ARYAMAN MISHRA

19BCE1027

Part A:

You are given 9 one-line documents here. Consider the following keywords to represent the documents in the vector space model:

[1] Automotive [2] Car [3] motorcycles [4] self-drive [5] IoT [6] hire [7] Dhoni
Represent the documents in vector space Model using these keywords and use it as input to cluster the documents using Manhattan distance as parameter. Ignore case differences.

You need to do hierarchical clustering with single-link, complete-link, average-link agglomerative clustering.

Documents for use in question

Doc1

Electric automotive maker Tesla Inc. is likely to introduce its products in India sometime in the summer of 2017.

Doc 2

Automotive major Mahindra likely to introduce driverless cars

Doc 3

BMW plans to introduce its own motorcycles in india

Doc 4

Just drive, a self-drive car rental firm uses smart vehicle technology based on IoT

Doc 5

Automotive industry going to hire thousands in 2018

Doc 6

Famous cricket player Dhoni brought his priced car Hummer which is an SUV

Doc 7

Dhoni led india to its second world cup victory

Doc 8

IoT in cars will lead to more safety and make driverless vehicle revolution possible

Doc 9

Sachin recommended Dhoni for the indian skipper post

Data Structure Proposed: Dictionaries.

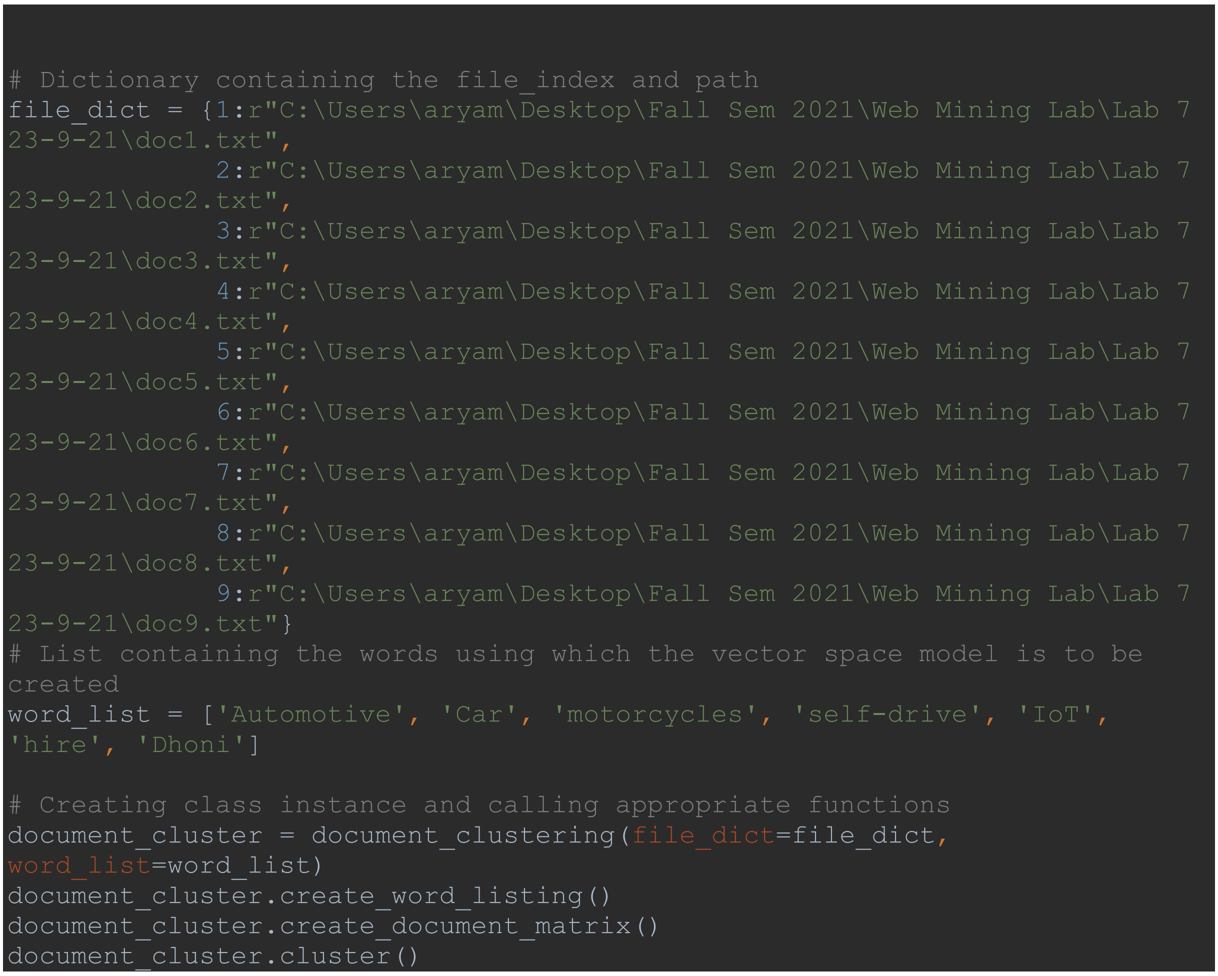
ALGORITHM:

- **Step-1:**
  Consider each alphabet as a single cluster and calculate the distance of one cluster from all the other clusters.
- **Step-2:**
  In the second step comparable clusters are merged together to form a single cluster. Let’s say cluster (B) and cluster (C) are very similar to each other therefore we merge them in the second step similarly with cluster (D) and (E) and at last, we get the clusters [(A), (BC), (DE), (F)]
- **Step-3:**
  We recalculate the proximity according to the algorithm and merge the two nearest clusters([(DE), (F)]) together to form new clusters as [(A), (BC), (DEF)]
- **Step-4:**
  Repeating the same process; The clusters DEF and BC are comparable and merged together to form a new cluster. We’re now left with clusters [(A), (BCDEF)].
- **Step-5:**
  At last the two remaining clusters are merged together to form a single cluster [(ABCDEF)].

