05_generalizibility_silhouette_score

April 17, 2025

Load necessary libraries.

[9]:
```
### Basics
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import seaborn as sns
import os

### ML packages
from sklearn.cluster import KMeans, SpectralClustering, AgglomerativeClustering␣
 ↪#Hierarchial Clustering
from sklearn.mixture import GaussianMixture
import scipy.cluster.hierarchy as sch
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from collections import defaultdict
import umap
from sklearn.metrics import silhouette_samples

### Msc
import warnings
from PIL import Image

### OOP
from ml_utils import SilhouetteEvaluator, ClusterEvaluator, make_gmm,␣
 ↪make_hierarchical, make_spectral
```

Load dataset.

[10]:
```
df = pd.read_csv('00_authors.csv').rename(columns={'Unnamed: 0': 'Author'}).
 ↪drop(columns='BookID')
authors = df['Author'].values  # n_samples-length array

X_clean = df.drop(columns=['Author'])
X = X_clean.to_numpy()
```

```
# UMAP dimensionality reduction
warnings.filterwarnings("ignore", category=UserWarning, module="umap")  #␣
 ↪suppress joblib warning
umap_model = umap.UMAP(n_neighbors=10, min_dist=0.3, random_state=42)
X_umap = umap_model.fit_transform(X)
```

1 Validation

The method with the highest silhouette score generalizes best!

1.1 Kmeans++

Kmeans++ with and without UMAP:

1.1.1 Generalizibility

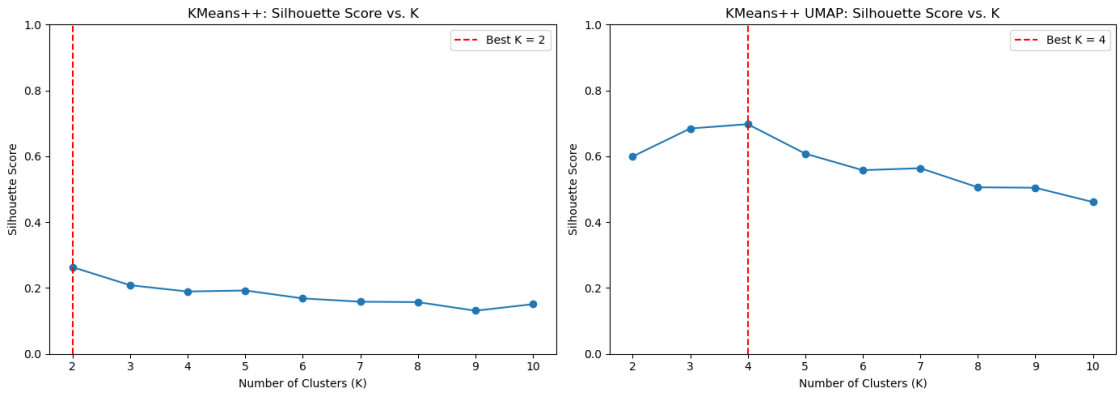
[11]:
```
# Kmeans++
print('\033[1m' + 'Kmeans++ without dimension reduction:' + '\033[0m')
evaluator_kmeans = SilhouetteEvaluator(X, KMeans, k_range=range(2, 11),␣
 ↪init='k-means++', n_init=10, max_iter=300)
scores, best_k = evaluator_kmeans.evaluate()
print(f'The scores across K = {scores}')
print(f"Best K: {best_k}")

# Kmeans++ with UMAP
print('\033[1m' + 'Kmeans++ with dimension reduction (UMAP):' + '\033[0m')
evaluator_kmeans_umap = SilhouetteEvaluator(X_umap, KMeans, k_range=range(2,␣
 ↪11), init='k-means++', n_init=10, max_iter=300)
scores, best_k = evaluator_kmeans_umap.evaluate()
print(f'The scores across K = {scores}')
print(f"Best K: {best_k}")

y_min = 0
y_max = 1

fig, axs = plt.subplots(1, 2, figsize=(14, 5))

evaluator_kmeans.plot("KMeans++", ax=axs[0])
axs[0].set_ylim(y_min, y_max)

evaluator_kmeans_umap.plot("KMeans++ UMAP", ax=axs[1])
axs[1].set_ylim(y_min, y_max)

plt.tight_layout()
plt.savefig('Media/viz/05/05_kmeans_silhouette_score')
plt.show()
```

Kmeans++ without dimension reduction:

```
The scores across K = {2: 0.2632255060236338, 3: 0.20836751818875782, 4:
0.18927180616599434, 5: 0.1924461042273646, 6: 0.16832953504579296, 7:
0.15834519448558668, 8: 0.15708860765446275, 9: 0.13097719253825474, 10:
0.15080653215185813}
Best K: 2
```

Kmeans++ with dimension reduction (UMAP):

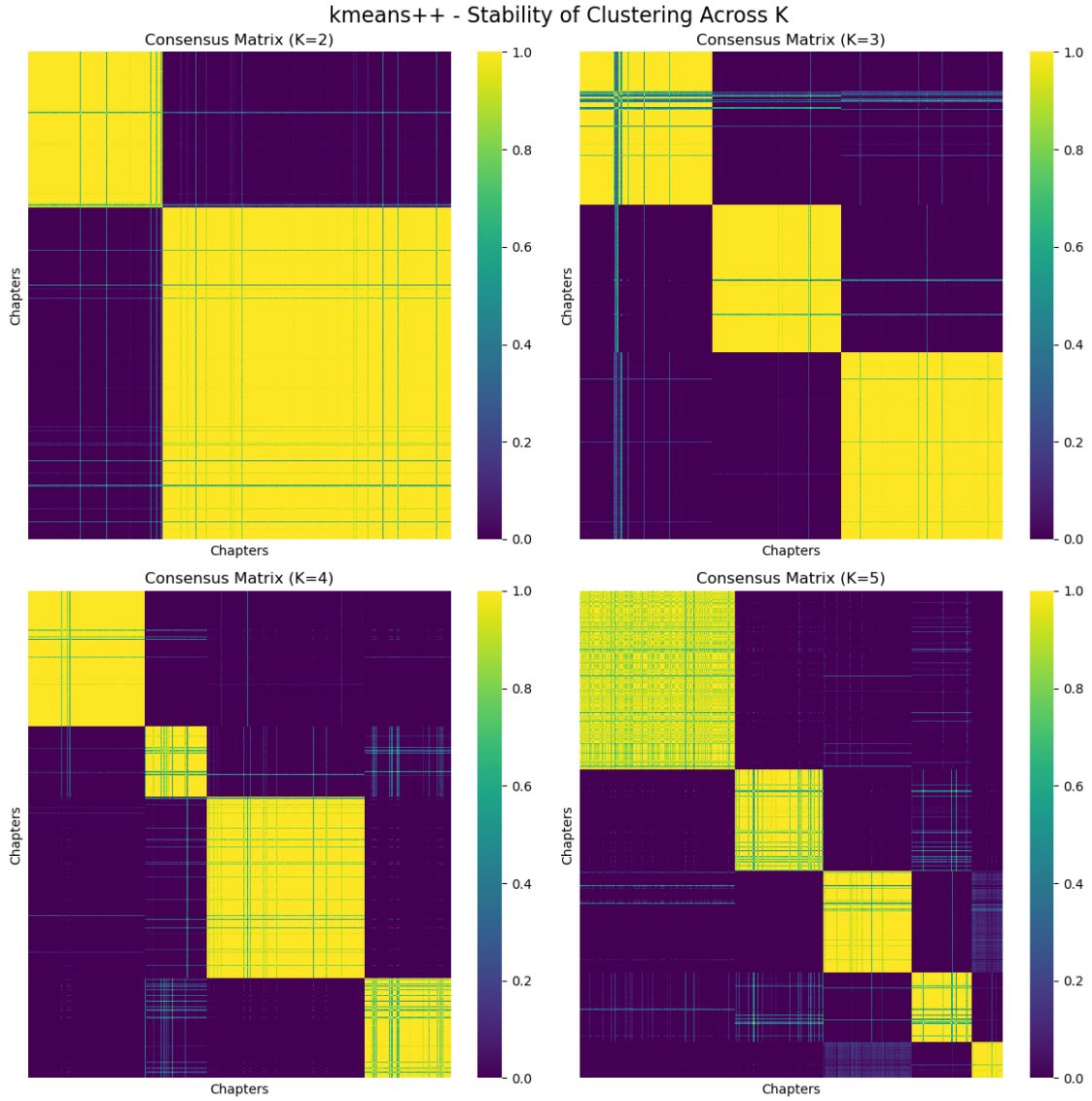
```
The scores across K = {2: 0.59943473, 3: 0.68435156, 4: 0.6975769, 5:
0.60805935, 6: 0.55773854, 7: 0.5637968, 8: 0.505771, 9: 0.5044795, 10:
0.4610399}
Best K: 4
```



