Sentence Transformers

from sentence\_transformers import SentenceTransformer, util

# 1. Load a pre-trained SBERT bi-encoder model

# 'all-MiniLM-L6-v2' is a common and efficient choice

model = SentenceTransformer('all-MiniLM-L6-v2')

# 2. Define the sentences to encode

sentences = [

"The cat sat on the mat.",

"A feline rested on the floor covering.",

"The dog barked loudly.",

"What is the capital of France?"

]

# 3. Encode the sentences into embeddings

# The encode method converts each sentence into a vector

sentence\_embeddings = model.encode(sentences, convert\_to\_tensor=True)

# 4. Perform a semantic search (e.g., find the most similar sentence to a query)

query = "A cat is on a rug."

query\_embedding = model.encode(query, convert\_to\_tensor=True)

# Calculate cosine similarity between the query and all sentences

cosine\_scores = util.cos\_sim(query\_embedding, sentence\_embeddings)[0]

# Find the index of the most similar sentence

most\_similar\_index = cosine\_scores.argmax().item()

most\_similar\_sentence = sentences[most\_similar\_index]

similarity\_score = cosine\_scores[most\_similar\_index].item()

print(f"Query: '{query}'")

print(f"Most similar sentence: '{most\_similar\_sentence}' (Similarity: {similarity\_score:.4f})")

# You can also compare any two sentences directly

sentence1 = "I love eating pizza."

sentence2 = "Pizza is my favorite food."

embedding1 = model.encode(sentence1, convert\_to\_tensor=True)

embedding2 = model.encode(sentence2, convert\_to\_tensor=True)

similarity\_between\_sentences = util.cos\_sim(embedding1, embedding2).item()

print(f"Similarity between '{sentence1}' and '{sentence2}': {similarity\_between\_sentences:.4f}")