Web Mining – Lab 5

Vector Space model based Clustering of documents

By Abhijeet Ambadekar (16BCE1156)

Problem statement:

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 Euclidean distance as parameter. Ignore case differences.

You also need to hierarchically organize the cluster by representing each identified cluster by a virtual document that is the "centroid" of the members of the cluster.

Part B -

Use the same program which you have developed for part A to do preliminary "nearest neighbourhood clustering" of the following web documents and also do hierarchical clustering. 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.

Program Code:

```
import numpy as np
from pprint import pprint
import requests
from bs4 import BeautifulSoup
import re
from math import inf

def get_web_text(url):
	try:
		page = requests.get(url)
	except Exception as e:
		try:
			print("Failed to reached {}".format(url))
		except UnicodeEncodeError:
			print("Failed to reached and cant show the URL")
		return None

	soup = BeautifulSoup(page.text, 'html.parser')

	# Removing style blocks
	[tag.decompose() for tag in soup("style")]

	# Removing scripts
	[tag.decompose() for tag in soup("script")]

	text = re.sub('\n+', ' ', soup.get_text()).strip().lower()
	return text


def get_word_count(words, text):
	if type(words) == str:
		return text.count(words)
	elif type(words) == list:
		return sum([text.count(wordi) for wordi in words])

def get_set_index(doc, set_list):
	for i, curr_set in enumerate(set_list):
		if doc in curr_set:
			return i


class VSCluster():
	def __init__(self, threshold):
		self.threshold = threshold
		self.doclist = {}
		self.wordmap = {}
		self.doclist = {}
		self.freq_matrix = []
		self.clusterlist = {}
```

```
	def get_text_data(self, docfile):
		with open(docfile) as fp:
			content = list(map(str.lower, fp.readlines()))

		for textline in content:
			docname, doctext = textline.split(" : ")
			self.doclist[docname] = doctext


	def get_URL_data(self, docfile):
		with open(docfile) as fp:
			urls = list(map(str.strip, fp.readlines()))

		for url in urls:
			self.doclist[url] = get_web_text(url)


	def get_hier_data(self, doclist=None, wordmap=None):
		if doclist == None or wordmap == None:
			print("Given empty doclist or wordmap. Can't fit it")
		else:
			self.doclist = doclist
			self.wordmap = wordmap


	def get_word_freq(self):
		for doc in self.doclist:
			self.wordmap[doc] = np.array([get_word_count(word,
self.doclist[doc]) for word in self.wordlist])

	def get_doc_vec(self, doc):
		return self.wordmap[doc]


	def fit(self, wordlist, docfile, input_type, wordmap=None):
		# self.wordlist = list(map(str.lower, wordlist))
		self.wordlist = wordlist
		self.input_type = input_type

		if input_type == "URL":
			self.get_URL_data(docfile)
			self.get_word_freq()
		elif input_type == "TEXT":
			self.get_text_data(docfile)
			self.get_word_freq()
		elif input_type == "HIER":
			self.get_hier_data(docfile, wordmap)

		self.freq_matrix = [[0 for i in range(len(self.doclist.keys()))] for
i in range(len(self.doclist.keys()))]
```

```
            self.clusterlist = {doc:[] for doc in self.doclist.keys()}

            self.compute_difference()


      def get_doc_dist(self, doc1, doc2):
            return round((sum((self.wordmap[doc1] -
self.wordmap[doc2])**2))**(0.5), 4)

      def compute_difference(self):
            for i, doci in enumerate(sorted(self.wordmap.keys())):
                  for j, docj in enumerate(sorted(self.wordmap.keys())):
                        if i == j:
                              self.freq_matrix[i][j] = inf
                        else:
                              self.freq_matrix[i][j] = self.get_doc_dist(doci,
docj)

      def get_min_doc_dist(self, docindex, return_dist = False):

            doc_vec = self.freq_matrix[docindex]

            mind = min(doc_vec)
            mindocs = [sorted(self.clusterlist.keys())[j] for j in
