out[166]: ["If you feel content, you're satisfied and happy",
           ' This word has two main meanings']
```

```
In [169]: # matched content is written in new_file2.txt
```

```
In [170]: with open("new_file2.txt",'w') as file3:
              for line in final_list1:
                  file3.write(line)
```

plagiarism percentage

```
In [171]: from difflib import SequenceMatcher
          with open("file1.txt",errors='ignore') as file_1,open ("file2.txt",errors='ignore')as file_2:
              file_1_data=file_1.read()
              file_2_data=file_2.read()
              similarity=SequenceMatcher(None,file_1_data,file_2_data)
              print(similarity)
          similarity.ratio()

          <difflib.SequenceMatcher object at 0x0000029D9A33C0B8>

Out[171]: 0.9984152139461173
```



99% plagiarism is found b/w file1 and file2

```
In [172]: from difflib import SequenceMatcher
          with open("file1.txt",errors='ignore') as file_1,open ("file4.txt",errors='ignore')as file_2:
              file_1_data=file_1.read()
              file_2_data=file_2.read()
              similarity=SequenceMatcher(None,file_1_data,file_2_data)
              print(similarity)
          similarity.ratio()

          <difflib.SequenceMatcher object at 0x0000029D9A0AF668>

Out[172]: 0.01620745542949757
```

1.6% plagiarism is found b/w file1 and file4

```
In [173]: from difflib import SequenceMatcher
          with open("file1.txt",errors='ignore') as file_1,open ("file5.txt",errors='ignore')as file_2:
              file_1_data=file_1.read()
              file_2_data=file_2.read()
              similarity=SequenceMatcher(None,file_1_data,file_2_data)
              print(similarity)
          similarity.ratio()

          <difflib.SequenceMatcher object at 0x0000029D9A09F780>

Out[173]: 0.15810276679841898
```

15% plagiriasm is found b/w file1 and file5

```
In [174]: from difflib import SequenceMatcher
          with open("file1.txt",errors='ignore') as file_1,open ("file3.txt",errors='ignore')as file_2:
              file_1_data=file_1.read()
              file_2_data=file_2.read()
              similarity=SequenceMatcher(None,file_1_data,file_2_data)
              print(similarity)
          similarity.ratio()

          <difflib.SequenceMatcher object at 0x0000029D9A09F2E8>

Out[174]: 0.8566243194192378
```

85% plagiriasm is found b/w file1 and file3

Conclusion:

Plagiarism detection process is very important as the trend of copying someone else's work is increasing these days. Different data mining techniques can be used to enhance the plagiarism detection process. Here, we used a simple technique through which it is thought that the process efficiency can be improved. Pre-processing and further processing techniques can be used to decrease the overhead of the process. Moreover, similarity score can be calculated through separated plagiarized data so that efficiency can be improved.