Here's the updated code:

Step 1: Install HNSW Library

If you haven't already installed `hnswlib`, you can do so with:

bash Copy

```bash
pip install hnswlib
```

Step 2: Build the HNSW Index

Replace the FAISS index with an HNSW index.

python Copy

```python
import hnswlib
import numpy as np

# Assuming passage_embeddings is a numpy array of shape (num_passages, embedding_dim)
num_passages, embedding_dim = passage_embeddings.shape

# Create an HNSW index
index = hnswlib.Index(space="cosine", dim=embedding_dim)  # Use "cosine" for cosine similarity

# Initialize the index
index.init_index(max_elements=num_passages, ef_construction=200, M=16)

# Add embeddings to the index
index.add_items(passage_embeddings)

# Set the number of nearest neighbors to retrieve
index.set_ef(50)  # ef should be >= k (number of nearest neighbors to retrieve)
```

Step 3: Query the HNSW Index

Update the retrieval part of the code to use HNSW.

python Copy

```python
def retrieve_passages(query, k=5):
    # Encode the query
    query_embedding = encoder.encode([query])[0]  # Get the first (and only) embedding

    # Retrieve the top-k passages
    nearest_neighbors, distances = index.knn_query(query_embedding, k=k)

    # Get the retrieved passages
    retrieved_passages = [passages[i] for i in nearest_neighbors[0]]
    return retrieved_passages
```

Step 4: Integrate with the RAG Generator

Now, integrate the HNSW retriever with the RAG generator.

python Copy

```python
from transformers import RagTokenizer, RagSequenceForGeneration

# Load the RAG tokenizer and generator
tokenizer = RagTokenizer.from_pretrained("facebook/rag-sequence-nq")
generator = RagSequenceForGeneration.from_pretrained("facebook/rag-sequence-nq")

# Define the RAG query function
def rag_query(query, k=5):
```

```
    # Retrieve relevant passages
    retrieved_passages = retrieve_passages(query, k=k)

    # Combine retrieved passages into a single context
    context = " ".join(retrieved_passages)

    # Generate an answer using the RAG model
    inputs = tokenizer(context, query, return_tensors="pt", max_length=512, truncation=True)
    generated_ids = generator.generate(inputs["input_ids"])
    generated_answer = tokenizer.batch_decode(generated_ids, skip_special_tokens=True)[0]

    return generated_answer, retrieved_passages
```

Step 5: Test the RAG Model

Test the updated RAG pipeline with a sample query.

python Copy

```python
query = "What is the capital of France?"
answer, retrieved_passages = rag_query(query)

print("Generated Answer:", answer)
print("Retrieved Passages:", retrieved_passages)
```

Step 6: Save and Load the HNSW Index

To avoid rebuilding the index every time, you can save and load it.

Save the HNSW Index

python Copy

```python
index.save_index("hnsw_index.bin")
```

Load the HNSW Index

python Copy

```python
index = hnswlib.Index(space="cosine", dim=embedding_dim)
index.load_index("hnsw_index.bin")
index.set_ef(50)  # Set ef after loading the index
```

Full Updated Code

Here's the complete updated code for your RAG pipeline using HNSW:

python Copy

```python
import hnswlib
import numpy as np
from sentence_transformers import SentenceTransformer
from transformers import RagTokenizer, RagSequenceForGeneration

# Step 1: Encode Passages
encoder = SentenceTransformer("all-MiniLM-L6-v2")
passage_embeddings = encoder.encode(passages)  # Assuming passages is a list of strings
num_passages, embedding_dim = passage_embeddings.shape

# Step 2: Build HNSW Index
index = hnswlib.Index(space="cosine", dim=embedding_dim)
index.init_index(max_elements=num_passages, ef_construction=200, M=16)
index.add_items(passage_embeddings)
index.set_ef(50)
```

```python
# Step 3: Define Retriever
def retrieve_passages(query, k=5):
    query_embedding = encoder.encode([query])[0]
    nearest_neighbors, distances = index.knn_query(query_embedding, k=k)
    return [passages[i] for i in nearest_neighbors[0]]

# Step 4: Load RAG Generator
tokenizer = RagTokenizer.from_pretrained("facebook/rag-sequence-nq")
generator = RagSequenceForGeneration.from_pretrained("facebook/rag-sequence-nq")

# Step 5: Define RAG Query Function
def rag_query(query, k=5):
    retrieved_passages = retrieve_passages(query, k=k)
    context = " ".join(retrieved_passages)
    inputs = tokenizer(context, query, return_tensors="pt", max_length=512, truncation=True)
    generated_ids = generator.generate(inputs["input_ids"])
    generated_answer = tokenizer.batch_decode(generated_ids, skip_special_tokens=True)[0]
    return generated_answer, retrieved_passages

# Step 6: Test the RAG Model
query = "What is the capital of France?"
answer, retrieved_passages = rag_query(query)
print("Generated Answer:", answer)
print("Retrieved Passages:", retrieved_passages)
```

Key Points

HNSW is used for efficient retrieval of passages.

The RAG generator uses the retrieved passages to generate answers.

You can save and load the HNSW index for faster startup times.