1.1.2 Stability

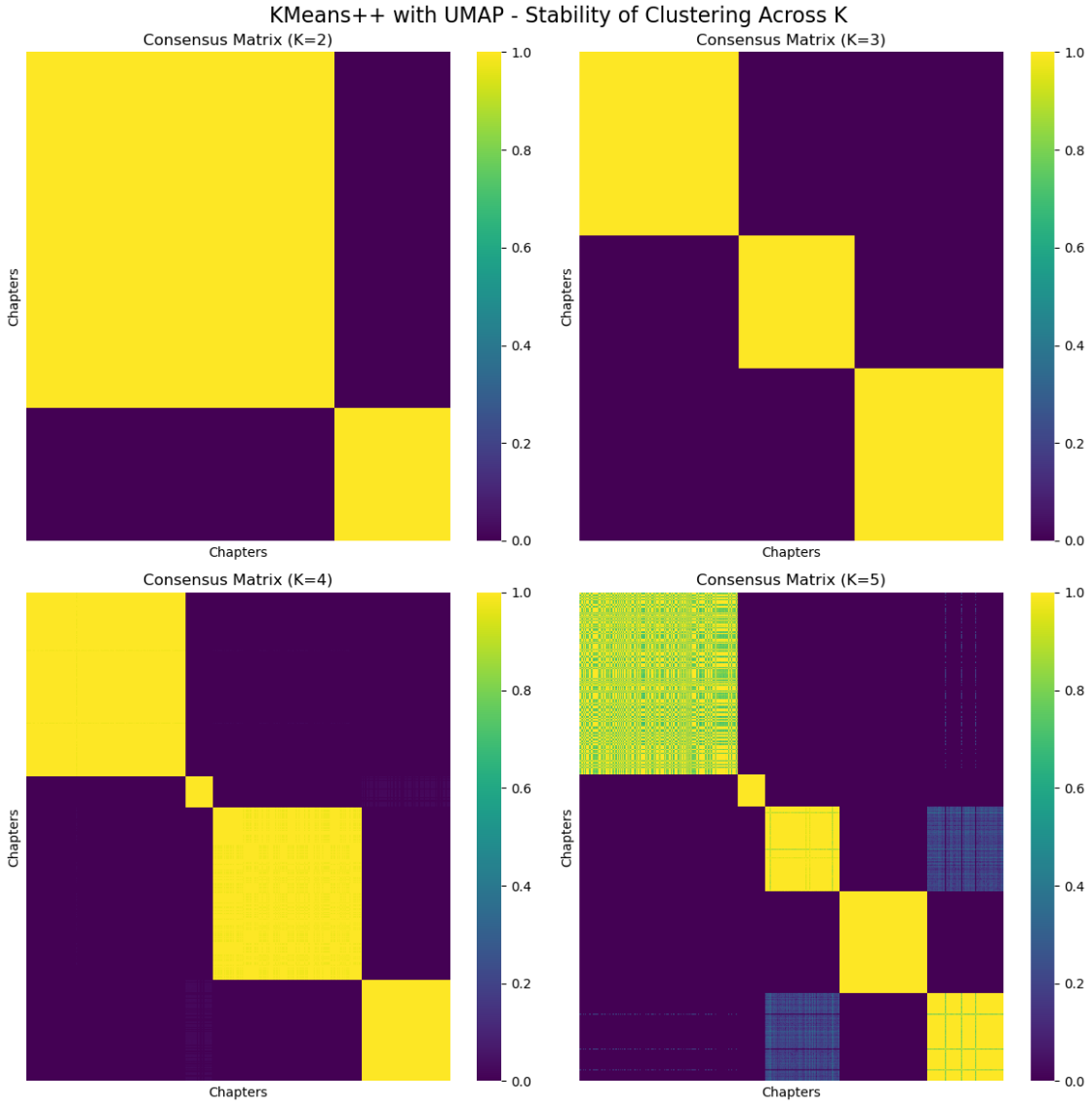
[12]:
```
method = 'kmeans++'

rng = np.random.default_rng(42)
iterations = 100
K_values = [2, 3, 4, 5]
n_samples = X.shape[0]
sample_names = X_clean.index.to_numpy()

fig, axes = plt.subplots(2, 2, figsize=(12, 12))
axes = axes.flatten()

for idx, K in enumerate(K_values):
    consensus_matrix = np.zeros((n_samples, n_samples))
    sampled_matrix = np.zeros((n_samples, n_samples))

    for n in range(iterations):
        train_idx = rng.choice(n_samples, size=int(0.7 * n_samples),␣
↪replace=False)
        X_train = X[train_idx]  # Using saved NumPy array
```

```
        km = KMeans(n_clusters=K, random_state=n, n_init=10)
        cluster_labels = km.fit_predict(X_train)

        sampled_matrix[np.ix_(train_idx, train_idx)] += 1
        co_members = np.equal.outer(cluster_labels, cluster_labels)
        consensus_matrix[np.ix_(train_idx, train_idx)] += co_members

    consensus_matrix = np.divide(consensus_matrix, sampled_matrix,␣
↪where=(sampled_matrix != 0))
    consensus_matrix = np.nan_to_num(consensus_matrix)

    final_labels = KMeans(n_clusters=K, random_state=42, n_init=10).
↪fit_predict(X)
    order_idx = np.argsort(final_labels)
    consensus_matrix = consensus_matrix[order_idx][:, order_idx]

    sns.heatmap(consensus_matrix, ax=axes[idx], cmap='viridis',
                xticklabels=sample_names[order_idx],␣
↪yticklabels=sample_names[order_idx])
    axes[idx].set_title(f'Consensus Matrix (K={K})')
    axes[idx].tick_params(axis='x', rotation=90, labelsize=8)
    axes[idx].tick_params(axis='y', labelsize=8)
    axes[idx].set_xticks([])
    axes[idx].set_yticks([])
    axes[idx].set_xlabel('Chapters')
    axes[idx].set_ylabel('Chapters')

fig.suptitle(f'{method} - Stability of Clustering Across K', fontsize=16)
plt.tight_layout()
plt.savefig('Media/viz/05/05_consensus_heatmaps_kmeans')
plt.show()
```



[13]:
```
method = 'KMeans++ with UMAP'
rng = np.random.default_rng(42)
iterations = 100
K_values = [2, 3, 4, 5]
n_samples = X_umap.shape[0]
sample_names = X_clean.index.to_numpy()

fig, axes = plt.subplots(2, 2, figsize=(12, 12))
axes = axes.flatten()

for idx, K in enumerate(K_values):
    consensus_matrix = np.zeros((n_samples, n_samples))
```

```
    sampled_matrix = np.zeros((n_samples, n_samples))

    for i in range(iterations):
        idx_sample = rng.choice(n_samples, size=int(0.7 * n_samples),␣
↪replace=False)
        X_sample = X_umap[idx_sample]

        km = KMeans(n_clusters=K, n_init=10, init='k-means++', random_state=i)
        labels = km.fit_predict(X_sample)

        sampled_matrix[np.ix_(idx_sample, idx_sample)] += 1
        co_members = np.equal.outer(labels, labels)
        consensus_matrix[np.ix_(idx_sample, idx_sample)] += co_members

    consensus_matrix = np.divide(consensus_matrix, sampled_matrix,␣
↪where=sampled_matrix != 0)
    consensus_matrix = np.nan_to_num(consensus_matrix)

    final_labels = KMeans(n_clusters=K, n_init=10, init='k-means++',␣
↪random_state=42).fit_predict(X_umap)
    order = np.argsort(final_labels)
    matrix_sorted = consensus_matrix[order][:, order]

    sns.heatmap(matrix_sorted, ax=axes[idx], cmap='viridis',
                xticklabels=sample_names[order],␣
↪yticklabels=sample_names[order])
    axes[idx].set_title(f'Consensus Matrix (K={K})')
    axes[idx].set_xticks([])
    axes[idx].set_yticks([])
    axes[idx].set_xlabel('Chapters')
    axes[idx].set_ylabel('Chapters')

fig.suptitle(f'{method} - Stability of Clustering Across K', fontsize=16)
plt.tight_layout()
plt.savefig('Media/viz/05/05_consensus_heatmaps_kmeans_umap')
plt.show()
```



