TOOL

Hierarchical and K-Means Clustering

In this course, you've examined the process behind hierarchical and k-means clustering, and you've practiced using these algorithms to categorize movies according to genre. Use this tool as a reference when clustering data with either of these two algorithms and plotting the algorithm's results.

Using Python With This Tool

The portions of this tool with a gray background are code text you can use to do the examples included in this tool. You can also modify them to use with your data. In these examples:

- Commands are the lines of code that don't begin with a pound sign (#). Type these lines into Python to carry out the command.
- Commented text begins with a one-pound sign and explains what the code does.

Part 1: Preprocess Your Data Set

Setup

Use the following code to reset your environment and import all necessary modules.

```
%reset -f
from IPython.core.interactiveshell import InteractiveShell as IS
IS.ast_node_interactivity = "all"
import numpy as np, pandas as pd, seaborn as sns, matplotlib.pyplot as plt,
scipy
from sentence_transformers import SentenceTransformer
from sklearn.cluster import AgglomerativeClustering, KMeans
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
import plotly.graph_objects as go
pd.set_option('max_rows', 5, 'max_columns', 20, 'max_colwidth', 100,
'precision', 2)
```

Load the Data File Into a Dataframe

Before you can cluster your data, you will first want to convert your raw data into a dataframe of vector embeddings. Here, load the data file containing the information you want to cluster; this tool uses a dataset of movies and descriptions. To save processing time, the movies in this sample are limited to those with both 'Action' and 'Family' genre tags.

```
df0 = pd.read_csv('movies.zip').fillna('')  # load TMDB file
df0.index = df0.title        # set index labels for visual convenience only
dfS = df0[df0.genres.str.contains('Action') & df0.genres.str.
contains('Family')]
df = (dfS.title +'. '+ dfS.tagline +'. '+ dfS.overview).to_frame().
rename(columns={0:'Desc'})
df
```

By the end of this step, your dataframe should closely resemble the one below:

original_title	Desc
Jack the Giant Slayer	Jack the Giant Slayer. Prepare for a giant adventure. The story of an ancient war that is reigni...
Brave	Brave. Change your fate.. Brave is set in the mystical Scottish Highlands, where Mérida is the p...
...	
Letters to God	Letters to God. . A young boy fighting cancer writes letters to God, touching lives in his neigh...
Love's Abiding Joy	Love's Abiding Joy. . The continued Westward journey of settlers Missie and Willie Lahaye. Their...

Encode Movies with SBERT Language Model

Next, load the pre-trained language model, which you will use to embed the dataframe.

```
# Load a pre-trained model, 250MB
SBERT = SentenceTransformer('paraphrase-albert-small-v2')

# Encoding textual descriptions with numeric vectors
%time mEmb = SBERT.encode(df.Desc)
dfEmb = pd.DataFrame(mEmb, index=df.index)
dfEmb
```

After embedding, your dataframe should resemble the one below.

title	0	1	2	3	4	...	763	764	765	766	767
Jack the Giant Slayer	0.09	-0.13	0.17	-0.25	0.02	...	-0.47	0.10	-0.49	0.62	-0.58
Brave	0..37	0.22	0.18	-0.32	0.06	...	0.24	-0.07	-0.26	0.27	0.33
...	...	...	...	...	...	...	...	...	...	...	...
Letters to God	-0.06	0.14	-0.46	-0.26	0.21	...	0.34	-0.40	-0.02	0.13	-0.83
Love's Abiding Joy	-0.28	-0.44	-0.07	-0.34	0.08	...	0.32	-0.55	-0.39	0.14	-0.14

You only need to implement one clustering approach from this section before moving on to Part 3: Plotting Your Results.

Part 2: Select a Clustering Algorithm and Group Your Data

Now that your sentences are encoded as vectors, you can begin clustering your data using either hierarchical or k-means clustering. Follow along with the steps for either algorithm in the section below before continuing on to the next section.