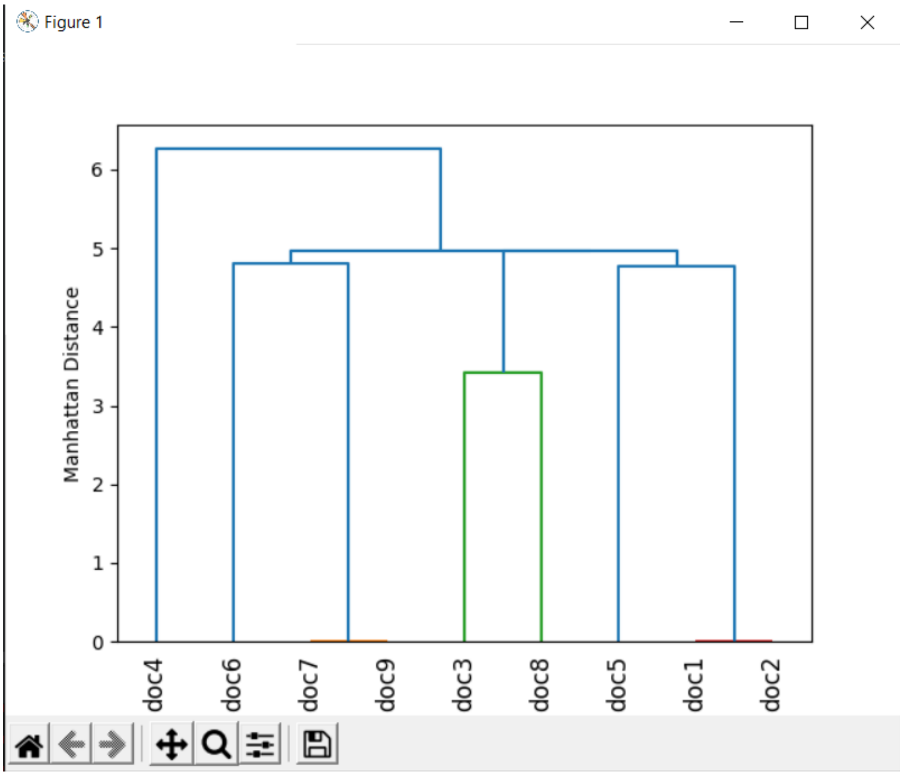
IMPLEMENTATION CODE AND RESULTS:

single-link

```
# Importing the libraries
import string
import pandas as pd
import math
import matplotlib.pyplot as plt


class document_clustering(object):

    def __init__(self, file_dict, word_list):
        self.file_dict = file_dict
        self.word_list = word_list
    def tokenize_document(self, document):
        """Returns a list of words contained in the document after 
converting
        it to lowercase and striping punctuation marks"""
        terms = document.lower().split()
        return [term.strip(string.punctuation) for term in terms]

    def create_word_listing(self):
        """Function to create the word listing of the objects"""

        # Dictionary to hold the frequency of words in word_list with 
file_index as key
        self.listing_dict_ = {}

        for id in self.file_dict:
            temp_word_list = []
            f = open(self.file_dict[id], 'r')
            document = f.read()
            terms = self.tokenize_document(document)
            for term in self.word_list:
                temp_word_list.append(terms.count(term.lower()))
            self.listing_dict_[id] = temp_word_list

        print('Word listing of each document')
        for id in self.listing_dict_:
            print('%d:  %s' % (id, self.listing_dict_[id]))

    def create_document_matrix(self):
        """Function to create the document distance matrix"""
        self.labels_ = ['doc%d' % (id) for id in self.file_dict]
        main_list = []
        for id1 in self.file_dict:
            temp_list = []
            for id2 in self.file_dict:
                dist = 0
                for term1, term2 in zip(self.listing_dict_[id1], 
self.listing_dict_[id2]):
                    dist += abs(term1 - term2)
                temp_list.append(round(math.sqrt(dist), 4))
            main_list.append(temp_list)

        self.distance_matrix_ = pd.DataFrame(main_list, index=self.labels_, 
columns=self.labels_)
        print('\nDistance Matrix')
        print(self.distance_matrix_)

    def cluster(self):
        """Create the vector space model from the documents. Perform 
Hierarchical
        Clustering"""
        from scipy.cluster.hierarchy import linkage
        row_cluster = linkage(self.distance_matrix_.values,
                              method='single',
                              metric='cityblock')
        from scipy.cluster.hierarchy import dendrogram
        dn = dendrogram(row_cluster, labels=self.labels_)
        plt.ylabel('Manhattan Distance')
        plt.xticks(rotation=90)
        plt.savefig('dendrogram1.png', dpi=300)
        plt.show()
```

