Refine the database:

- To support efficient searching and retrieval with question, keywords, answers to store QnA pairs in a normalized and well-structured way such as tables.
- Properly index and format the data.
- Add categories, keyword indexing.
- Pre-process questions to normalize them for improved query matching.

Refine the code:

- Implement semantic search (RAG with domain-specific embeddings) rather than simple keyword matching.
- Improve error handling to catch missing entries, undefined tokens, or query mismatches.

**Rephrase the question to be more explicit in context.**
**Ensure the output format is consistent:**

- **Model changed from phi-1.5 to gpt2**
- Simplify the wording of the database
- explicitly tell the tokenizer **how to truncate** long sequences.
- It's **defaulting to 'longest_first' truncation strategy**, which may not always be optimal.

**Back to using the old model but with a refined database.**

- **Max database size:** Kaggle limits are 20gb disk space, 16gb ram per session

```
# Monitoring snippet to add to your code
def print_performance_stats(self):
    cursor = self.conn.execute("""
        SELECT
            COUNT(*) as total_entries,
            SUM(LENGTH(question)) / 1024 / 1024 as question_mb,
            SUM(LENGTH(answer)) / 1024 / 1024 as answer_mb,
            SUM(LENGTH(vector)) / 1024 / 1024 as embeddings_mb
        FROM qna_pairs
        JOIN embeddings ON qna_pairs.embedding_id = embeddings.embedding_id
    """)
    stats = cursor.fetchone()

    logger.info(f"""
    Database Performance Metrics:
    - Total entries: {stats[0]:,}
    - Question storage: {stats[1]:.2f}MB
    - Answer storage: {stats[2]:.2f}MB
    - Embeddings storage: {stats[3]:.2f}MB
    - Memory usage: {psutil.Process().memory_info().rss / 1024 /
1024:.2f}MB
    """)
```

gzip qna_data.jsonl
import gzip

```
import json

with gzip.open("qna_data.jsonl.gz", "rt", encoding="utf-8") as f:
    for line in f:
        qna = json.loads(line.strip())
        print(qna)

import faiss
import numpy as np

dim = 512  # Assuming embedding size is 512
index = faiss.IndexFlatL2(dim)  # L2 distance-based index

embeddings = np.random.rand(1000, dim).astype('float32')  # Example embeddings
index.add(embeddings)  # Add vectors to FAISS index

faiss.write_index(index, "/kaggle/working/qna_faiss.index")  # Save index
```