Option 1: Hierarchical Clustering

Now that you've embedded your data into vectors, use `KMeans` to cluster the dataframe of embeddings, `dfEmb`, into *n* groups.

```
clustering_model = AgglomerativeClustering(n_clusters=3)
clustering_model.fit(dfEmb)    # build a hierarchical tree and assign
cluster labels to movies
```

Option 2: *k*-Means Clustering

Now that you've embedded your data into vectors, use `KMeans` to cluster the dataframe of embeddings, `dfEmb`, into *n* groups.

```
clustering_model = KMeans(n_clusters=3, random_state=0, n_init=30, max_
iter=1000)
clustering_model.fit(dfEmb)
```

Part 3: Plot Your Algorithm's Results

First, convert each vector embedding to two dimensions using Principal Component Analysis (PCA). You will use the coordinates in the resulting dataframe to create your plot.

```
mPC12 = PCA(n_components=2).fit_transform(dfEmb)
dfPC12 = pd.DataFrame(mPC12, columns=['x','y'], index=df.index)
dfPC12
```

title	x	y
Jack the Giant Slayer	2.83	-0.93
Brave	0.69	0.41
...	...	...
Letters to God	-0.32	2.78
Love's Abiding Joy	1.55	4.93

Draw a Color-Coded Scatter Plot

Next, define *n* colors in your plot, one for each cluster (in this example, *n* = 3). These colors will visually differentiate your groups.

```
# strings of RGB color values
LsPalette = [f'rgb({c[0]},{c[1]},{c[2]})' for c in sns.color_
palette('bright', clustering_model.n_clusters)]

# vector of colors (as RGB string) for each point
vColors = np.array(LsPalette)[clustering_model.labels_]
vColors[:2]
```

Next, define *n* colors in your plot, one for each cluster (in this example, *n* = 3). These colors will visually differentiate your groups.

```
# point labels with title+genre
sMovieGenres = [a + '; ' + b for for a,b in zip(dfS.index, dfS.genres)]
DMarkers = dict(size=2, line=dict(width=1, color=vColors), color=vColors)
goMargin = go.layout.Margin(l=0, r=0, b=0, t=0)

# draw scatter plot
goS = go.Scatter(x=dfPC12.x, y=dfPC12.y, mode='markers', marker=DMarkers,
                 text=sMovieGenres, name='movies');
goLayout = go.Layout(hovermode='closest', margin=goMargin, width=800,
                  height=500, xaxis={'title':'PC1'},
yaxis={'title':'PC2'});
fig = go.Figure(layout=goLayout)  # prepare a figure with specified layout
fig.add_trace(goS)                # plot movie points
```

Your final plot should generally resemble the one below. Note that this plot was produced with the k-means clustering algorithm for Action and Family movies, but your plot will vary with a different dataset or clustering approach.

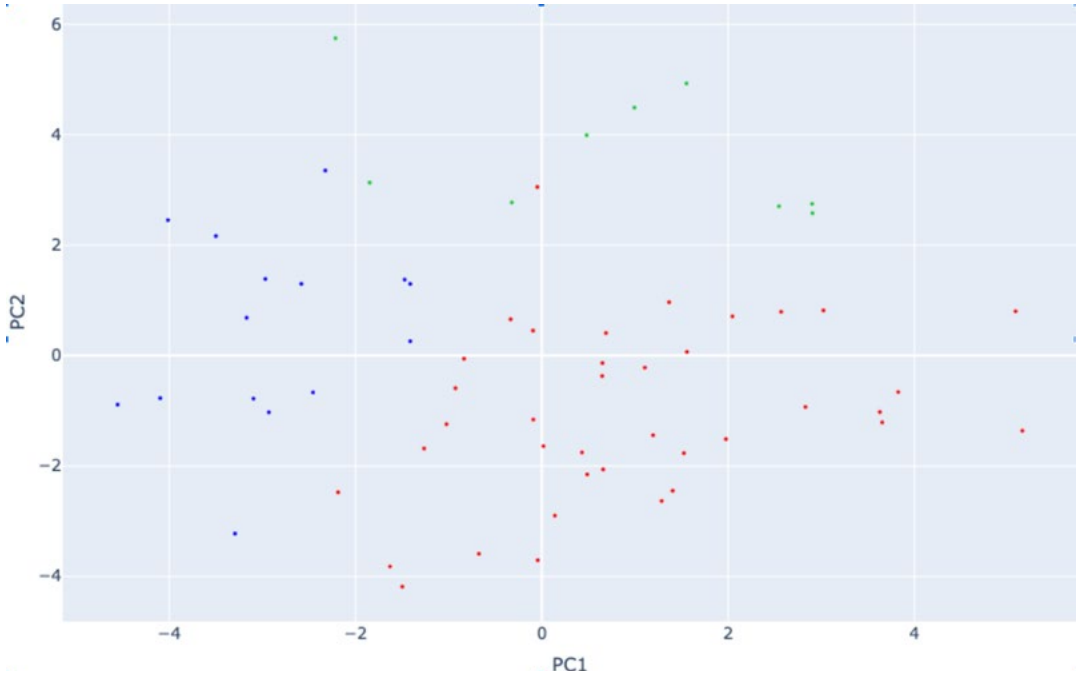

Figure A. See appendix.

