به نام خدا

گزارش فنی پروژه‌ی داده کاوی با پایتون

فصل 15

خوشه بندی اسناد

رامین بدری

شماره دانشجویی : 9414163002

نام استاد : دکتر امیرخانی

فهرست

بخش اول: اهداف کلی فصل

در این فصل به‌طور کلی می‌خواهیم کاربرد داده‌کاوی در مجموعه داده‌های متنی را بررسی نماییم. با ادغام مفاهیم داده‌های متنی با خوشه‌بندی به مبحث کلی‌تری به‌نام خوشه‌بندی اسناد (Text Clustering) خواهیم رسید. در واقع از الگوریتم‌های داده‌کاوی استفاده خواهیم نمود با این رویکرد که در اینجا داده‌ی مورد نظر ما –برخلاف معمول- متن و بطور کلی سند می‌باشد؛ در متن‌کاوی می‌خواهیم از داده‌های معمول و به ظاهر بی‌ارزش متنی، نکات و روابط پنهان، ارزشمند و معنی‌دار رو استخراج نماییم.

بخش دوم: خصوصیات مجموعه داده

داده‌ی استفاده شده در این پروژه، از یک شرکت معروف در زمینه برنامه‌ریزی سفر به‌نام TripAdvisor می‌باشد. این مجموعه‌ی داده شامل 1850 داده متنی(سند) می‌باشد که هرسند حاوی نظرات و امتیازات داده شده توسط سایت – براساس نظر مسافران قبلی - در رابطه با هر هتل می‌باشد. از آنجاییکه این تعداد سند برای انجام کار ما زیاد و سرعت اجرا را پایین می‌آورد، تعداد 100 سند اول از این مجموعه به عنوان ورودی و داده‌خام انتخاب شده است. این انتخاب ترتیب خاصی نداشته و صرفا بصورت تصادفی هم می‌تواند باشد.

بخش سوم: معرفی زبان و محیط برنامه‌نویسی

زبان‌های برنامه‌نویسی R و Python معروفترین و قدرتمندترین زبان‌های برنامه‌نویسی برای علم داده می‌باشند. در اینجا از زبان برنامه‌نویسی Python 3.5 استفاده شده است. توزیع قدرتمندی از پایتون به نام Anaconda

وجود دارد که بطور خاص برای انجام این قبیل پروژه‌ها طراحی و توسعه داده شده است. در این توزیع از پایتون پکیج‌ها و کتابخانه‌های متنوعی برای پیاده‌سازی تکنیک‌های داده‌کاوی موجود می‌باشند.

برای نوشتن و اجرای کد برنامه هم می‌توان از خود محیط Anaconda و هم از تمامی IDEهای پایتون استفاده نمود. در این پروژه از محیط برنامه‌نویسی معروف پایتون به نام PyCharm و ورژن 2016.1.4 استفاده شده است.

تنها نکته باقیمانده این است که باید در ابتدا از قسمت File → setting→ project interpreter آدرس مفسر Anaconda را به نرم افزار بدهیم تا کتابخانه‌های آن عملیاتی و قابل استفاده بشوند.

بخش چهارم: توضیحات کد برنامه

بطور کلی برنامه از 4 تابع تشکیل شده است، در شکل زیر این 4 تابع مشخص می‌باشند:

```
def tokenize_and_stem_and_removestopwords(input_text):...

def k_means(tfidfvector, numofclusters):...

def show_result(result_array, clustercounts):...

def draw_shape(a, b, c):...
```

روند فراخوانی توابع فوق بصورت زیر می‌باشد:

