Semantic-Driven Topic Modeling Using Transformer-Based Embeddings and DBSCAN Clustering

نصب نیازمندی ها شامل کتابخانه های اصلی، گرافیکی و داده ها:

- sentence-transformers, scikit-learn, hdbscan, umap-learn, pandas, numpy
- matplotlib , seaborn
- datasets

```
pip install sentence-transformers scikit-learn hdbscan umap-learn pandas numpy matplotlib seaborn datasets
```

بارگذاری داده‌ها و اجرای مدل:

برای استفاده از مدل و انجام آموزش/آزمایش، ابتدا باید داده‌ها را بارگذاری کرده و مدل را آموزش دهید. این کد نمونه برای بارگذاری داده‌ها از **دیتاست AG News** استفاده می‌کند.

```
from datasets import load_dataset
dataset = load_dataset("ag_news", split="train[:500]")
documents = dataset["text"]
true_labels = dataset["label"]
print(f"Data loaded: {len(documents)} documents")
```

آموزش مدل SemanticTopicModelDBSCAN:

```
from semantic_topic_model_dbscan import SemanticTopicModelDBSCAN
stm_dbscan = SemanticTopicModelDBSCAN(n_topics=3, min_cluster_size=25)
dbscan_topics = stm_dbscan.fit_transform(documents)
```

ارزیابی مدل:

مدل شما می‌تواند با استفاده از ارزیابی‌هایی مانند Cosine Similarity یا Semantic Coherence ارزیابی شود. این ارزیابی‌ها به شما کمک می‌کنند تا کیفیت خوشه‌بندی‌های مدل را اندازه‌گیری کنید.

ارزیابی انسجام معنایی واژگان:

- این مرحله به‌طور **مستقیم روی واژگان هر موضوع** انجام می‌شود.
- برای هر خوشه (یا موضوع) مجموعه‌ای از کلمات استخراج‌شده داریم.
- **شباهت کسینوسی** بین بردارهای معنایی این کلمات محاسبه می‌شود.
- سپس میانگین شباهت‌ها برای آن خوشه به‌دست می‌آید.
- این مقدار **نشان‌دهنده میزان نزدیکی معنایی کلمات داخل همان خوشه** است.

```
from sklearn.metrics.pairwise import cosine_similarity
def calculate_semantic_coherence(topics_dict, model):
   scores[] =
   for topic, words in topics_dict.items:()
      if not words: continue
      embeddings = model.encode(words)
      sim_matrix = cosine_similarity(embeddings)
      avg_sim = (np.sum(sim_matrix) - len(words)) / (len(words) * (len(words)-1))
      scores.append(avg_sim)
   return np.mean(scores)
```

محاسبه انسجام معنایی مدل:

- در این مرحله، خروجی مرحله قبل یعنی **مقادیر انسجام هر موضوع** را برای تمام خوشه‌ها/موضوعات مدل **میانگین‌گیری** می‌کنیم.
- نتیجه یک عدد واحد است که **کل مدل و کیفیت خوشه‌بندی معنایی آن** را ارزیابی می‌کند.

```
stm_score = calculate_semantic_coherence(dbscan_topics, stm_dbscan.model)
print(f"Semantic Coherence Score: {stm_score:.4f}")
```

نمایش نتایج:

```
import matplotlib.pyplot as plt
import seaborn as sns
def plot_semantic_clusters(model_instance):
```

کاهش ابعاد به ۲ بعد برای نمایش بصری:

```
    umap_2d = umap.UMAP(n_neighbors=15, n_components=2, metric="cosine",
random_state=42).fit_transform(model_instance.doc_embeddings)
```

نمایش نمودار:

```
    plt.figure(figsize=(12, 8))
    sns.scatterplot(x=umap_2d[:, 0], y=umap_2d[:, 1], hue=model_instance.labels, palette="viridis")
    plt.title('Topic Clusters Visualized with UMAP')
    plt.xlabel('UMAP-1')
    plt.ylabel('UMAP-2')
    plt.show()
```

نمایش نمودار خوشه‌بندی:

```
plot_semantic_clusters(stm_dbscan)
```

- **قابل اجرا در با CPU , GPU**

مثال اجرای مدل روی فایل ورودی نمونه:

برای اجرای مدل روی فایل ورودی دلخواه، کافی است فایل متنی خود را بارگذاری کنید و آن را به مدل بدهید.

بارگذاری داده‌ها از فایل

```
import pandas as pd
```

فرض کنید فایل ورودی به نام input_data.csv که حاوی ستون متنی است

```
df = pd.read_csv('input_data.csv')
documents = df['text'].tolist()
```

اجرای مدل روی داده‌های جدید

```
stm_dbscan = SemanticTopicModelDBSCAN(n_topics=3, min_cluster_size=25)
dbscan_topics = stm_dbscan.fit_transform(documents)
```