range(len(doc_vec)) if doc_vec[j] == mind]

            if return_dist == True:
                  return mind, mindocs

            return mindocs[0]

      def cluster(self, return_for_hier = False):
            for i, doci in enumerate(sorted(self.doclist.keys())):
                  for j, docj in enumerate(sorted(self.doclist.keys())):
                        if i!=j and self.freq_matrix[i][j] < self.threshold and
docj not in self.clusterlist[doci]:
                              self.clusterlist[doci].append(docj)


            clusters = []
            for curr_docindex, curr_doc in
enumerate(sorted(self.clusterlist.keys())):
                  nearest_doc = self.get_min_doc_dist(curr_docindex)
                  clusters.append({curr_doc, nearest_doc})

            end_clusters = []

            while clusters:
                  flag_merged = True
                  first = clusters.pop(0)
                  while flag_merged:
```

```
                        flag_merged = False
                        for i in range(len(clusters)):
                              if first.intersection(clusters[i]):
                                    first.update(clusters[i])
                                    clusters[i] = set()
                                    flag_merged = True

                  clusters = [j for j in clusters if len(j) != 0]
                  end_clusters.append(first)

            if self.input_type != "HIER":
                  self.clusters = {"Cluster "+str(i):end_clusters[i] for i in
range(len(end_clusters))}
            else:
                  self.clusters = {"Topic "+str(i):end_clusters[i] for i in
range(len(end_clusters))}

            if return_for_hier:
                  hier_data = {}

                  for curr_clus in self.clusters.keys():
                        # centroid = np.empty((self.wordmap[0], 1), np.float64)
                        centroid = np.array([0 for i in
range(len(self.wordlist))])
                        for doc in self.clusters[curr_clus]:
                              centroid += self.wordmap[doc]

                        centroid = centroid / len(self.clusters[curr_clus])
                        hier_data[curr_clus] = np.round(centroid, 4)

                  return(self.clusters, hier_data)




      def print_details(self):
            print("The document vectors look as follows:")
            pprint(self.wordmap)

            print("The documents are clustered using the Nearest Neighbour
method as follows:")
            pprint(self.clusters)


if __name__ == "__main__":
      wordlist = ["automotive", "car", "motorcycle", "self-drive", "iot",
"hire", "dhoni"]
      docfile = "docfile.txt"

      print("\nClustering of document files", '-'*15)
      vsobj = VSCluster(threshold=1.5)
```

```
      vsobj.fit(wordlist, docfile, input_type="TEXT")
      lvl1_cluslist, lvl1_clusmap = vsobj.cluster(return_for_hier=True)
      vsobj.print_details()


      print("\nHierrachial/Topical Clustering of document files", '-'*15)
      vsobj_hier = VSCluster(threshold=0.1)
      vsobj_hier.fit(wordlist=wordlist, docfile=lvl1_cluslist,
input_type="HIER", wordmap=lvl1_clusmap)
      vsobj_hier.cluster()
      vsobj_hier.print_details()



      urlwordlist = ["tesla", "electric", ["car", "vehicle", "automobile"],
"pollution", "de-monetisation" , "gst" , "black money"]
      urldocfile = "urldocfile.txt"

      print("\nClustering of document Webpages", '-'*15)
      vsobj2 = VSCluster(threshold=75)
      vsobj2.fit(urlwordlist, urldocfile, input_type="URL")
      url_cluslist, url_clusmap = vsobj2.cluster(return_for_hier=True)
      vsobj2.print_details()

      print("\nHierrachial/Topical Clustering of document Webpages", '-'*15)
      vsobj2_hier = VSCluster(threshold=0.1)
      vsobj2_hier.fit(wordlist=urlwordlist, docfile=url_cluslist,
input_type="HIER", wordmap=url_clusmap)
      vsobj2_hier.cluster()
      vsobj2_hier.print_details()
```