در ابتدا قطعه کد زیر اجرا میشود:

```
1   import pandas as pd
2   import nltk
3   from nltk.stem.porter import *
4   import string
5   from sklearn.feature_extraction.text import TfidfVectorizer
6   from sklearn.cluster import KMeans
7   import os
8   import matplotlib.pyplot as plt
9   from sklearn.manifold import MDS
10  from sklearn.metrics.pairwise import cosine_similarity
11  from scipy.cluster.hierarchy import ward, dendrogram
12
13  path = "C:/Users/vaio/Desktop/ramin-project/source/TripAdvisor_First_100_Hotels"
14  token_dict = {}
15  stopwords = nltk.corpus.stopwords.words('english')
16  stemmer = PorterStemmer()
17  document_names = []
```

در شکل فوق ابتدا تمامی کتابخانه‌ها و پکیج‌های مورد نیاز در پروژه، عملیاتی می‌شوند. سپس در خطوط 12 تا 17، مسیر مجموعه داده را مشخص کرده و مقدمات کار را فراهم می‌نماییم.

توجه: برای اجرای برنامه باید مسیر داده شده را به مسیر موجود بروی هارد دیسک خود تغییر دهید.

سپس قطعه کد زیر اجرا خواهد شد:

```
120 # iterate throw data collection
121 for subdir, dirs, files in os.walk(path):
122     for file in files:
123         file_path = subdir + os.path.sep + file
124         shakes = open(file_path, encoding="utf8")
125         new_path = file_path.replace(file_path, file_path[72:90])
126         document_names.append(new_path)
127         text = shakes.read()
128         lowers = text.lower()
129         no_punctuation = lowers.translate(string.punctuation)
130         token_dict[file] = no_punctuation
```

در این قسمت می‌خواهیم فایل‌های موجود در مسیر اولیه (path) را یکی یکی خوانده و متن داخل آنها را به حروف کوچک تبدیل کرده و حاصل را در token_dict ذخیره می‌نماییم. ضمنا لیستی از اسامی هرسند را در document_names ذخیره می‌نماییم.

سپس نوبت به اجرای کد زیر می‌رسد:

```
140  flag = True
141  ans_1 = input("Please enter number of clusters(k for k-means): ")
142  while flag:
143      try:
144          val = int(ans_1)
145          flag = False
146      except ValueError:
147          print("That's not an int!")
148  tfidf = TfidfVectorizer(tokenizer=tokenize_and_stem_and_removestopwords, stop_words='english', ngram_range=(1, 2))
149  tfs = tfidf.fit_transform(token_dict.values())
```

با اجرای کدهای فوق، ابتدا از کاربر درباره تنظیم پارامتر K که همان تعداد خوشه‌ها می‌باشد، سوال می‌شود. سپس با استفاده ار تعریف انجام شده در خط 133 تنظیمات اولیه برای ایجاد ماتریس tf-idf برای توکن‌ها و برقرار می‌شوند. نکته مهم لحاظ کردن پارامتر ngram_range= (1,2) می‌باشد که نشان می‌دهد – مطابق تنظیمات داخل رپیدماینر – خواستار بدست آوردن1-gram و 2-gramهای هر توکن نیز هستیم. پارامتر tokenizer هم نحوه استخراج توکن‌ها را مشخص می‌کند که با تابع tokeniz_and_stem_and_removestopwords برابر شده است، در ادامه آنرا توضیح می‌دهیم. درنهایت هم token_dict را برای اجرا به tfidf نسبت می‌دهیم.(خط 135)

حال تابع tokeniz_and_stem_and_removestopwords اجرا می‌شود. بدنه داخلی آن در شکل زیر آمده است، همانطور که از نامش پیداست، این تابع متن را به عنوان ورودی گرفته و عملیات tokenization, stemming و حذف کلمات پرتکرار در زبان انگلیسی(stopword) را انجام می‌دهد. در نهایت خروجی این تابع کلمات آماده برای انجام داده کاوی می‌باشد. برای اجرای این بخش از کتابخانه‌ی nltk کمک گرفته‌ایم.

```
21  def tokenize_and_stem_and_removestopwords(input_text):  # Tokenize, Stem and remove stopwords
22      tokens = [word for sent in nltk.sent_tokenize(input_text) for word in nltk.word_tokenize(sent)]
23      filtered_stopwords = [w for w in tokens if w not in stopwords]
24      filtered_tokens = []
25      for token in filtered_stopwords:
26          if re.search('[a-zA-Z]', token):
27              filtered_tokens.append(token)
28      stems = [stemmer.stem(t) for t in filtered_tokens]
29      return stems
```

