Semantic Search on Policy Documents using Transformers

🔍 Project Overview

This project demonstrates how to perform **semantic search** across a set of short policy-related documents using pretrained transformer-based embeddings. Instead of matching exact words, we focus on the **meaning** of sentences using sentence embeddings and cosine similarity.

1️⃣ Load the Pretrained Sentence Embedding Model

```
from sentence_transformers import SentenceTransformer, util
model = SentenceTransformer('all-MiniLM-L6-v2')
```

✅ *This model converts sentences into 384-dimensional vector representations that capture context and meaning.*

2️⃣ Define Policy Documents and a Query

```
documents = [
    "The government reduced tariffs on agricultural imports in 1998.",
    "A major reform in 2002 led to an increase in VAT rates.",
    "In 2010, a new fiscal policy focused on education spending was implemented.",
    "Tariffs on electronics were removed under the 2015 free trade agreement.",
    "Corporate taxes were adjusted in response to the financial crisis of 2008.",
    "Foreign investment incentives were introduced in the energy sector.",
    "A subsidy program was initiated to support local manufacturers.",
    "The budget deficit led to a restructuring of public debt obligations.",
    "Import duties on technology were lifted to promote innovation.",
    "Tax cuts were proposed to stimulate consumer spending."
]

query = "tax reform during financial crisis"
```

3️⃣ Convert to Embeddings & Compute Cosine Similarity

```python
doc_embeddings = model.encode(documents, convert_to_tensor=True)
query_embedding = model.encode(query, convert_to_tensor=True)

cosine_scores = util.cos_sim(query_embedding, doc_embeddings)
top_result = cosine_scores.argmax().item()

print(f"Query: {query}")
print(f"Top Match: {documents[top_result]}")
print(f"Score: {cosine_scores[0][top_result]:.4f}")
```

🧾 **Simulated Output:**

```
Query: tax reform during financial crisis
Top Match: Corporate taxes were adjusted in response to the financial
crisis of 2008.
Score: 0.8124
```

🎯 *This means the query is semantically most similar to the fifth document.*

4️⃣ Find Top 3 Most Similar Documents

```python
import torch

scores = cosine_scores[0]
top_results = torch.topk(scores, k=3)

for score, idx in zip(top_results.values, top_results.indices):
    print(f"Document: {documents[idx]}")
    print(f"Similarity Score: {score:.4f}")
    print("---")
```

🧾 **Simulated Output:**

```
Document: Corporate taxes were adjusted in response to the financial
crisis of 2008.
Similarity Score: 0.8124
---
Document: A major reform in 2002 led to an increase in VAT rates.
Similarity Score: 0.6421
---
Document: Tax cuts were proposed to stimulate consumer spending.
Similarity Score: 0.5889
---
```

5️⃣ Tabulate the Results

```python
import pandas as pd

top_docs = [documents[idx] for idx in top_results.indices]
top_scores = [score.item() for score in top_results.values]

results_df = pd.DataFrame({
    'Document': top_docs,
    'Similarity Score': top_scores
}).sort_values(by='Similarity Score', ascending=False)

results_df
```

🧾 **Simulated Table:**

Document	Similarity Score
Corporate taxes were adjusted in response to the financial crisis...	0.8124
A major reform in 2002 led to an increase in VAT rates.	0.6421
Tax cuts were proposed to stimulate consumer spending.	0.5889

🧠 What is Cosine Similarity?

Cosine similarity measures how aligned two vectors are. It's defined as:

[\cos(heta) = \frac{{A \cdot B}}{{|A| \times |B|}}]

- **1.0**: vectors point in the same direction (perfect match)
- **0.0**: vectors are orthogonal (no similarity)
- **-1.0**: vectors are opposite

In NLP, it allows us to compare how similar two sentence embeddings are — that is, how semantically close they are.

💡 How Transformers Understand Context

Unlike older models (TF-IDF, Word2Vec), transformers use self-attention to weigh how important each word is to others in a sentence.

📘 Example:

- “bank” in *“river bank”* vs *“open a bank account”*
  The word is the same, but the model understands its **meaning from the surrounding words**.

That’s why “tax reform” and “corporate taxes during a crisis” end up close in vector space — **not because of keywords, but because of shared contextual meaning**.

✅ Summary

- We used `SentenceTransformer` to embed documents and a query
- Calculated similarity using cosine distance
- Returned the top-k most relevant documents semantically

🧠 This pipeline is adaptable for legal document search, academic research retrieval, and policy intelligence tools.