1.2 Gaussian Mixture Models

1.2.1 Generalizibility

[14]:
```
# GMM
print('\033[1m' + 'GMM without dimension reduction:' + '\033[0m')
evaluator_gmm = SilhouetteEvaluator(X, make_gmm(), k_range=range(2, 11))
scores, best_k = evaluator_gmm.evaluate()
print(f'The scores across K = {scores}')
print(f"Best K: {best_k}")

# GMM with UMAP
print('\033[1m' + 'GMM with dimension reduction (UMAP):' + '\033[0m')
```

```
evaluator_gmm_umap = SilhouetteEvaluator(X_umap, make_gmm(), k_range=range(2,␣
↪11))
scores, best_k = evaluator_gmm_umap.evaluate()
print(f'The scores across K = {scores}')
print(f"Best K: {best_k}")

fig, axs = plt.subplots(1, 2, figsize=(14, 5))

evaluator_gmm.plot("GMM", ax=axs[0])
axs[0].set_ylim(y_min, y_max)

evaluator_gmm_umap.plot("GMM UMAP", ax=axs[1])
axs[1].set_ylim(y_min, y_max)

plt.tight_layout()

plt.savefig('Media/viz/05/05_gmm_silhouette_score')
plt.show()
```

GMM without dimension reduction:

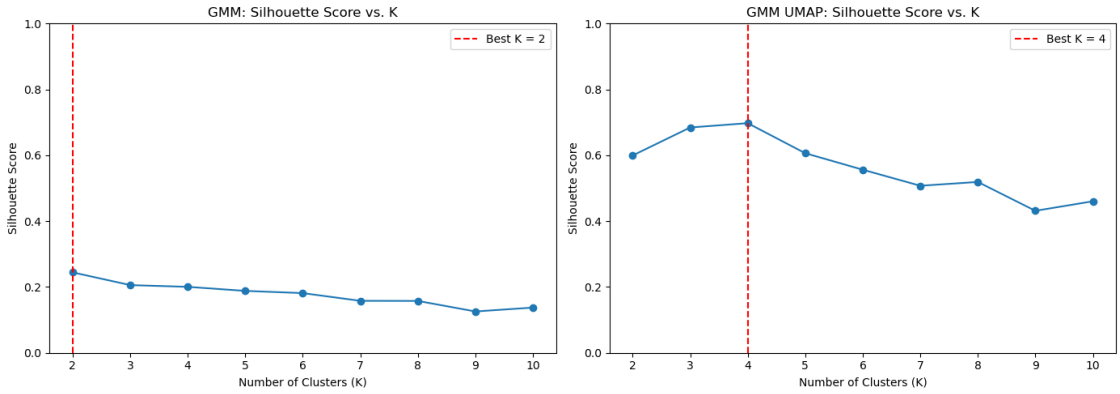
The scores across K = {2: 0.24434605026692022, 3: 0.20578434655866124, 4: 0.2004213780628659, 5: 0.18814970124375807, 6: 0.18136222277608166, 7: 0.15798484734742102, 8: 0.15765164283870803, 9: 0.1256388670716472, 10: 0.1374306014397366}

Best K: 2

GMM with dimension reduction (UMAP):

The scores across K = {2: 0.59943473, 3: 0.68435156, 4: 0.6975769, 5: 0.60627675, 6: 0.55619246, 7: 0.5074211, 8: 0.51900756, 9: 0.43136632, 10: 0.4602903}