متغیرهای stopwords و stemmer که در ابتدا مقدار دهی شده بودند در اینجا مورد استفاده قرار گرفتند. در این تابع به ترتیب عملیات استخراج توکن(خط 22)، حذف کلمات پرتکرار(خط 23)، حذف تمامی کاراکترهای غیر از الفبای انگلیسی(خطوط 25 تا 27) و در نهایت stemming(خط 28) صورت گرفته است. خروجی این تابع پس از تبدیل به بردارهای عددی tf-idf در متغیر tfs ذخیره می‌شود.

حال کد زیر اجرا میشود:

```
136     result = k_means(tfs, ans_1)
```

تابع k_means با پارامترهای tfs و ans_1(که همان تعداد خوشه‌هاست) فراخوانی می‌شود.

در این تابع عملیات اصلی خوشه‌بندی اسناد صورت می‌گیرد. پارامترهای ورودی این تابع ماتریس tf-idf اسناد و تعداد خوشه‌ها می‌باشند. بدنه‌ی داخلی این تابع در شکل زیر مشخص می‌باشد.

در این تابع ابتدا خوشه‌بندی بروی اسناد با تعداد خوشه‌های مشخص شده صورت می‌گیرد و پس از آن در لیست clusters لیست مربوط به برچسبی را که هر سند گرفته است مشاهده می‌نماییم. دو لیست متفاوت را متناسب با تعداد خوشه‌ها تشکلی می‌دهیم. یکی به‌نام cluster_id که لیستی از لیست شماره‌ی خوشه‌های هر سند می‌باشد.(ساختار آن به این شکل است [[....],[....],[....],[....],[....]]) اسناد خوشه‌ی شماره صفر در لیست اول و به‌همین ترتیب ذخیره می‌شوند در حین همین ذخیره سازی، اندیس آن سند در لیست clusters نیز برای map شدن به نام آن سند در لیست دوم یعنی لیست doc_id_per_cluster_id ذخیره می‌شوند. خروجی این تابع این سه لیست می‌باشد.

```
32      def k_means(tfidfvector, numofclusters):
33          cluster_id = []
34          doc_id_per_cluster = []
35          km = KMeans(n_clusters=numofclusters)
36          km.fit(tfidfvector)
37          clusters = km.labels_.tolist()
38          for k in range(0, numofclusters):
39              cluster_id.append([])
40              doc_id_per_cluster.append([])
41          for k in range(0, len(clusters)):
42              for i in range(0, numofclusters):
43                  if clusters[k] == i:
44                      cluster_id[i].append(clusters[k])
45                      doc_id_per_cluster[i].append(k)
46          return clusters, cluster_id, doc_id_per_cluster
```

حال نوبت به نمایش نتایج بدست آمده از خوشه‌بندی در خروجی می‌رسد. برای این کار خروجی تابع k_means را به تابع show_result(result, ans_1) می‌دهیم. ضمنا تعداد خوشه‌ها نیز یکی از پارامترهای ورودی این تابع است. (خط 151). بدنه داخلی این تابع در ادامه نمایش داده شده است.

```
51  def show_result(result_array, clustercounts):
52      print("**There are ", len(result_array[0]), " documents in TripAdvisor data collection**")
53      print("**The data collection is clustered in ", clustercounts, "clusters as shown in below**")
54      print("##################")
55      for j in range(0, len(result_array[1])):
56          print("There are: ", len(result_array[1][j]), " documents in cluster: ", j)
57          print("Here we can see names of the documents:")
58          print()
59          for m in result_array[2][j]:
60              print(document_names[m])
61          print()
62          print("==================")
```

پس از اجرای تابع فوق، ما می‌توانیم خوشه‌بندی را به تفکیک نام و تعداد اسنادی که در هرخوشه وجود دارند، مشاهده نماییم. در بخش تحلیل نتایج این نتیجه را خواهیم دید.

