Applying Clustering Algorithms for Topic Modeling

Objective

To understand and apply clustering algorithms such as Latent Dirichlet Allocation (LDA) and K-Means to group similar documents for topic modeling and gaining insights from large collections of text data.

1. Introduction

In the age of big data, processing and extracting meaningful insights from large text corpora has become crucial. Topic modeling and document clustering are widely used techniques to summarize and categorize text. Two powerful methods for such tasks are:

- Latent Dirichlet Allocation (LDA): a probabilistic model for topic modeling.
- K-Means Clustering: an unsupervised algorithm for grouping documents based on content similarity.

2. Dataset

We use the 20 Newsgroups dataset, a collection of approximately 20,000 newsgroup documents across 20 categories.

3. Preprocessing Steps

Before applying clustering algorithms, the data undergoes:
1. Lowercasing
2. Tokenization
3. Stopword Removal
4. Lemmatization
5. Vectorization using CountVectorizer (for LDA)

4. Latent Dirichlet Allocation (LDA)

```python
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation

# Vectorize the corpus
vectorizer = CountVectorizer(max_df=0.9, min_df=2, stop_words='english')
doc_term_matrix = vectorizer.fit_transform(documents)

# Apply LDA
lda = LatentDirichletAllocation(n_components=5, random_state=42)
lda.fit(doc_term_matrix)

# Display topics
for idx, topic in enumerate(lda.components_):
    print(f"Topic {idx}:")
    print([vectorizer.get_feature_names_out()[i] for i in topic.argsort()[-10:]])
```

5. K-Means Clustering

K-Means groups similar documents into K clusters based on vectorized features.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans

# Vectorize with TF-IDF
vectorizer = TfidfVectorizer(stop_words='english')
X = vectorizer.fit_transform(documents)

# Apply KMeans
k = 5
kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(X)

# Print top keywords per cluster
order_centroids = kmeans.cluster_centers_.argsort()[:, ::-1]
terms = vectorizer.get_feature_names_out()

for i in range(k):
    print(f"Cluster {i}:")
    print([terms[ind] for ind in order_centroids[i, :10]])
```

6. Comparison and Conclusion

```
+------------------+--------------------------+-----------------------------+
| Aspect           | LDA                      | K-Means                     |
+------------------+--------------------------+-----------------------------+
| Type             | Probabilistic            | Distance-based              |
| Output           | Topics from word dists.  | Groups of documents         |
| Interpretability | High                     | Moderate                    |
| Best for         | Topic Modeling           | Document Clustering         |
+------------------+--------------------------+-----------------------------+
```

Conclusion: LDA is ideal for discovering topics, while K-Means effectively groups documents. Together, they provide a robust solution for text analysis.