**WEB MINING**

**HIERARCHICAL CLUSTERING**

**16BCE1186**

**HARSH TRIPATHI**

**LINKS CLUSTERINGS**

**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 is to be**

**created.**

**Attributes:**

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

**listing\_dict\_: dictionary**

**Contains the frequency of the words in each document as file\_index 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 converting**

**it to lowercase and striping punctuation marks"""**

**ps = PorterStemmer()**

**terms = []**

**for i in document:**

**temp = i.lower().replace('vehicle', 'car').replace('automobile', 'car').split()**

**for j in temp:**

**terms.append(j)**

**return [ps.stem(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 = []**

**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.listing\_dict\_[id2]):**

**dist += abs(term1-term2)**

**temp\_list.append(dist)**

**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 = 'cosine')**

**from scipy.cluster.hierarchy import dendrogram**

**dn = dendrogram(row\_cluster, labels = self.labels\_)**

**plt.ylabel('cosine 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/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'}**

**# 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**

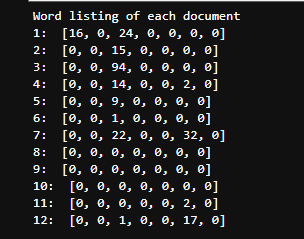
**document\_cluster = document\_clustering(file\_dict = file\_dict, word\_list = word\_list)**

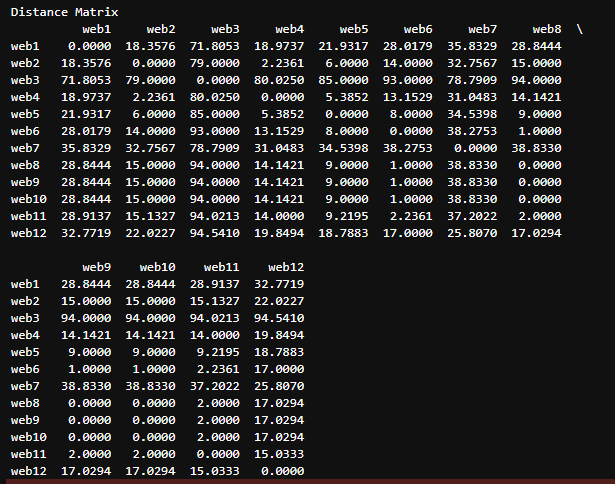
**document\_cluster.create\_word\_listing()**

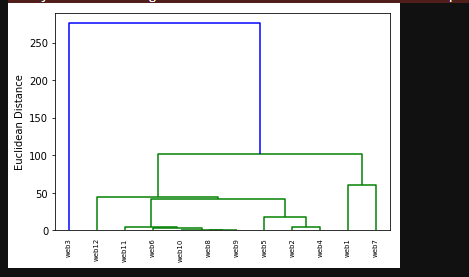
**document\_cluster.create\_document\_matrix()**

**document\_cluster.cluster()**

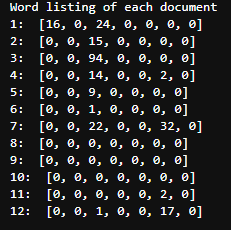
**EUCLEDIAN DISTANCE**

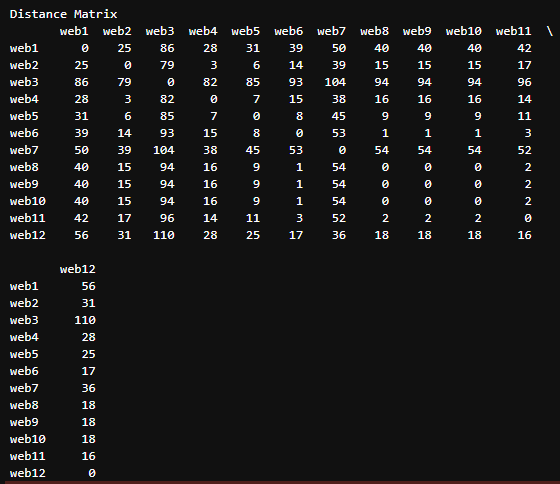


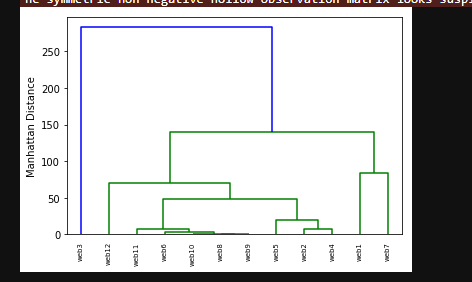




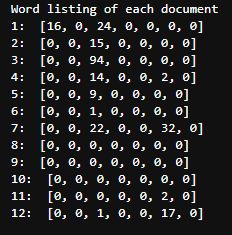
**MANHATTAN DISTANCE**

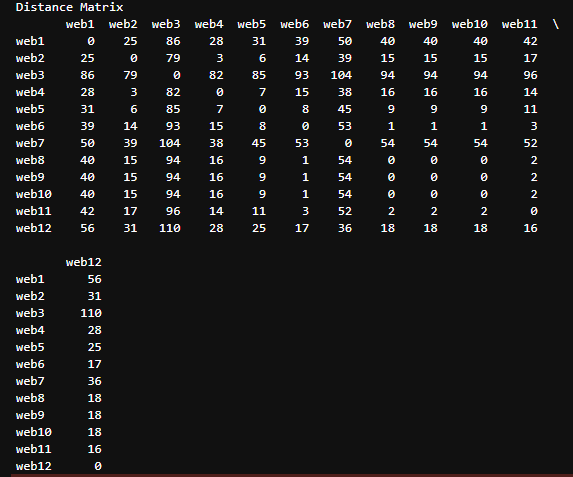
****

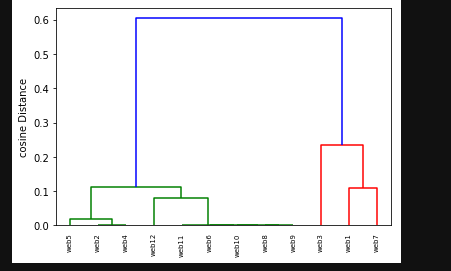
****

****

**COSINE DISTANCE**

****

****

****