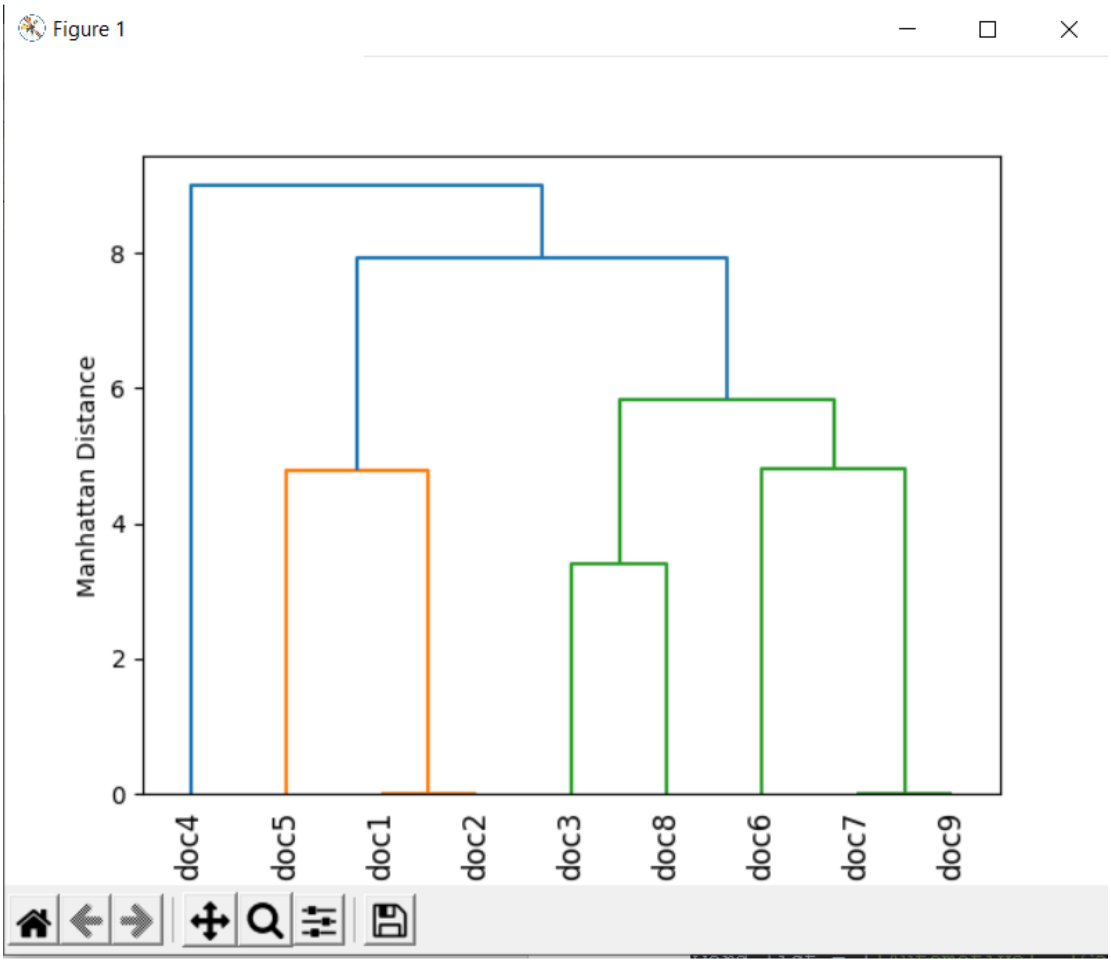


```
# Dictionary containing the file_index and path
file_dict = {1:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc1.txt",
             2:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc2.txt",
             3:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc3.txt",
             4:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc4.txt",
             5:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc5.txt",
             6:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc6.txt",
             7:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc7.txt",
             8:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc8.txt",
             9:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc9.txt"}
# List containing the words using which the vector space model is to be
created
word_list = ['Automotive', 'Car', 'motorcycles', 'self-drive', 'IoT',
'hire', 'Dhoni']

# Creating class instance and calling appropriate functions
document_cluster = document_clustering(file_dict=file_dict,
word_list=word_list)
document_cluster.create_word_listing()
document_cluster.create_document_matrix()
document_cluster.cluster()
```