Output:

```
/media/anonymous/Work/Vit/Semester 5/WM/Lab/L6_VSClustering echo 16BCE1156
16BCE1156
/media/anonymous/Work/Vit/Semester 5/WM/Lab/L6_VSClustering python3 L6_VSClustering_1156.py

Clustering of document files ---------------
The document vectors look as follows:
{'doc 1': array([1, 0, 0, 0, 0, 0, 0]),
 'doc 2': array([1, 1, 0, 0, 0, 0, 0]),
 'doc 3': array([0, 0, 1, 0, 0, 0, 0]),
 'doc 4': array([0, 1, 0, 1, 1, 0, 0]),
 'doc 5': array([1, 0, 0, 0, 0, 1, 0]),
 'doc 6': array([0, 1, 0, 0, 0, 0, 1]),
 'doc 7': array([0, 0, 0, 0, 0, 0, 1]),
 'doc 8': array([0, 1, 0, 0, 1, 0, 0]),
 'doc 9': array([0, 0, 0, 0, 0, 0, 1])}
The documents are clustered using the Nearest Neighbour method as follows:
{'Cluster 0': {'doc 2', 'doc 3', 'doc 1', 'doc 5'},
 'Cluster 1': {'doc 4', 'doc 8'},
 'Cluster 2': {'doc 6', 'doc 9', 'doc 7'}}

Hierrachial/Topical Clustering of document files ---------------
The document vectors look as follows:
{'Cluster 0': array([0.75, 0.25, 0.25, 0.  , 0.  , 0.25, 0.  ]),
 'Cluster 1': array([0. , 1. , 0. , 0.5, 1. , 0. , 0. ]),
 'Cluster 2': array([0.    , 0.3333, 0.    , 0.    , 0.    , 0.    , 1.    ])}
The documents are clustered using the Nearest Neighbour method as follows:
{'Topic 0': {'Cluster 1', 'Cluster 2', 'Cluster 0'}}
```

```
Clustering of document Webpages ---------------
The document vectors look as follows:
{'https://economictimes.indiatimes.com/industry/auto/auto-news/government-plans-new-policy-to-promote-electric-vehicles/articleshow/652
37123.cms': array([ 0, 22, 36,  0,  0,  2,  0]),
 'https://economictimes.indiatimes.com/small-biz/policy-trends/is-gst-helping-the-indian-economy-for-the-better/articleshow/65319874.cm
s': array([ 0,  0,  6,  0,  0, 42,  0]),
 'https://en.wikipedia.org/wiki/Toyota_Prius': array([  3,  50, 298,   3,   0,   0,   0]),
 'https://inc42.com/buzz/electric-vehicles-this-week-centre-reduces-gst-on-lithium-ion-batteries-hyundai-to-launch-electric-suv-in-indi
a-and-more/': array([0, 2, 1, 0, 0, 1, 0]),
 'https://indianexpress.com/article/india/demonetisation-effects-cash-crisis-mobile-wallets-internet-banking-4406005/': array([ 0,  0,
24,  0,  0,  0,  0]),
 'https://indianexpress.com/article/india/india-news-india/demonetisation-hits-electric-vehicles-industry-society-of-manufacturers-of-e
lectric-vehicles-4395104/': array([ 0, 12, 16,  0,  0,  0,  1]),
 'https://www.financialexpress.com/auto/car-news/mahindra-to-launch-indias-first-electric-suv-in-2019-all-new-e-verito-sedan-on-cards/1
266853/': array([ 0, 31, 26,  0,  0,  0,  0]),
 'https://www.hrblock.in/blog/impact-gst-automobile-industry-2/': array([ 0,  0, 28,  0,  0, 61,  0]),
 'https://www.livemint.com/Politics/ySbMKTIC4MINsz1btccBJO/How-demonetisation-affected-the-Indian-economy-in-10-charts.html': array([0,
 0, 1, 0, 0, 1, 3]),
 'https://www.news18.com/news/business/how-gst-will-curb-tax-evasion-1446035.html': array([ 0,  0,  0,  0,  0, 16,  0]),
 'https://www.youthkiawaaz.com/2017/12/impact-of-demonetisation-on-the-indian-economy/': array([0, 0, 5, 0, 0, 0, 0]),
 'https://www.zigwheels.com/newcars/Tesla': array([22,  6, 68,  0,  0,  0,  0])}
```

```
The documents are clustered using the Nearest Neighbour method as follows:
{'Cluster 0': {'https://economictimes.indiatimes.com/industry/auto/auto-news/government-plans-new-policy-to-promote-electric-vehicles/a
rticleshow/65237123.cms',
               'https://en.wikipedia.org/wiki/Toyota_Prius',
               'https://www.financialexpress.com/auto/car-news/mahindra-to-launch-indias-first-electric-suv-in-2019-all-new-e-verito-se
dan-on-cards/1266853/',
               'https://www.zigwheels.com/newcars/Tesla'},
 'Cluster 1': {'https://economictimes.indiatimes.com/small-biz/policy-trends/is-gst-helping-the-indian-economy-for-the-better/articlesh
ow/65319874.cms',
               'https://inc42.com/buzz/electric-vehicles-this-week-centre-reduces-gst-on-lithium-ion-batteries-hyundai-to-launch-electr
ic-suv-in-india-and-more/',
               'https://www.hrblock.in/blog/impact-gst-automobile-industry-2/',
               'https://www.livemint.com/Politics/ySbMKTIC4MINsz1btccBJO/How-demonetisation-affected-the-Indian-economy-in-10-charts.ht
ml',
               'https://www.news18.com/news/business/how-gst-will-curb-tax-evasion-1446035.html',
               'https://www.youthkiawaaz.com/2017/12/impact-of-demonetisation-on-the-indian-economy/'},
 'Cluster 2': {'https://indianexpress.com/article/india/demonetisation-effects-cash-crisis-mobile-wallets-internet-banking-4406005/',
               'https://indianexpress.com/article/india/india-news-india/demonetisation-hits-electric-vehicles-industry-society-of-manu
facturers-of-electric-vehicles-4395104/'}}

Hierrachial/Topical Clustering of document Webpages ---------------
The document vectors look as follows:
{'Cluster 0': array([  6.25,  27.25, 107.  ,   0.75,   0.  ,   0.5 ,   0.  ]),
 'Cluster 1': array([ 0.    ,  0.3333,  6.8333,  0.    ,  0.    , 20.1667,  0.5   ]),
 'Cluster 2': array([ 0. ,  6. , 20. ,  0. ,  0. ,  0. ,  0.5])}
The documents are clustered using the Nearest Neighbour method as follows:
{'Topic 0': {'Cluster 1', 'Cluster 2', 'Cluster 0'}}
```