Image Appendix

Figure A.

A scatter plot with x-axis PC1 and y-axis PC2 includes points corresponding to each movie in the movies.zip file having both “Action” and “Family” tags. Both axes range from -4 to 6. The color-coded points show three clusters: red, green, and blue. For details on individual points, refer to the corresponding data table below:

title	x	y	cluster
Jack the Giant Slayer	2.83	-0.93	1
Brave	0.69	0.41	1
Maleficent	-0.34	0.66	1
Big Hero 6	0.02	-1.64	1
Night at the Museum: Battle of the Smithsonian	0.65	-0.13	1
G-Force	-1.63	-3.83	1
The Last Airbender	5.16	-1.36	1
Kung Fu Panda 3	0.49	-2.16	1
How to Train Your Dragon 2	3.65	-1.21	1
Puss in Boots	-0.05	-3.71	1
Megamind	0.14	-2.9	1
Speed Racer	0.67	-3.59	1
Night at the Museum	0.66	-2.06	1
Eragon	3.63	-1.02	1
The Incredibles	0.65	-0.36	1
The Nutcracker: The Untold Story	0.48	3.99	2

title	x	y	cluster
Last Action Hero	1.19	-1.45	1
The Pink Panther	-2.19	-2.47	1
Mighty Joe Young	-0.83	-0.05	1
Inspector Gadget	-4.56	-0.89	0
Shark Tale	1.1	-0.22	1
Titan A.E.	3.83	-0.67	1
Osmosis Jones	-3.3	-3.22	0
Astro Boy	-0.09	0.45	1
The Pacifier	-2.46	-0.67	0
Thunderbirds	-1.26	-1.69	1
Race to Witch Mountain	1.53	-1.77	1
Baby's Day Out	-1.85	3.13	2
Journey to the Center of the Earth	2.9	2.75	2
Foodfight!	-1.5	-4.2	1
Spy Kids 3-D: Game Over	-1.41	0.25	0
Stormbreaker	-0.93	-0.59	1
Spy Kids 2: The Island of Lost Dreams	-1.41	1.29	0
Paul Blart: Mall Copy 2	-3.1	-0.78	0
Spy Kids	-3.5	2.16	0
Warriors of Virtue	1.97	-1.52	1
The Borrowers	-2.32	3.36	0

title	x	y	cluster
The Spy Next Door	-4.01	2.46	0
Spy Kids: All the Time in the World	-2.58	1.29	0
Paul Blart: Mall Cop	-2.93	-1.03	0
Jimmy Neutron: Boy Genius	-3.17	0.68	0
Teenage Mutant Ninja Turtles II: The Secret of the Ooze	0.43	-1.76	1
Teenage Mutant Ninja Turtles III	1.29	-2.64	1
The Black Hole	2.05	0.7	1
Nancy Drew	-0.05	3.05	1
3 Ninjas Kick Back	1.4	-2.44	1
Young Sherlock Holmes	-2.97	1.39	0
Catch That Kid	-1.47	1.38	0
See Spot Run	-4.1	-0.77	0
White Fang	3.02	0.82	1
Jonah: A VeggieTales Movie	2.57	0.79	1
The Crocodile Hunter: Collision Course	-0.09	-1.16	1
Crossroads	0.99	4.49	2
Legend of a Rabbit	-1.02	-1.24	1
Nicholas Nickleby	-2.21	5.75	2
The Other Side of Heaven	2.9	2.58	2
The Adventures of Huck Finn	5.08	0.8	1
The Man from Snowy River	2.54	2.71	2

title	x	y	cluster
Pokemon: Spell of the Unknown	1.37	0.97	1
Letters to God	-0.32	2.77	2
Love's Abiding Joy	1.55	4.94	2