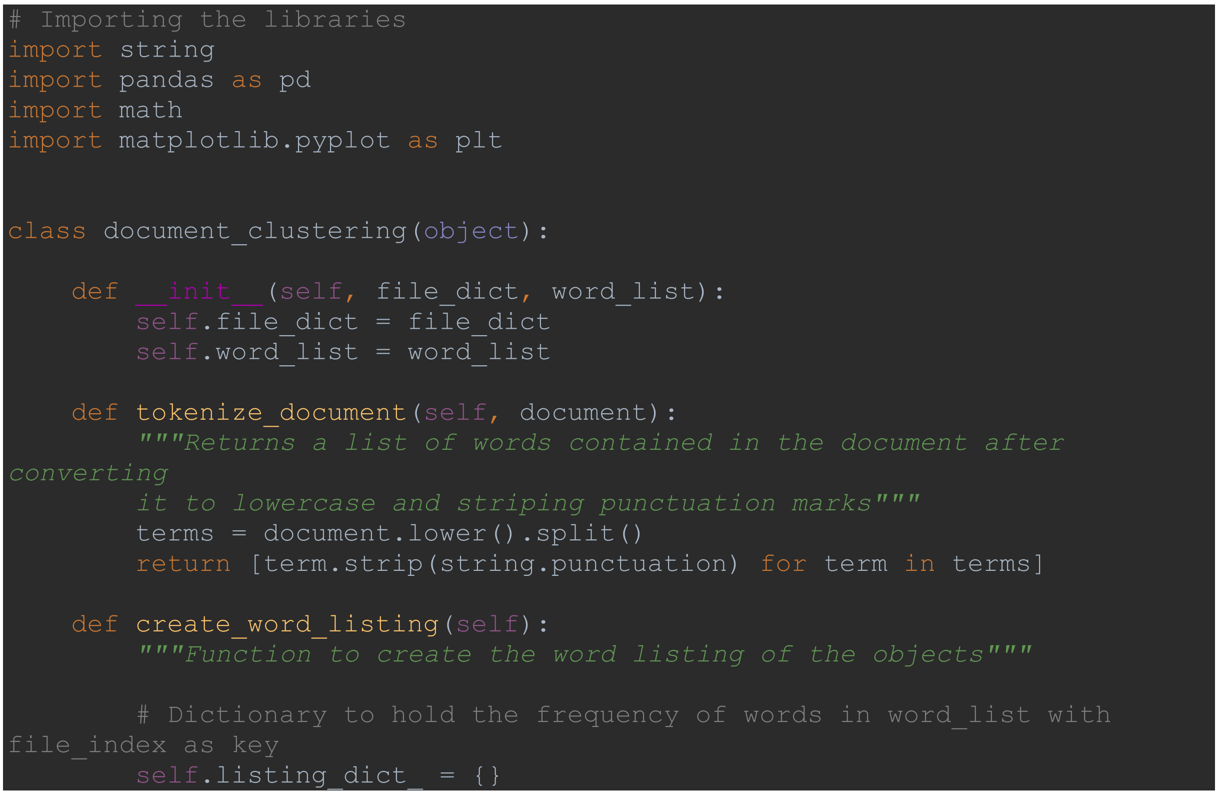


complete-link

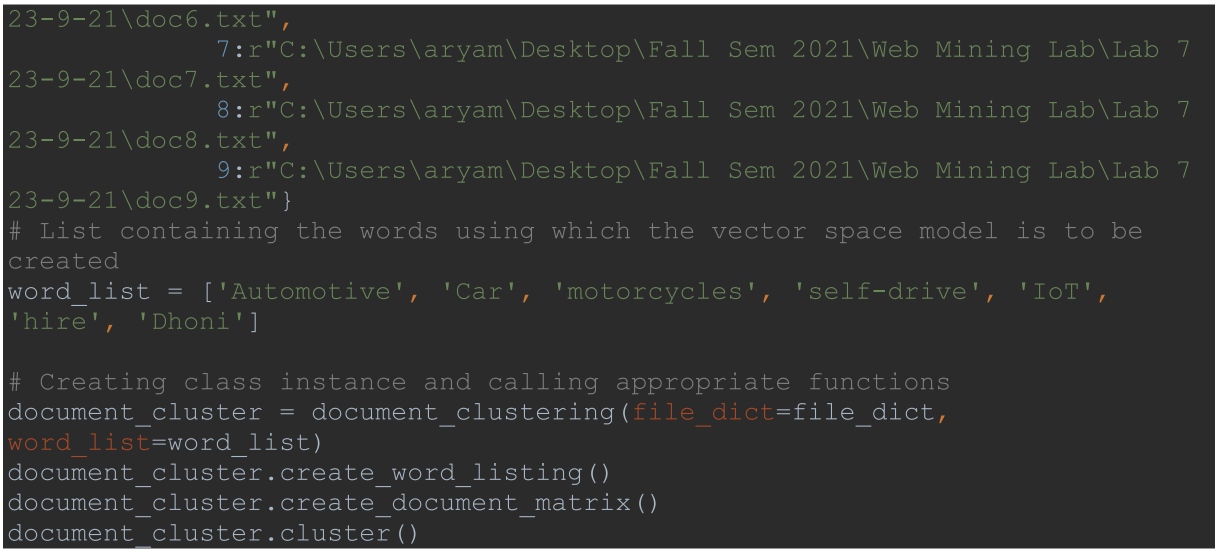
```
# Importing the libraries
import string
import pandas as pd
import math
import matplotlib.pyplot as plt


class document_clustering(object):

    def __init__(self, file_dict, word_list):
        self.file_dict = file_dict
        self.word_list = word_list

    def tokenize_document(self, document):
        """Returns a list of words contained in the document after
converting
        it to lowercase and striping punctuation marks"""
        terms = document.lower().split()
        return [term.strip(string.punctuation) for term in terms]

    def create_word_listing(self):
        """Function to create the word listing of the objects"""

        # Dictionary to hold the frequency of words in word_list with
file_index as key
        self.listing_dict_ = {}

        for id in self.file_dict:
            temp_word_list = []
            f = open(self.file_dict[id], 'r')
            document = f.read()
            terms = self.tokenize_document(document)
            for term in self.word_list:
                temp_word_list.append(terms.count(term.lower()))
            self.listing_dict_[id] = temp_word_list

        print('Word listing of each document')
        for id in self.listing_dict_:
            print('%d:  %s' % (id, self.listing_dict_[id]))

    def create_document_matrix(self):
        """Function to create the document distance matrix"""
        self.labels_ = ['doc%d' % (id) for id in self.file_dict]
        main_list = []
        for id1 in self.file_dict:
            temp_list = []
            for id2 in self.file_dict:
                dist = 0
                for term1, term2 in zip(self.listing_dict_[id1],
self.listing_dict_[id2]):
                    dist += abs(term1 - term2)
                temp_list.append(round(math.sqrt(dist), 4))
            main_list.append(temp_list)

        self.distance_matrix_ = pd.DataFrame(main_list, index=self.labels_,
columns=self.labels_)
        print('\nDistance Matrix')
        print(self.distance_matrix_)
    def cluster(self):
        """Create the vector space model from the documents. Perform
Hierarchical
        Clustering"""
        from scipy.cluster.hierarchy import linkage
        row_cluster = linkage(self.distance_matrix_.values,
                              method='complete',
                              metric='cityblock')
        from scipy.cluster.hierarchy import dendrogram
        dn = dendrogram(row_cluster, labels=self.labels_)
        plt.ylabel('Manhattan Distance')
        plt.xticks(rotation=90)
        plt.savefig('dendrogram1.png', dpi=300)
        plt.show()


# Dictionary containing the file_index and path
file_dict = {1:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc1.txt",
             2:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc2.txt",
             3:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc3.txt",
             4:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc4.txt",
             5:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc5.txt",
             6:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc6.txt",
             7:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc7.txt",
             8:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc8.txt",
             9:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc9.txt"}
# List containing the words using which the vector space model is to be
created
word_list = ['Automotive', 'Car', 'motorcycles', 'self-drive', 'IoT',
'hire', 'Dhoni']

# Creating class instance and calling appropriate functions
document_cluster = document_clustering(file_dict=file_dict,
word_list=word_list)
document_cluster.create_word_listing()
document_cluster.create_document_matrix()
document_cluster.cluster()
```