پس از اجرای تابع فوق بیشتر کار خوشه‌بندی انجام شده است. تنها قسمت باقیمانده قسمت زیر می‌باشد که با یک سوال از کاربر شروع میشود.

```
152 ans_2 = input("Do you want to see clustering visual shapes? y or n: ")
153 if ans_2 == "y":
154     draw_shape(result[0], document_names, tfs, int(ans_1))
```

دو نوع نمودار مختلف برای نمایش گرافیکی خوشه‌بندی اسناد آماده شده است که اگر کاربر تمایل به رویت آنها داشته باشد در پاسخ به سوال فوق باید جواب ''y'' را نوشته و دکمه‌ی اینتر را فشار بدهد. در این صورت آخرین تابع این پروژه یعنی تابع draw_shape با پارامترهای ورودی result[0] = clusters, document_names, tfs و تعداد خوشه‌ها(ans_1) فراخوانی می‌شود. قسمت اول بدنه‌ی داخلی این تابع در شکل زیر آمده است.

```
65  def draw_shape(a, b, c, d):
66      dist = 1 - cosine_similarity(c)
67      mds = MDS(n_components=2, dissimilarity="precomputed", random_state=1)
68      pos = mds.fit_transform(dist)
69      xs, ys = pos[:, 0], pos[:, 1]
70      cluster_colors = []
71      cluster_names = []
72      prefix = "cluster_"
73      for i in range(0, d):
74          cluster_colors.append(colorize())
75          cluster_names.append(prefix+str(i))
76      df = pd.DataFrame(dict(x=xs, y=ys, label=a, title=b))
77      groups = df.groupby('label')
78      fig, ax = plt.subplots(figsize=(17, 9))  # set size
79      ax.margins(0.05)  # Optional, just adds 5% padding to the autoscaling
80      for name, group in groups:
81          ax.plot(group.x, group.y, marker='o', linestyle='', ms=12,
82                  label=cluster_names[name], color=cluster_colors[name], mec='none')
83          ax.set_aspect('auto')
84          ax.tick_params(
85              axis='x',  # changes apply to the x-axis
86              which='both',  # both major and minor ticks are affected
87              bottom='off',  # ticks along the bottom edge are off
88              top='off',  # ticks along the top edge are off
89              labelbottom='off')
90          ax.tick_params(
91              axis='y',  # changes apply to the y-axis
92              which='both',  # both major and minor ticks are affected
93              left='off',  # ticks along the bottom edge are off
94              top='off',  # ticks along the top edge are off
95              labelleft='off')
96      ax.legend(numpoints=1)  # show legend with only 1 point
97      # add label in x,y position with the label as the film title
98      for i in range(len(df)):
99          ax.text(df.ix[i]['x'], df.ix[i]['y'], df.ix[i]['title'], size=5)
100     plt.show()
101     plt.close()
```

با اجرای قسمت فوق و با کمک کتابخانه matplotlib.pyplot که مخصوص رسم نمودارهای متفاوت می‌باشد، می‌توانیم نتیجه‌ی خوشه‌بندی را بصورت گرافیکی و به تفکیک رنگ (اسناد هر خوشه یک رنگ منحصر بفرد دارند) مشاهده می‌نماییم. نکته مهم این است که به تعداد خوشه‌هایی که کاربر در اولین قدم به برنامه داده است(پارامتر k) رنگ و نام متفاوت تولید می‌شود.(خطوط 70 تا 75) مابقی خطوط برنامه تنظیمات رسم نمودار می‌باشند که از توضیح آن صر نظر می‌نماییم. در بخش بعد این نمودار را خواهیم دید.