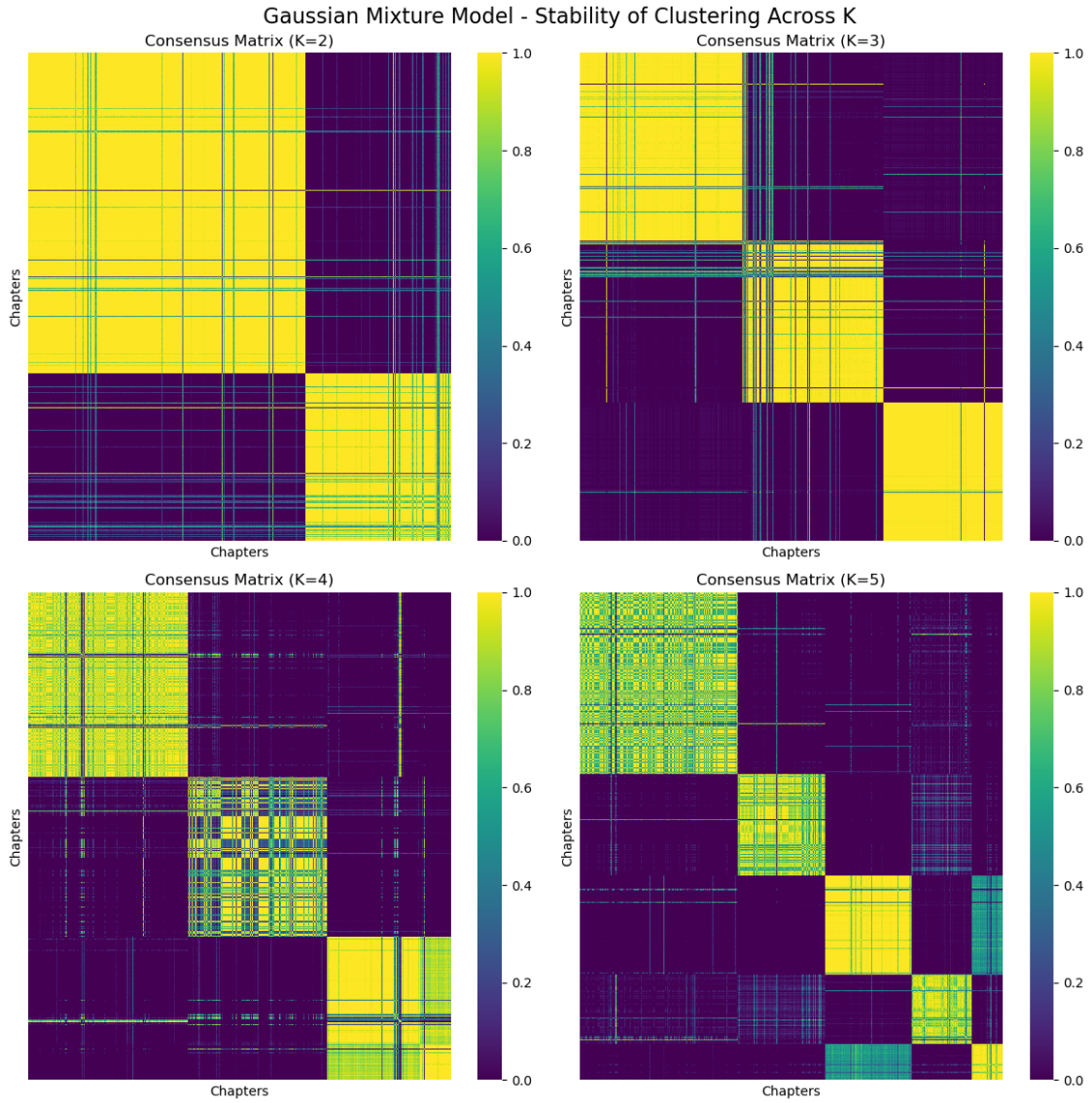
Best K: 4



1.2.2 Stability

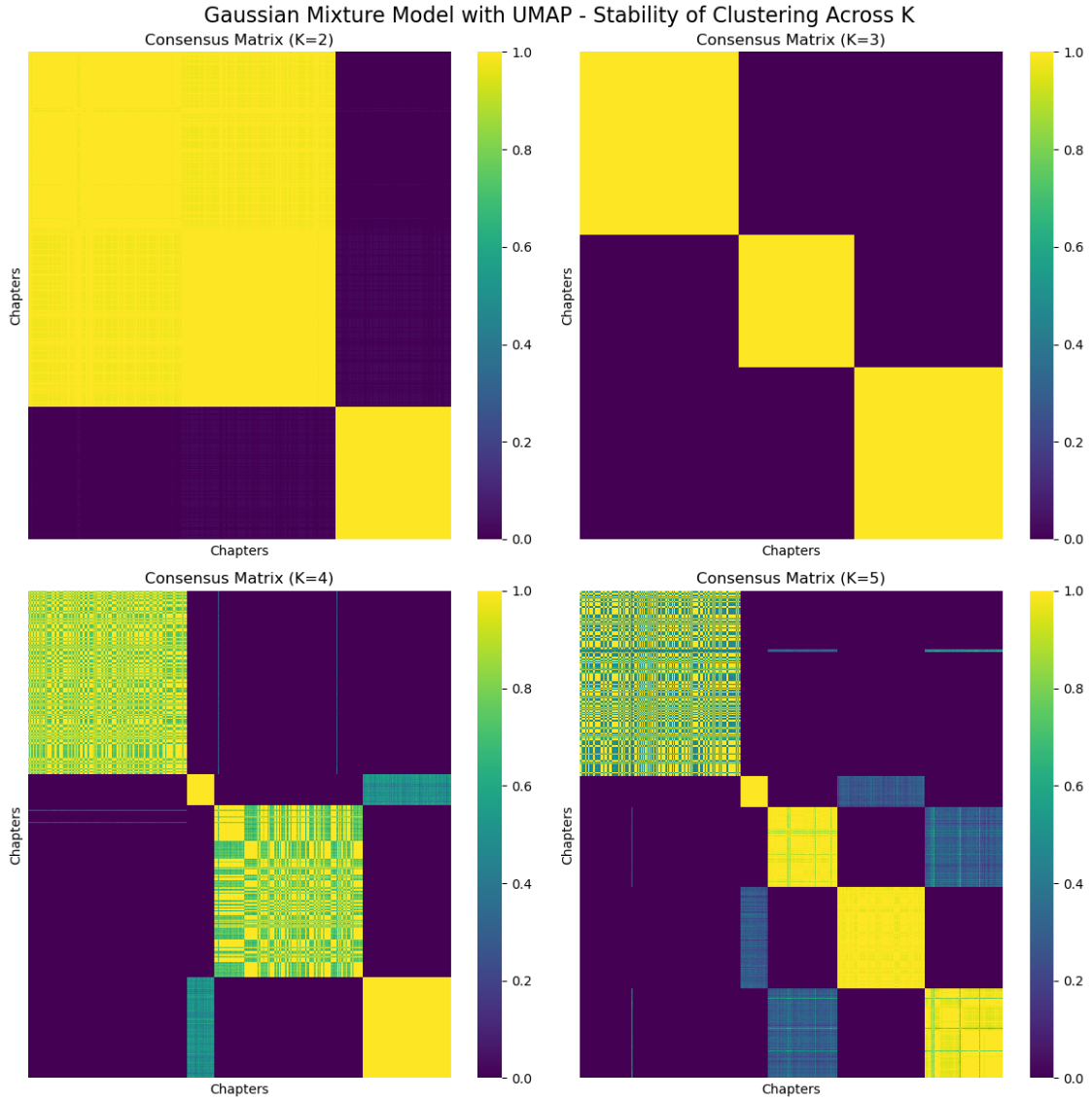
[15]:
```
method = 'Gaussian Mixture Model'
rng = np.random.default_rng(42)
iterations = 100
K_values = [2, 3, 4, 5]
n_samples = X.shape[0]
sample_names = X_clean.index.to_numpy()

fig, axes = plt.subplots(2, 2, figsize=(12, 12))
axes = axes.flatten()

for idx, K in enumerate(K_values):
    consensus = np.zeros((n_samples, n_samples))
    sampled = np.zeros((n_samples, n_samples))

    for i in range(iterations):
        idx_sample = rng.choice(n_samples, size=int(0.7 * n_samples),␣
↪replace=False)
        X_sample = X[idx_sample]  # NumPy version for GMM

        gmm = GaussianMixture(n_components=K, random_state=i)
        labels = gmm.fit(X_sample).predict(X_sample)

        sampled[np.ix_(idx_sample, idx_sample)] += 1
        co_members = np.equal.outer(labels, labels)
        consensus[np.ix_(idx_sample, idx_sample)] += co_members

    consensus = np.divide(consensus, sampled, where=sampled != 0)
    consensus = np.nan_to_num(consensus)

    final_labels = GaussianMixture(n_components=K, random_state=42).fit(X).
↪predict(X)
    order = np.argsort(final_labels)
    matrix_sorted = consensus[order][:, order]

    sns.heatmap(matrix_sorted, ax=axes[idx], cmap='viridis',
                xticklabels=sample_names[order],␣
↪yticklabels=sample_names[order])
    axes[idx].set_title(f'Consensus Matrix (K={K})')
    axes[idx].set_xticks([])
    axes[idx].set_yticks([])
    axes[idx].set_xlabel('Chapters')
    axes[idx].set_ylabel('Chapters')
    axes[idx].tick_params(axis='x', rotation=90, labelsize=8)
    axes[idx].tick_params(axis='y', labelsize=8)
```

```
fig.suptitle(f'{method} - Stability of Clustering Across K', fontsize=16)
plt.tight_layout()
plt.savefig('Media/viz/05/05_consensus_heatmaps_gmm')
plt.show()
```



[16]:
```
method = 'Gaussian Mixture Model with UMAP'
rng = np.random.default_rng(42)
iterations = 100
K_values = [2, 3, 4, 5]
n_samples = X_umap.shape[0]
sample_names = X_clean.index.to_numpy()
```

```
fig, axes = plt.subplots(2, 2, figsize=(12, 12))
axes = axes.flatten()

for idx, K in enumerate(K_values):
    consensus = np.zeros((n_samples, n_samples))
    sampled = np.zeros((n_samples, n_samples))

    for i in range(iterations):
        idx_sample = rng.choice(n_samples, size=int(0.7 * n_samples),␣
 ↪replace=False)
        X_sample = X_umap[idx_sample]

        gmm = GaussianMixture(n_components=K, random_state=i)
        labels = gmm.fit(X_sample).predict(X_sample)

        sampled[np.ix_(idx_sample, idx_sample)] += 1
        co_members = np.equal.outer(labels, labels)
        consensus[np.ix_(idx_sample, idx_sample)] += co_members

    consensus = np.divide(consensus, sampled, where=sampled != 0)
    consensus = np.nan_to_num(consensus)

    final_labels = GaussianMixture(n_components=K, random_state=42).fit(X_umap).
 ↪predict(X_umap)
    order = np.argsort(final_labels)
    matrix_sorted = consensus[order][:, order]

    sns.heatmap(matrix_sorted, ax=axes[idx], cmap='viridis',
                xticklabels=sample_names[order],␣
 ↪yticklabels=sample_names[order])
    axes[idx].set_title(f'Consensus Matrix (K={K})')
    axes[idx].set_xticks([])
    axes[idx].set_yticks([])
    axes[idx].set_xlabel('Chapters')
    axes[idx].set_ylabel('Chapters')
    axes[idx].tick_params(axis='x', rotation=90, labelsize=8)
    axes[idx].tick_params(axis='y', labelsize=8)

fig.suptitle(f'{method} - Stability of Clustering Across K', fontsize=16)
plt.tight_layout()
plt.savefig('Media/viz/05/05_consensus_heatmaps_gmm_umap')
plt.show()
```