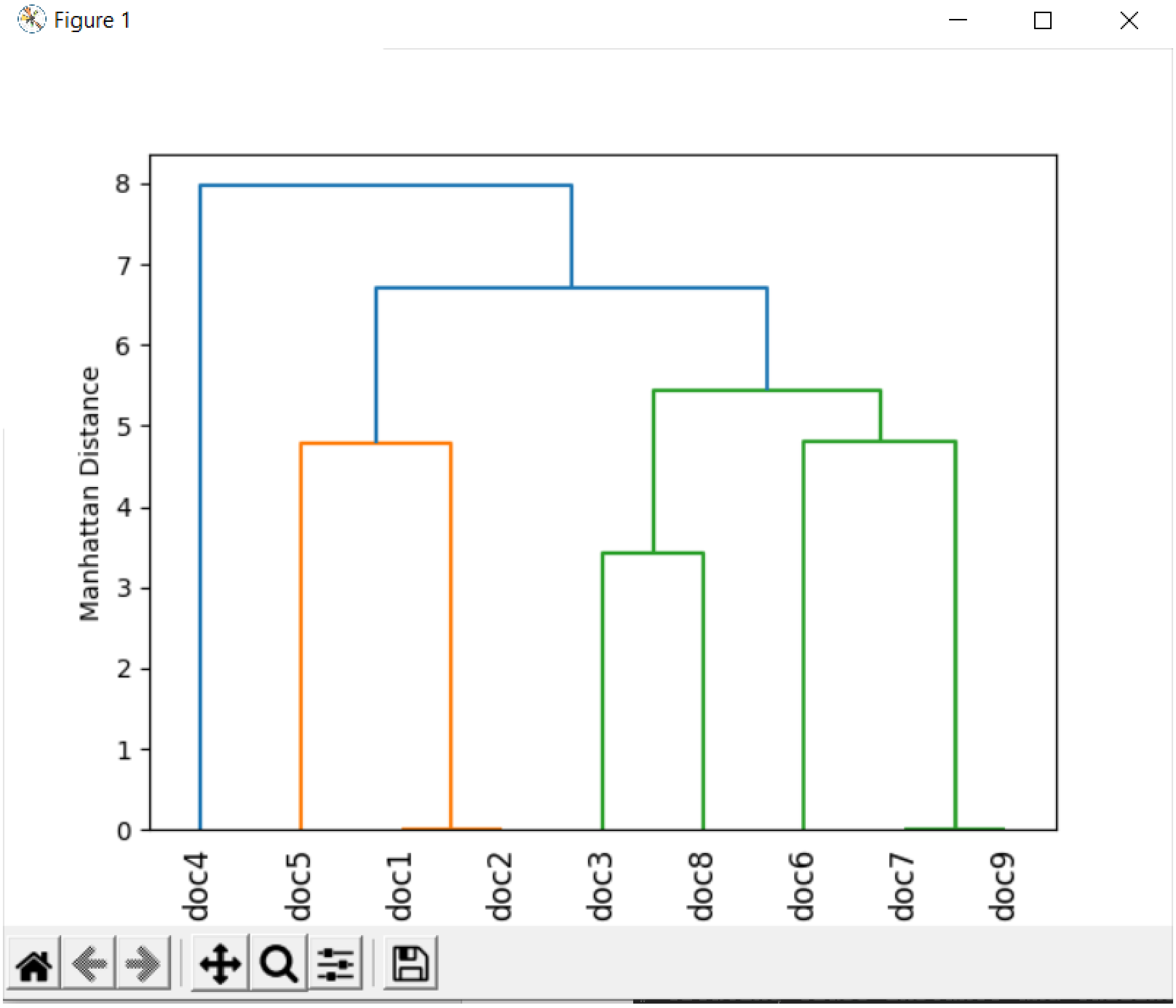
average-link



```
# Importing the libraries
import string
import pandas as pd
import math
import matplotlib.pyplot as plt


class document_clustering(object):

    def __init__(self, file_dict, word_list):
        self.file_dict = file_dict
        self.word_list = word_list

    def tokenize_document(self, document):
        """Returns a list of words contained in the document after
converting
        it to lowercase and striping punctuation marks"""
        terms = document.lower().split()
        return [term.strip(string.punctuation) for term in terms]

    def create_word_listing(self):
        """Function to create the word listing of the objects"""

        # Dictionary to hold the frequency of words in word_list with
file_index as key
        self.listing_dict_ = {}
        for id in self.file_dict:
            temp_word_list = []
            f = open(self.file_dict[id], 'r')
            document = f.read()
            terms = self.tokenize_document(document)
            for term in self.word_list:
                temp_word_list.append(terms.count(term.lower()))
            self.listing_dict_[id] = temp_word_list

        print('Word listing of each document')
        for id in self.listing_dict_:
            print('%d:  %s' % (id, self.listing_dict_[id]))

    def create_document_matrix(self):
        """Function to create the document distance matrix"""
        self.labels_ = ['doc%d' % (id) for id in self.file_dict]
        main_list = []
        for id1 in self.file_dict:
            temp_list = []
            for id2 in self.file_dict:
                dist = 0
                for term1, term2 in zip(self.listing_dict_[id1],
self.listing_dict_[id2]):
                    dist += abs(term1 - term2)
                temp_list.append(round(math.sqrt(dist), 4))
            main_list.append(temp_list)

        self.distance_matrix_ = pd.DataFrame(main_list, index=self.labels_,
columns=self.labels_)
        print('\nDistance Matrix')
        print(self.distance_matrix_)

    def cluster(self):
        """Create the vector space model from the documents. Perform
Hierarchical
        Clustering"""
        from scipy.cluster.hierarchy import linkage
        row_cluster = linkage(self.distance_matrix_.values,
                              method='average',
                              metric='cityblock')
        from scipy.cluster.hierarchy import dendrogram
        dn = dendrogram(row_cluster, labels=self.labels_)
        plt.ylabel('Manhattan Distance')
        plt.xticks(rotation=90)
        plt.savefig('dendrogram1.png', dpi=300)
        plt.show()


# Dictionary containing the file_index and path
file_dict = {1:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc1.txt",
             2:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc2.txt",
             3:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc3.txt",
             4:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc4.txt",
             5:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc5.txt",
             6:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
```



```
23-9-21\doc6.txt",
             7:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc7.txt",
             8:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc8.txt",
             9:r"C:\Users\aryam\Desktop\Fall Sem 2021\Web Mining Lab\Lab 7
23-9-21\doc9.txt"}
# List containing the words using which the vector space model is to be
created
word_list = ['Automotive', 'Car', 'motorcycles', 'self-drive', 'IoT',
'hire', 'Dhoni']

# Creating class instance and calling appropriate functions
document_cluster = document_clustering(file_dict=file_dict,
word_list=word_list)
document_cluster.create_word_listing()
document_cluster.create_document_matrix()
document_cluster.cluster()
```



