Semantic Search and Topic Modeling on Policy Documents

This report presents a complete walkthrough of a semantic search and topic modeling pipeline applied to a synthetic dataset of 200 policy-related sentences. The goal is to help you intuitively and technically understand how semantic embeddings and clustering can be used to uncover structure in unstructured political texts.

---

1. Introduction

Policy documents often contain nuanced expressions that are hard to analyze with simple keyword matching. Our approach uses **sentence embeddings** and **topic modeling** to capture semantic similarities and identify recurring themes.

We leverage:

- **SentenceTransformer**: For encoding sentences into high-dimensional vectors
- **BERTopic**: For clustering semantically similar documents
- **UMAP + HDBSCAN**: For unsupervised topic extraction
- **Optional**: Summarization and topic labeling for better interpretation

---

2. Sentence Embedding with SentenceTransformer

We use `all-mpnet-base-v2`, a pre-trained transformer model optimized for sentence-level representations.

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer('all-mpnet-base-v2')
embeddings = model.encode(documents, show_progress_bar=True)
```

🤔 Why Embeddings?

Embedding models map sentences into a vector space where semantically similar sentences are closer. For example, "subsidies for solar panels" and "support for renewable energy" may have similar vector representations even if they share few words.

---

3. Topic Modeling with BERTopic

BERTopic builds on top of embeddings to group similar documents and extract the top keywords per topic. It uses:

- **UMAP**: Reduces the dimensionality of embeddings for clustering

- **HDBSCAN**: Groups nearby points into clusters (topics)

```
from bertopic import BERTopic
from umap import UMAP
from hdbscan import HDBSCAN

umap_model = UMAP(n_neighbors=10, n_components=5, min_dist=0.0,
metric='cosine')
hdbscan_model = HDBSCAN(min_cluster_size=5, metric='euclidean')

topic_model = BERTopic(embedding_model=model,
                       umap_model=umap_model,
                       hdbscan_model=hdbscan_model,
                       language="english")
topics, probs = topic_model.fit_transform(documents)
```

---

4. Understanding Topics

```
topic_info = topic_model.get_topic_info()
topic_info.head()
```

📌 Interpretation

Each topic includes:

- `Topic`: Numeric ID
- `Count`: Number of documents in the topic
- `Name`/`Representation`: Keywords summarizing the topic
- `Representative_Docs`: Top examples for the topic

---

5. Visualizing Topics

BERTopic supports interactive visualization:

```
topic_model.visualize_topics()
topic_model.visualize_barchart(top_n_topics=10)
topic_model.visualize_documents(documents)
```

These help explore the density and separability of topics.

---

6. Optional: Topic Merging

```
merged_model = topic_model.merge_topics(documents,
similarity_threshold=0.8)
```

You can reduce noise and redundancy by merging similar topics using cosine similarity.

---

7. Optional: Topic Summarization

We can use `transformers` to summarize the representative documents for each topic.

```
from transformers import pipeline

summarizer = pipeline("summarization", model="facebook/bart-large-cnn")
summary = summarizer(" ".join(docs[:2]), max_length=80, min_length=30,
do_sample=False)[0]['summary_text']
```

---

8. Project Output

- Jupyter Notebook (`.ipynb`)
- English & Turkish markdown reports
- PDF exports
- GitHub-ready README

---

9. Conclusion

This project demonstrates how modern NLP tools can be combined to extract interpretable structure from political or economic documents. It's ideal for use cases in policy research, social science, and government transparency.