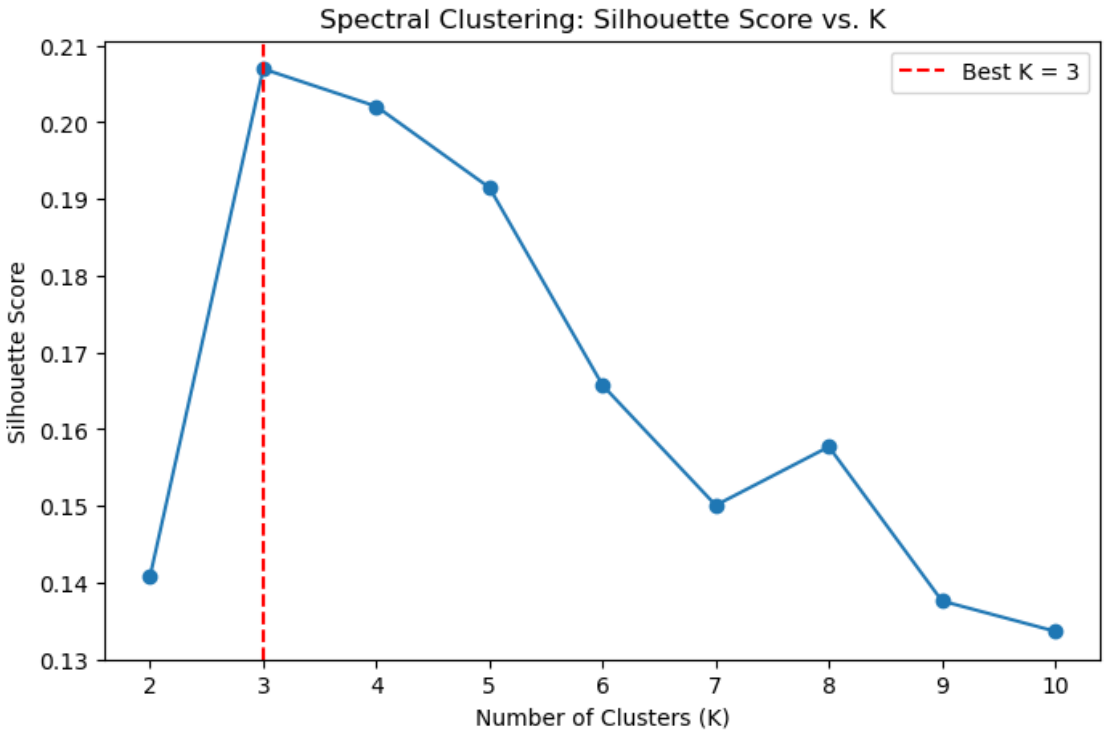
1.3 Spectral Clustering

1.3.1 Generalizibility

[17]:
```
print('\033[1m' + 'Spectral Clustering:' + '\033[0m')
evaluator_spectral = SilhouetteEvaluator(X, make_spectral(affinity =␣
 ↪'nearest_neighbors'), k_range=range(2, 11))
scores, best_k = evaluator_spectral.evaluate()
print(f'The scores across K = {scores}')
print(f"Best K: {best_k}")

evaluator_spectral.plot('Spectral Clustering')
```

```
plt.savefig('Media/viz/05/05_spectral_silhouette_score')
```

```
Spectral Clustering:
The scores across K = {2: 0.1408300984760483, 3: 0.20696881798137212, 4:
0.2020762541851448, 5: 0.1914837165759898, 6: 0.16570630789830376, 7:
0.15006558354299535, 8: 0.1577000857169114, 9: 0.13758976585902763, 10:
0.1336400256361507}
Best K: 3
```



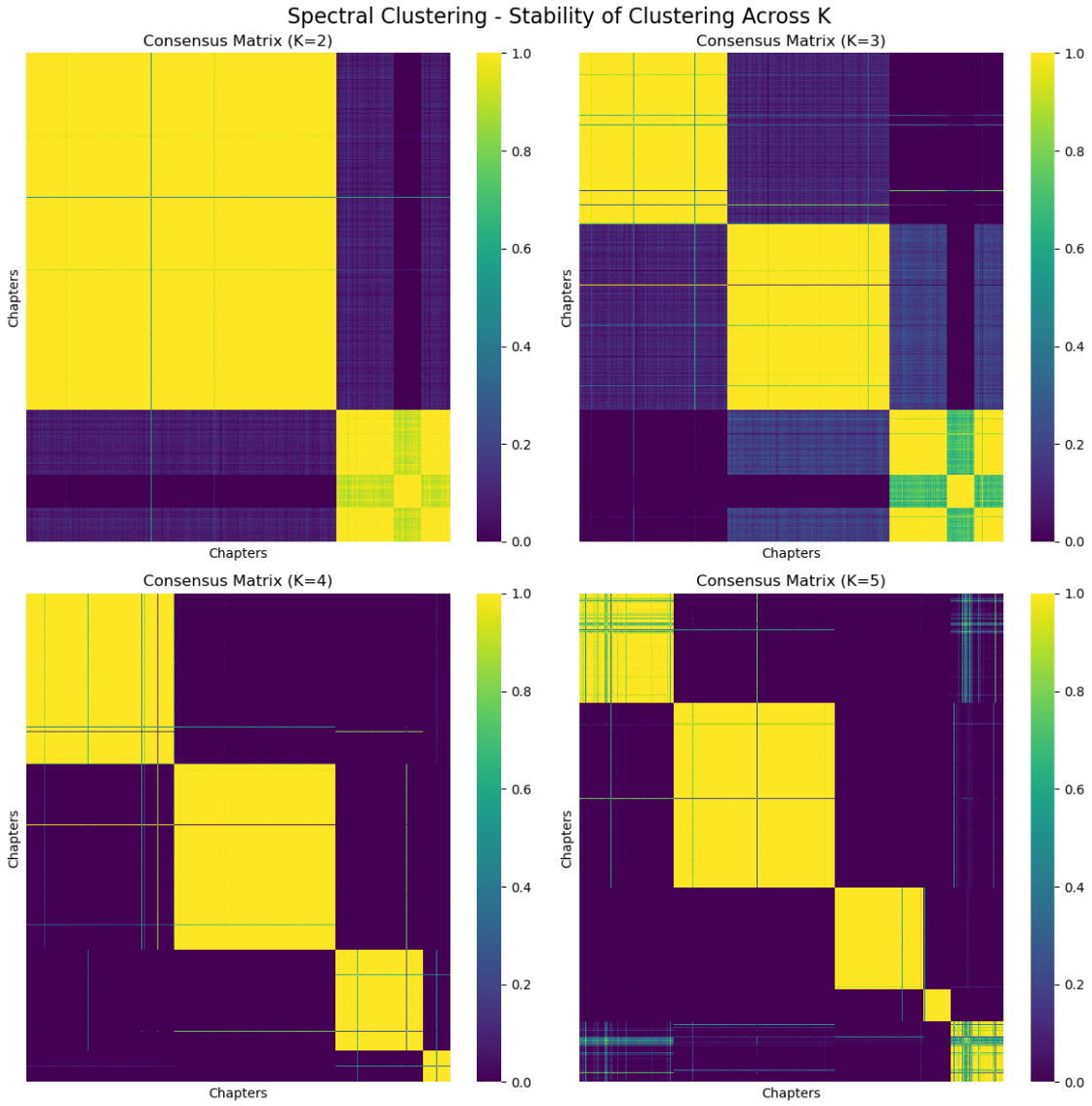
Although the silhouette score is commonly used to evaluate clustering performance, it assumes that clusters are spherical in shape, which does not align with the strengths of spectral clustering. Spectral clustering excels at uncovering non-linear and arbitrarily shaped cluster structures, which silhouette score is not well-equipped to quantify. Therefore, the low silhouette score observed here may understate the actual performance of spectral clustering. In fact, the spectral embedding reveals visually distinct and meaningful clusters (see notebook 02), suggesting that the method is effective despite the numerical metric.