Part B

Use the same program which you have developed for part A to do "hierarchical clustering" of the following web documents. Use the keywords.

[1] Tesla [2] Electric [3] Car/Vehicle/Automobile [4] pollution [5] de-monetisation [6] GST [7] black money

Download the webpage into a .txt file [ignore images, tables and limit the size of the document to 500 words Max] and build your vector space model using Term frequency.

Ignore case differences. Treat singular and plural of nouns as same. Treat Car/vehicle/automobile as one word [synonyms]. Treat "black money" as a single word.

List of webpages:

[1] https://www.zigwheels.com/newcars/Tesla

[2] https://www.financialexpress.com/auto/car-news/mahindra-to-launch-indias-first-electric-suv-in-2019-all-new-e-verito-sedan-on-cards/1266853/

[3] https://en.wikipedia.org/wiki/Toyota_Prius

[4] https://economictimes.indiatimes.com/industry/auto/auto-news/government-plans-new-policy-to-promote-electric-vehicles/articleshow/65237123.cms

[5] https://indianexpress.com/article/india/india-news-india/demonetisation-hits-electric-vehicles-industry-society-of-manufacturers-of-electric-vehicles-4395104/

[6] https://www.livemint.com/Politics/ySbMKTIC4MINsz1btccBJO/How-demonetisation-affected-the-Indian-economy-in-10-charts.html

[7] https://www.hrblock.in/blog/impact-gst-automobile-industry-2/

[8] https://inc42.com/buzz/electric-vehicles-this-week-centre-reduces-gst-on-lithium-ion-batteries-hyundai-to-launch-electric-suv-in-india-and-more/

[9] https://www.youthkiawaaz.com/2017/12/impact-of-demonetisation-on-the-indian-economy/

[10] https://indianexpress.com/article/india/demonetisation-effects-cash-crisis-mobile-wallets-internet-banking-4406005/

[11] https://www.news18.com/news/business/how-gst-will-curb-tax-evasion-1446035.html

[12] https://economictimes.indiatimes.com/small-biz/policy-trends/is-gst-helping-the-indian-economy-for-the-better/articleshow/65319874.cms

Data Structure Proposed: Dictionaries.