قسمت دوم این تابع مربوط به رسم نمودار خوشه‌بندی سلسله مراتبی اسناد می‌باشد که در اینجا با تعداد خوشه‌ها کاری نداریم و صرفا باتوجه به شباهت بدست آمده بین اسناد(خط 66 – شباهت کسینوسی) به خوشه‌بندی

اسناد می‌پردازیم، نتیجه این قسمت هم نمودار معروف دندروگرام می‌باشد. کد این قسمت نیز در شکل زیر آمده است.

```
103     # dendrogram plot
104     linkage_matrix = ward(dist)  # define the linkage_matrix using ward clustering pre-computed distances
105     fig, ax = plt.subplots(figsize=(15, 20))  # set size
106     ax = dendrogram(linkage_matrix, orientation="top", labels=document_names, leaf_font_size=6)
107     plt.tick_params(
108         axis='x',  # changes apply to the x-axis
109         which='both',  # both major and minor ticks are affected
110         bottom='on',  # ticks along the bottom edge are off
111         top='on',  # ticks along the top edge are off
112         labelbottom='on')
113     plt.show()
114     plt.close()
115     return True
```

نکته این قسمت هم نام در خط 106 می‌باشد که برای پارامتر labels = document_names تنظیم شده است، این‌کار باعث می‌شود که اسمامی اسناد به عنوان برچسب برگ‌های نمودار نمایان بشوند. این نمودار هم به کمک کتابخانه matplotlib.pyplot رسم می‌شود. در بخش بعد نتیجه این قسمت را نیز مشاهده خواهیم نمود.

بخش پنجم: نمایش و تحلیل نتایج

قبل از شروع این بخش، ذکر این نکته ضروری‌ست که از آنجاییکه الگوریتم K-means در مرحله‌ی آغازین خود به‌صورت تصادفی عمل می‌نماید و نقاط شروع را بدون ترتیب خاص اتخاذ می‌نماید، در هربار اجرای برنامه ممکن است نتایج خروجی متفاوتی مشاهده بشود.

پس از اجرای برنامه با پیغام زیر مواجه می‌شویم:

Please enter number of clusters (k for k-means):

به عنوان مثال عدد 5 را وارد نموده و دکمه اینتر را می‌زنیم.

پس از اتمام برنامه و در قسمت خروجی، می‌توانیم نتیجه زیر را که قسمتی از خروجی این مرحله می‌باشد، مشاهده نماییم:

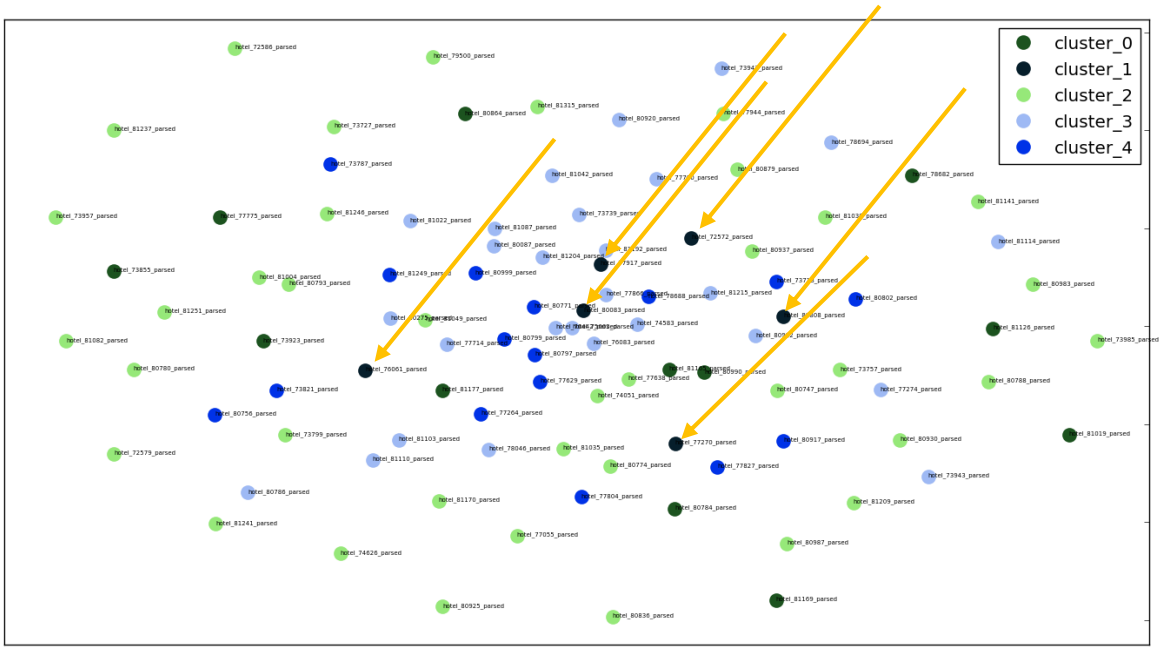
```
D:\Anaconda3\python.exe C:/Users/vaio/ramin/__init__.py
Please enter number of clusters(k for k-means): 5
**There are  100  documents in TripAdvisor data collection**
**The data collection is clustered in  5 clusters as shown in below**
##################
There are:  12  documents in cluster:  0
Here we can see names of the documents:

hotel_73855_parsed
hotel_73923_parsed
hotel_77775_parsed
hotel_78682_parsed
hotel_80784_parsed
hotel_80864_parsed
hotel_80990_parsed
hotel_81019_parsed
hotel_81126_parsed
hotel_81165_parsed
hotel_81169_parsed
hotel_81177_parsed

==================
There are:  6  documents in cluster:  1
Here we can see names of the documents:

hotel_72572_parsed
hotel_76061_parsed
hotel_77270_parsed
hotel_77917_parsed
hotel_80083_parsed
hotel_80808_parsed

==================
There are:  39  documents in cluster:  2
Here we can see names of the documents:

hotel_72579_parsed
hotel_72586_parsed
hotel_73727_parsed
```

در اینجا بصورت تفکیک شده می‌توانیم اسامی اسناد داخل هر خوشه را مشاهده نماییم. برای مثال در خوشه-شماره‌ی 0 تعداد 12 سند وجود دارند که اسامی آنان از بالا به پایین :

```
hotel_73855_parsed
hotel_73923_parsed
hotel_77775_parsed
hotel_78682_parsed
hotel_80784_parsed
hotel_80864_parsed
hotel_80990_parsed
hotel_81019_parsed
hotel_81126_parsed
hotel_81165_parsed
hotel_81169_parsed
hotel_81177_parsed
```

برای مثالی دیگر بروی اسناد داخل خوشه‌ی شماره 1 تمرکز می‌کنیم. در این خوشه 6 سند موجود می‌باشند که بروی شکل مشخص شده‌اند. حال به سراغ نمایش نمودارها می‌رویم.

برای مشاهده نمودارها در جواب سوال پرسیده شده برنامه باید حرف ''y'' را بزنیم. در این صورت ابتدا نمودار ظاهر می‌شود. البته به دلیل تعداد زیاد اسناد مشاهده و خواندن نام سند کمی دشوار است. در نمودار زیر همان 6 سند مشخص شده‌ی مربوط به خوشه‌ی 1 را مشاهده می‌نماییم که به رنگ سیاه می‌باشند.