1.3.2 Stability

[18]:
```
## METHOD
method = 'Spectral Clustering'

rng = np.random.default_rng(42)
iterations = 100
```

```
K_values = [2, 3, 4, 5]
n_samples = X_clean.shape[0]
sample_names = X_clean.index.to_numpy()

fig, axes = plt.subplots(2, 2, figsize=(12, 12))
axes = axes.flatten()

for idx, K in enumerate(K_values):
    consensus_matrix = np.zeros((n_samples, n_samples))
    sampled_matrix = np.zeros((n_samples, n_samples))

    for n in range(iterations):
        train_idx = rng.choice(n_samples, size=int(0.7 * n_samples),␣
 ↪replace=False)
        X_train = X_clean.iloc[train_idx].to_numpy()

        sc = SpectralClustering(n_clusters=K, affinity='nearest_neighbors',␣
 ↪assign_labels='kmeans', random_state=n)
        cluster_labels = sc.fit_predict(X_train)

        sampled_matrix[np.ix_(train_idx, train_idx)] += 1
        co_members = np.equal.outer(cluster_labels, cluster_labels)
        consensus_matrix[np.ix_(train_idx, train_idx)] += co_members

    consensus_matrix = np.divide(consensus_matrix, sampled_matrix,␣
 ↪where=(sampled_matrix != 0))
    consensus_matrix = np.nan_to_num(consensus_matrix)

    final_labels = SpectralClustering(n_clusters=K,␣
 ↪affinity='nearest_neighbors', assign_labels='kmeans', random_state=42).
 ↪fit_predict(X_clean)
    order_idx = np.argsort(final_labels)
    consensus_matrix = consensus_matrix[order_idx][:, order_idx]

    sns.heatmap(consensus_matrix, ax=axes[idx], cmap='viridis',
                xticklabels=sample_names[order_idx],␣
 ↪yticklabels=sample_names[order_idx])
    axes[idx].set_title(f'Consensus Matrix (K={K})')
    axes[idx].tick_params(axis='x', rotation=90, labelsize=8)
    axes[idx].tick_params(axis='y', labelsize=8)
    axes[idx].set_xticks([])
    axes[idx].set_yticks([])
    axes[idx].set_xlabel('Chapters')
    axes[idx].set_ylabel('Chapters')

fig.suptitle(f'{method} - Stability of Clustering Across K', fontsize=16)
```

```
plt.tight_layout()
plt.savefig('Media/viz/05/05_consensus_heatmaps_spectral')
plt.show()
```



$K = 4$ is the clear winner!

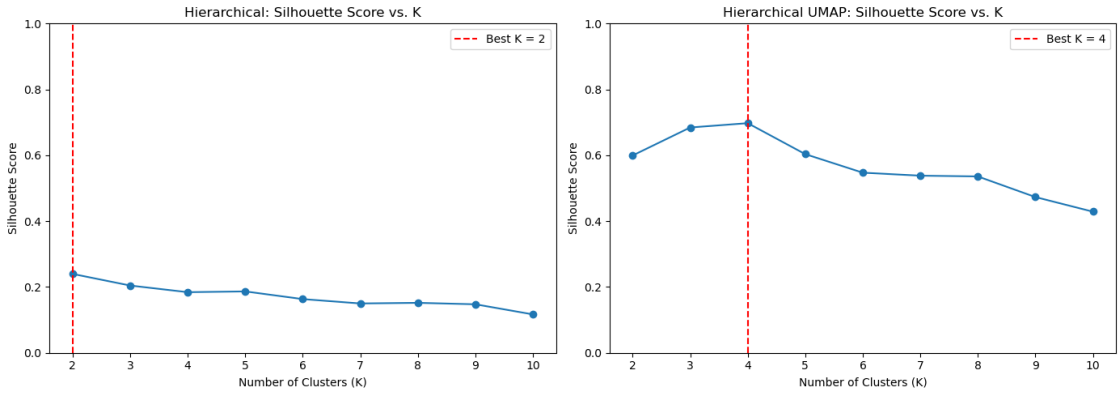
1.4 Hierarchial Clustering

Using the parameters 'ward' and 'euclidean' (because we have seen in notebook 04 that these have the highest accuracy)!

1.4.1 Generalizibility

[19]:
```
# Hierarchical Clustering
print('\033[1m' + 'Hierarchical Clustering without dimension reduction:' +␣
 ↪'\033[0m')
evaluator_hier = SilhouetteEvaluator(X, make_hierarchical(linkage='ward',␣
 ↪metric='euclidean'), k_range=range(2, 11))
scores, best_k = evaluator_hier.evaluate()
print(f'The scores across K = {scores}')
print(f"Best K: {best_k}")

# Hierarchical Clustering with UMAP
print('\033[1m' + 'Hierarchical Clustering with dimension reduction (UMAP):' +␣
 ↪'\033[0m')
evaluator_hier_umap = SilhouetteEvaluator(X_umap,␣
 ↪make_hierarchical(linkage='ward', metric='euclidean'), k_range=range(2, 11))
scores, best_k = evaluator_hier_umap.evaluate()
print(f'The scores across K = {scores}')
print(f"Best K: {best_k}")

fig, axs = plt.subplots(1, 2, figsize=(14, 5))

evaluator_hier.plot("Hierarchical", ax=axs[0])
axs[0].set_ylim(y_min, y_max)

evaluator_hier_umap.plot("Hierarchical UMAP", ax=axs[1])
axs[1].set_ylim(y_min, y_max)

plt.tight_layout()
plt.savefig('Media/viz/05/05_silhouette_score_across_methods.png')
plt.show()
```

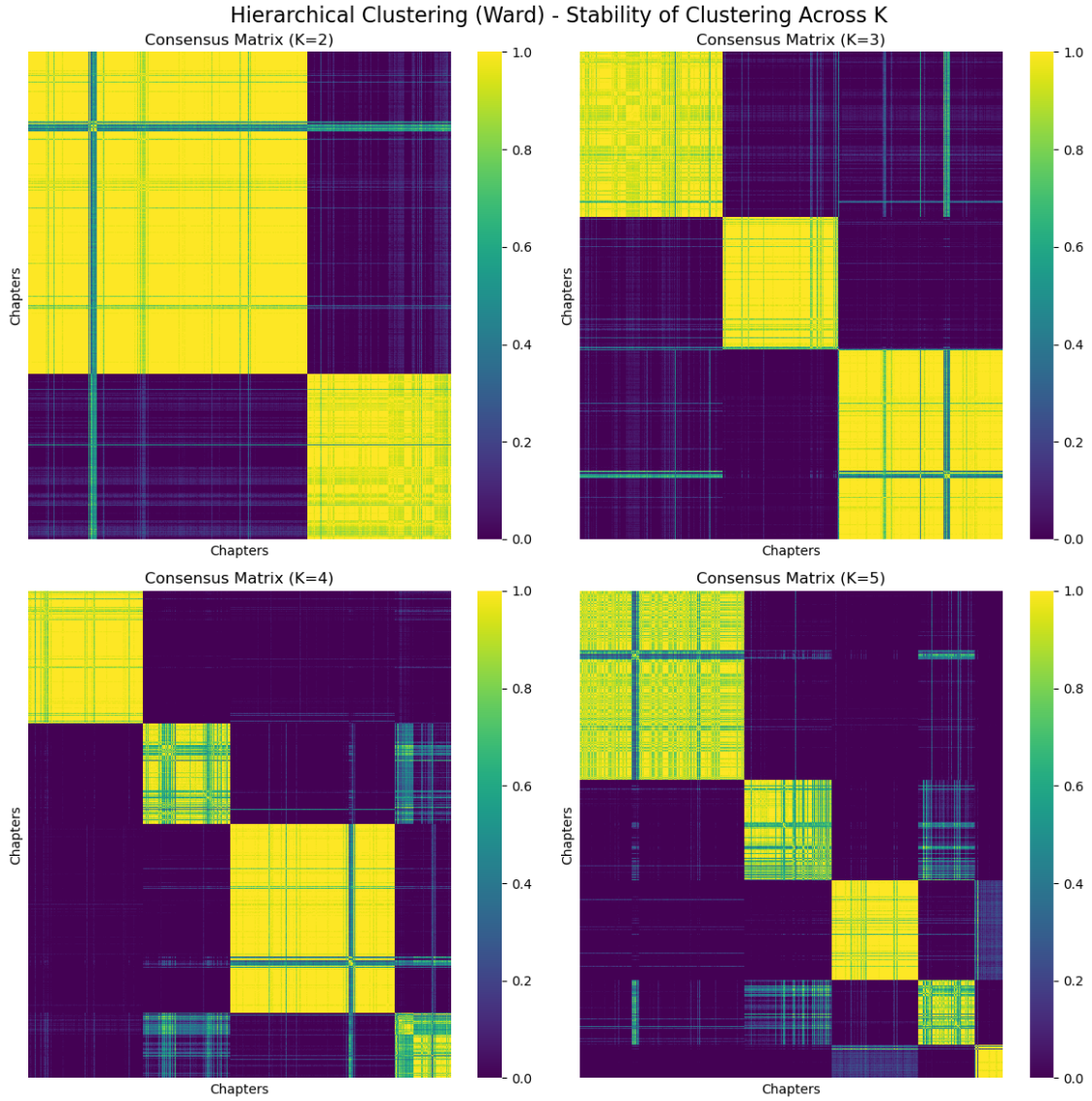
```
Hierarchical Clustering without dimension reduction:
The scores across K = {2: 0.23967574687100793, 3: 0.20437373269232859, 4:
0.18421197778513682, 5: 0.18654800991576861, 6: 0.16319550456880302, 7:
0.14995813568113975, 8: 0.1518500219855489, 9: 0.14745055233611987, 10:
0.1167264049290792}
Best K: 2
Hierarchical Clustering with dimension reduction (UMAP):
The scores across K = {2: 0.59943473, 3: 0.68435156, 4: 0.6975769, 5:
0.60349095, 6: 0.5471597, 7: 0.5380937, 8: 0.53579855, 9: 0.47313103, 10:
0.42873582}
Best K: 4
```