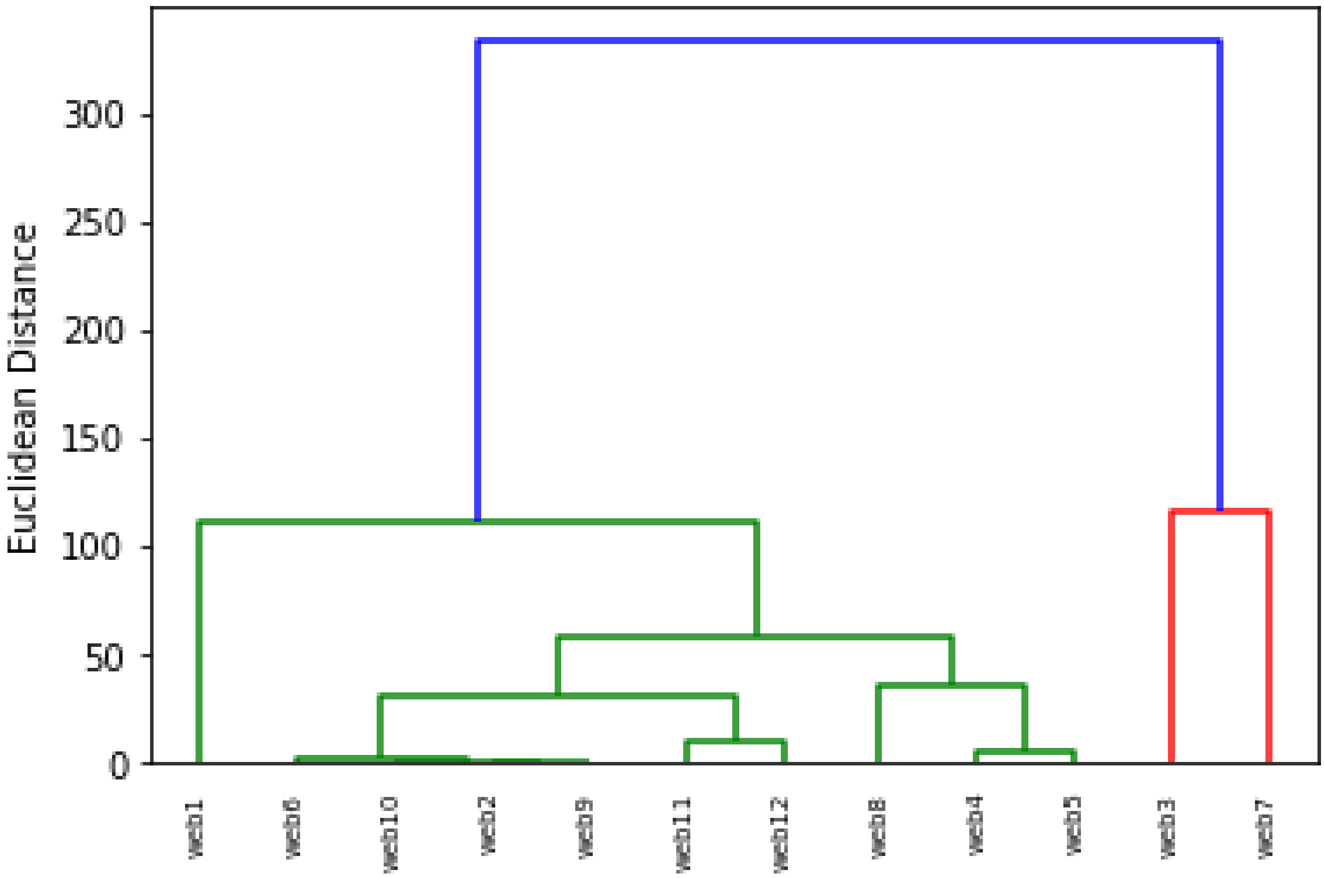
ALGORITHM:

- **Step-1:**
  Consider each alphabet as a single cluster and calculate the distance of one cluster from all the other clusters.
- **Step-2:**
  In the second step comparable clusters are merged together to form a single cluster. Let’s say cluster (B) and cluster (C) are very similar to each other therefore we merge them in the second step similarly with cluster (D) and (E) and at last, we get the clusters [(A), (BC), (DE), (F)]
- **Step-3:**
  We recalculate the proximity according to the algorithm and merge the two nearest clusters([(DE), (F)]) together to form new clusters as [(A), (BC), (DEF)]
- **Step-4:**
  Repeating the same process; The clusters DEF and BC are comparable and merged together to form a new cluster. We’re now left with clusters [(A), (BCDEF)].
- **Step-5:**
  At last the two remaining clusters are merged together to form a single cluster [(ABCDEF)].