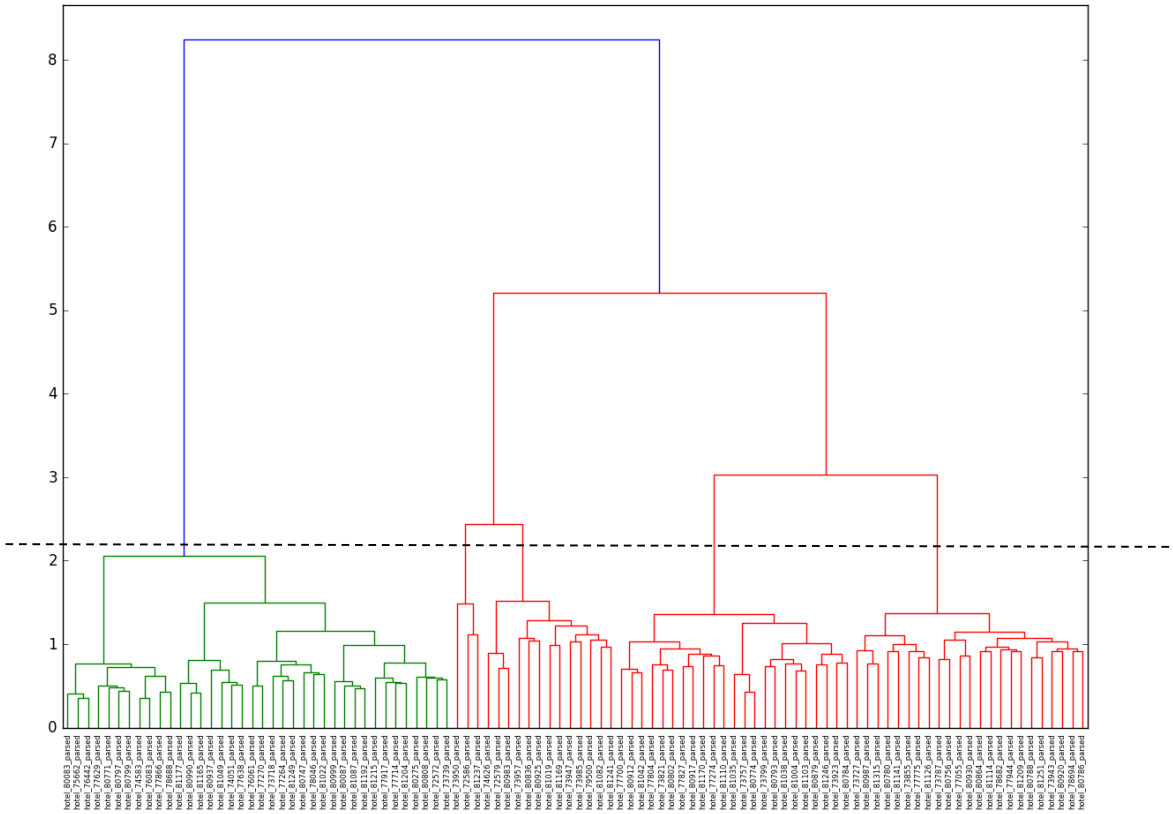


انتخاب رنگ‌ها هم بصورت تصادفی می‌باشند.

و در ادامه با بستن پنجره نمودار فوق نمودار دندروگرام مربوط به این خوشه‌بندی ظاهر می‌شود که در شکل زیر می‌توانیم آنرا مشاهده نماییم.

متاسفانه حتی با وجود بزرگنمایی شکل بازهم به دلیل تعداد زیاد برگ‌ها، اسامی اسناد به‌خوبی قابل مشاهده نمی‌باشند که در نهایت با اجرای برنامه توسط کاربر ایم مشکل رفع خواهد شد. می‌توان مشاهده نمود که بطور کلی اسناد به دو خوشه تقسیم شده‌اند.(در بالاترین سطح خوشه بندی)

برای رسیدن به خوشه‌بندی با k = 5 باید نمودار را از مقطع مشخص شده با خط چین در شکل برش دهیم.



تمامی کدهای برنامه به همراه خروجی آن در فایل جداگانه و بصورت متن خام در دسترس می‌باشند.(به انضمام عکس‌های نمودارها)