1.4.2 Stability

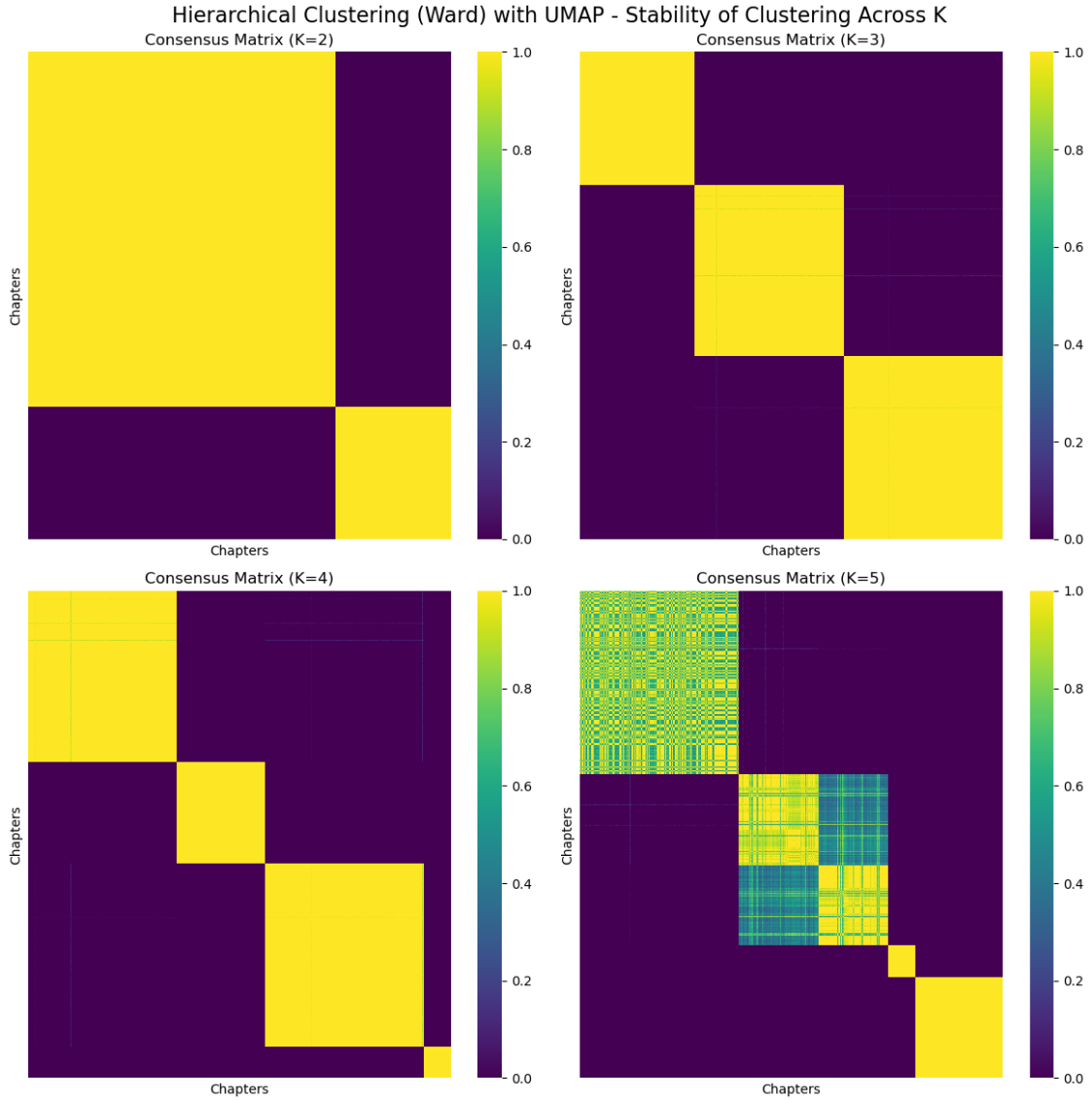
[20]:
```
method = 'Hierarchical Clustering (Ward)'
rng = np.random.default_rng(42)
iterations = 100
K_values = [2, 3, 4, 5]
n_samples = X.shape[0]
sample_names = X_clean.index.to_numpy()

fig, axes = plt.subplots(2, 2, figsize=(12, 12))
axes = axes.flatten()

for idx, K in enumerate(K_values):
    consensus = np.zeros((n_samples, n_samples))
    sampled = np.zeros((n_samples, n_samples))

    for i in range(iterations):
        idx_sample = rng.choice(n_samples, size=int(0.7 * n_samples),␣
↪replace=False)
        X_sample = X[idx_sample]

        model = AgglomerativeClustering(n_clusters=K, linkage='ward')
        labels = model.fit_predict(X_sample)

        sampled[np.ix_(idx_sample, idx_sample)] += 1
        co_members = np.equal.outer(labels, labels)
        consensus[np.ix_(idx_sample, idx_sample)] += co_members

    consensus = np.divide(consensus, sampled, where=sampled != 0)
    consensus = np.nan_to_num(consensus)

    final_labels = AgglomerativeClustering(n_clusters=K, linkage='ward').
↪fit_predict(X)
```

```
    order = np.argsort(final_labels)
    matrix_sorted = consensus[order][:, order]

    sns.heatmap(matrix_sorted, ax=axes[idx], cmap='viridis',
                xticklabels=sample_names[order],␣
 ↪yticklabels=sample_names[order])
    axes[idx].set_title(f'Consensus Matrix (K={K})')
    axes[idx].set_xticks([])
    axes[idx].set_yticks([])
    axes[idx].set_xlabel('Chapters')
    axes[idx].set_ylabel('Chapters')
    axes[idx].tick_params(axis='x', rotation=90, labelsize=8)
    axes[idx].tick_params(axis='y', labelsize=8)

fig.suptitle(f'{method} - Stability of Clustering Across K', fontsize=16)
plt.tight_layout()
plt.savefig('Media/viz/05/05_consensus_heatmaps_hierarchical')
plt.show()
```