IMPLEMENTATION CODE AND RESULTS:

```
import string
import pandas as pd
import math
import matplotlib.pyplot as plt
import requests
import re
from bs4 import BeautifulSoup
from bs4.element import Comment
from nltk.stem import PorterStemmer

# Function to filter the HTML tags and text
def visible_text(element):
    if element.parent.name in ['style', 'title', 'script', 'head', '[document]', 'class', 'a', 'li']:
        return False
    elif isinstance(element, Comment):
        return False
    elif re.match(r"[\s\r\n]+",str(element)):
        return False
    elif re.match(r"www.", str(element)):
        return False
    return True
```

```
class document_clustering(object):
    """Implementing the document clustering class.
    It creates the vector space model of the passed documents and then
    creates a Hierarchical Cluster to organize them.

    Parameters:
    -------------
    file_dict: dictionary
        Contains the path of the different files to be read.
        Format: {file_index: path}
    word_list: list
        Contains the list of words using which the vector space model i
s to be
        created.

    Attributes:
    -----------
    listing_dict_: dictionary
        Contains the frequency of the words in each document as file_in
dex as key
        and frequency list as value.
    distance_matrix_ : pandas-dataframe
        Contains the sqaure matrix of documents containing the pairwise
 distance between them
    labels_: list
        Contains the labels for document names
    """

    def __init__(self, file_dict, word_list):
        self.file_dict = file_dict
        self.word_list = word_list

    def tokenize_document(self, document):
        """Returns a list of words contained in the document after conv
erting
        it to lowercase and striping punctuation marks"""
        ps = PorterStemmer()
        terms = []
        for i in document:
            temp = i.lower().replace('vehicle', 'car').replace('automob
ile', 'car').split()
            for j in temp:
                terms.append(j)
        return [ps.stem(term.strip(string.punctuation)) for term in ter
ms]

    def create_word_listing(self):
        """Function to create the word listing of the objects"""
        # Dictionary to hold the frequency of words in word_list with f
ile_index as key
        self.listing_dict_ = {}

        for id in self.file_dict:
            temp_word_list = []
            response = requests.get(self.file_dict[id])
            soup = BeautifulSoup(response.text, 'html.parser')
            text = soup.find_all(text = True)
            text = list(filter(visible_text, text))
            terms = self.tokenize_document(text)
            for term in self.word_list[:500]:
                temp_word_list.append(terms.count(term.lower()))
            self.listing_dict_[id] = temp_word_list

        print('Word listing of each document')
        for id in self.listing_dict_:
            print('%d:  %s' % (id, self.listing_dict_[id]))

    def create_document_matrix(self):
        """Function to create the document distance matrix"""
        self.labels_ = ['web%d' % (id) for id in self.file_dict]
        main_list = []
        for id1 in self.file_dict:
            temp_list = []
            for id2 in self.file_dict:
                dist = 0
                for term1, term2 in zip(self.listing_dict_[id1], self.l
isting_dict_[id2]):
                    dist += (term1-term2)**2
                temp_list.append(round(math.sqrt(dist), 4))
            main_list.append(temp_list)

        self.distance_matrix_ = pd.DataFrame(main_list, index = self.la
bels_, columns = self.labels_)
        print('\nDistance Matrix')
        print(self.distance_matrix_)

    def cluster(self):
        """Create the vector space model from the documents. Perform Hi
erarchical
        Clustering"""
        from scipy.cluster.hierarchy import linkage
        row_cluster = linkage(self.distance_matrix_.values,
                              method = 'complete',
                              metric = 'euclidean')
        from scipy.cluster.hierarchy import dendrogram
        dn = dendrogram(row_cluster, labels = self.labels_)
        plt.ylabel('Euclidean Distance')
        plt.xticks(rotation = 90, fontsize = 7)
        plt.savefig('dendrogram2.png', dpi = 300)
        plt.show()


# Dictionary containing the file_index and path
file_dict = {1: 'https://www.zigwheels.com/newcars/Tesla',
             2: 'https://www.financialexpress.com/auto/car-
news/mahindra-to-launch-indias-first-electric-suv-in-2019-all-new-e-
verito-sedan-on-cards/1266853/',
             3: 'https://en.wikipedia.org/wiki/Toyota_Prius',
             4: 'https://economictimes.indiatimes.com/industry/auto/aut
o-news/government-plans-new-policy-to-promote-electric-
vehicles/articleshow/65237123.cms',
             5: 'https://indianexpress.com/article/india/india-news-
india/demonetisation-hits-electric-vehicles-industry-society-of-
manufacturers-of-electric-vehicles-4395104/',
             6: 'https://www.livemint.com/Politics/ySbMKTIC4MINsz1btccB
JO/How-demonetisation-affected-the-Indian-economy-in-10-charts.html',
             7: 'https://www.researchgate.net/publication/348959791_Imp
act_of_GST_on_Automobile_Industry_in_India',
             8: 'https://inc42.com/buzz/electric-vehicles-this-week-
centre-reduces-gst-on-lithium-ion-batteries-hyundai-to-launch-electric-
suv-in-india-and-more/',
             9: 'https://www.youthkiawaaz.com/2017/12/impact-of-
demonetisation-on-the-indian-economy/',
             10:'https://indianexpress.com/article/india/demonetisation
-effects-cash-crisis-mobile-wallets-internet-banking-4406005/',
             11: 'https://www.news18.com/news/business/how-gst-will-
curb-tax-evasion-1446035.html',
             12: 'https://economictimes.indiatimes.com/small-
biz/policy-trends/is-gst-helping-the-indian-economy-for-the-
better/articleshow/65319874.cms'}




# List containing the words using which the vector space model is to be
 created
word_list = ['Tesla', 'Electric', 'Car', 'pollution', 'de-
monetisation', 'GST' ,'black money']

# Creating class instance and calling appropriate functions
document_cluster = document_clustering(file_dict = file_dict, word_list
 = word_list)
document_cluster.create_word_listing()
document_cluster.create_document_matrix()
document_cluster.cluster()
Word listing of each document
1:  [27, 0, 31, 0, 0, 0, 0]
2:  [0, 0, 0, 0, 0, 0, 0]
3:  [0, 0, 97, 0, 0, 0, 0]
4:  [0, 0, 12, 0, 0, 0, 0]
5:  [0, 0, 10, 0, 0, 0, 0]
6:  [0, 0, 1, 0, 0, 0, 0]
7:  [0, 0, 89, 0, 0, 72, 0]
8:  [0, 0, 21, 0, 0, 6, 0]
9:  [0, 0, 0, 0, 0, 0, 0]
10:  [0, 0, 0, 0, 0, 0, 0]
11:  [0, 0, 0, 0, 0, 9, 0]
12:  [0, 0, 0, 0, 0, 13, 0]

Distance Matrix
          web1      web2     web3  ...     web10     web11     web12
web1    0.0000   41.1096  71.3092  ...   41.1096   42.0833   43.1161
web2   41.1096    0.0000  97.0000  ...    0.0000    9.0000   13.0000
web3   71.3092   97.0000   0.0000  ...   97.0000   97.4166   97.8673
web4   33.0151   12.0000  85.0000  ...   12.0000   15.0000   17.6918
web5   34.2053   10.0000  87.0000  ...   10.0000   13.4536   16.4012
web6   40.3609    1.0000  96.0000  ...    1.0000    9.0554   13.0384
web7   96.3172  114.4771  72.4431  ...  114.4771  109.0413  106.7801
web8   29.4109   21.8403  76.2365  ...   21.8403   21.2132   22.1359
web9   41.1096    0.0000  97.0000  ...    0.0000    9.0000   13.0000
web10  41.1096    0.0000  97.0000  ...    0.0000    9.0000   13.0000
web11  42.0833    9.0000  97.4166  ...    9.0000    0.0000    4.0000
web12  43.1161   13.0000  97.8673  ...   13.0000    4.0000    0.0000

[12 rows x 12 columns]
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:107:
ClusterWarning: scipy.cluster: The symmetric non-negative hollow
observation matrix looks suspiciously like an uncondensed distance matrix
```



CONCLUSIONALL TASKS HAVE BEEN SUCCESFULLY IMPLEMENTED AND EXECUTED.