[21]:
```
method = 'Hierarchical Clustering (Ward) with UMAP'
rng = np.random.default_rng(42)
iterations = 100
K_values = [2, 3, 4, 5]
n_samples = X_umap.shape[0]
sample_names = X_clean.index.to_numpy()

fig, axes = plt.subplots(2, 2, figsize=(12, 12))
axes = axes.flatten()

for idx, K in enumerate(K_values):
    consensus = np.zeros((n_samples, n_samples))
```

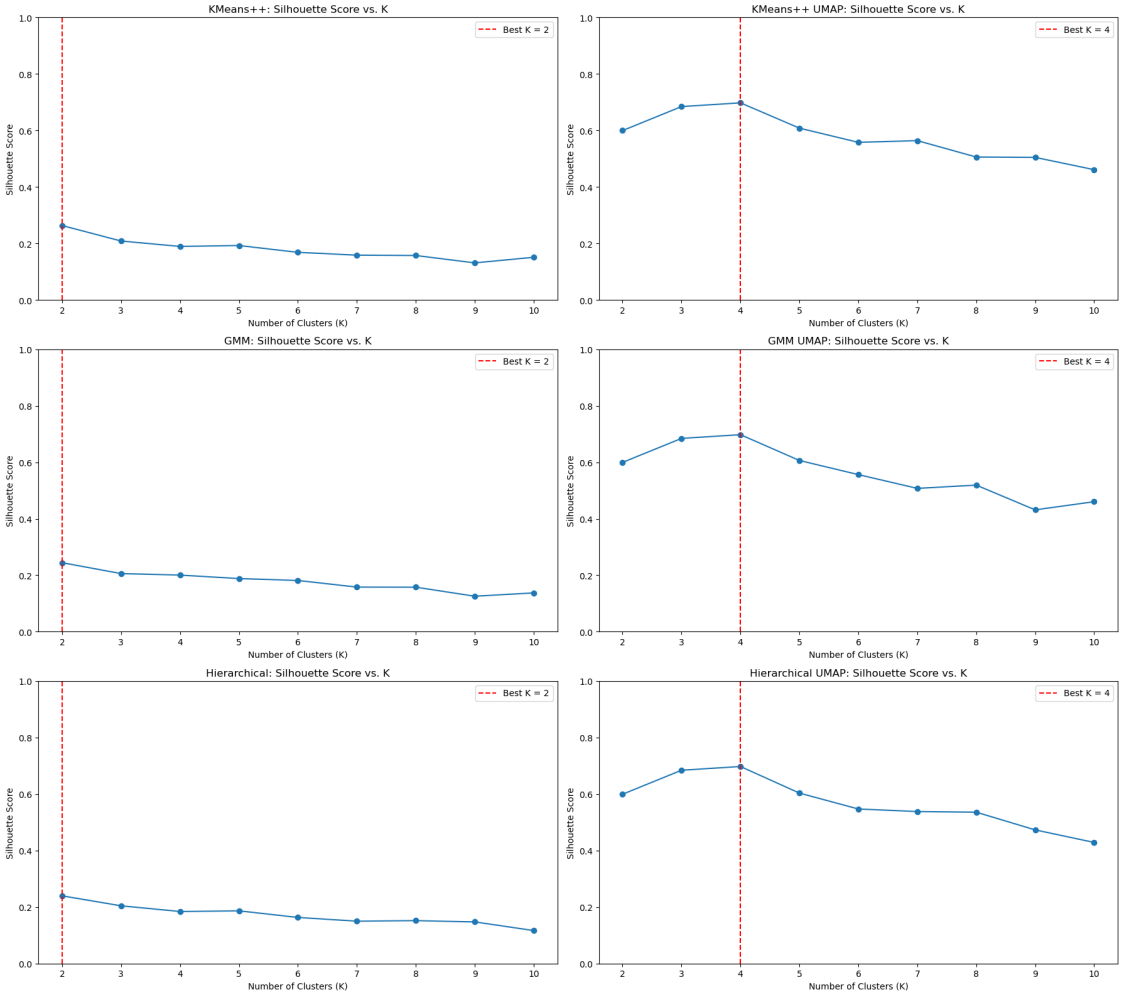
```
    sampled = np.zeros((n_samples, n_samples))

    for i in range(iterations):
        idx_sample = rng.choice(n_samples, size=int(0.7 * n_samples),␣
↪replace=False)
        X_sample = X_umap[idx_sample]

        model = AgglomerativeClustering(n_clusters=K, linkage='ward')
        labels = model.fit_predict(X_sample)

        sampled[np.ix_(idx_sample, idx_sample)] += 1
        co_members = np.equal.outer(labels, labels)
        consensus[np.ix_(idx_sample, idx_sample)] += co_members

    consensus = np.divide(consensus, sampled, where=sampled != 0)
    consensus = np.nan_to_num(consensus)

    final_labels = AgglomerativeClustering(n_clusters=K, linkage='ward').
↪fit_predict(X_umap)
    order = np.argsort(final_labels)
    matrix_sorted = consensus[order][:, order]

    sns.heatmap(matrix_sorted, ax=axes[idx], cmap='viridis',
                xticklabels=sample_names[order],␣
↪yticklabels=sample_names[order])
    axes[idx].set_title(f'Consensus Matrix (K={K})')
    axes[idx].set_xticks([])
    axes[idx].set_yticks([])
    axes[idx].set_xlabel('Chapters')
    axes[idx].set_ylabel('Chapters')
    axes[idx].tick_params(axis='x', rotation=90, labelsize=8)
    axes[idx].tick_params(axis='y', labelsize=8)

fig.suptitle(f'{method} - Stability of Clustering Across K', fontsize=16)
plt.tight_layout()
plt.savefig('Media/viz/05/05_consensus_heatmaps_hierarchical_umap')
plt.show()
```



2 Comparison Viz

[22]:
```
fig, axs = plt.subplots(3, 2, figsize=(18, 16))

evaluator_kmeans.plot("KMeans++", ax=axs[0,0])
axs[0,0].set_ylim(y_min, y_max)
evaluator_kmeans_umap.plot("KMeans++ UMAP", ax=axs[0,1])
axs[0,1].set_ylim(y_min, y_max)

evaluator_gmm.plot("GMM", ax=axs[1,0])
axs[1,0].set_ylim(y_min, y_max)
```

```
evaluator_gmm_umap.plot("GMM UMAP", ax=axs[1,1])
axs[1,1].set_ylim(y_min, y_max)

evaluator_hier.plot("Hierarchical", ax=axs[2,0])
axs[2,0].set_ylim(y_min, y_max)

evaluator_hier_umap.plot("Hierarchical UMAP", ax=axs[2,1])
axs[2,1].set_ylim(y_min, y_max)

plt.savefig('Media/viz/05/05_silhouette_score_across_methods')
plt.tight_layout()
```



[26]:
```
heatmap_paths = [
    "Media/viz/05/05_consensus_heatmaps_kmeans.png",
    "Media/viz/05/05_consensus_heatmaps_gmm.png",
```

```
    "Media/viz/05/05_consensus_heatmaps_hierarchical.png",
    "Media/viz/05/05_consensus_heatmaps_kmeans_umap.png",
    "Media/viz/05/05_consensus_heatmaps_gmm_umap.png",
    "Media/viz/05/05_consensus_heatmaps_hierarchical_umap.png"
]

heatmap_images = [Image.open(path) for path in heatmap_paths]

fig, axes = plt.subplots(2, 3, figsize=(20, 14), constrained_layout=True)
axes = axes.flatten()
for ax, img, path in zip(axes, heatmap_images, heatmap_paths):
    ax.imshow(img)
    ax.set_title(
        os.path.basename(path)
          .replace("05_consensus_heatmaps_", "")
          .replace(".png", "")
          .replace("_", " ")
          .title(),
        fontsize=12
    )
    ax.axis('off')
plt.savefig("Media/viz/05/05_all_consensus_heatmaps_grid.png")
plt.show()
```