How to optimize database workflow:

- Slow embedding problem: Use GPU-accelerated embeddings
- Exact FAISS search: Use approximate nearest neighbors
- Large file i/o: Store embeddings in binary
- Use parallelization processing. Use multiprocessing for tokenization and other cpu-bound tasks.
- Reduce data redundancy. Filter low-relevance texts before embedding.
- Use cache intermediate steps by saving embedding s and FAISS indexes once then reload.

How to optimize keyword searches and semantic matches:

In the next morning, test that the original model could process and answer the questions in the most recent, large database smoothly.

Evaluate domain coverage, quality & diversity, and size requirements:

- Do we have enough data for fine-tuning?

1. Check Domain Relevance
    a. Compare the key terms extracted from my target domain against the model's tokenizer vocabulary.
        i. If >20% key terms are missing, extend tokenizer vocabulary via custom token embeddings, or use a domain-specific pre-trained BERT model. Recommended that >80% of domain terms are in vocab.
2. Evaluate Data Quality & Diversity
    a. Perplexity Check.
        i. High perplexity (>50) means the model struggles with this domain. Recommended to be <30.
3. Assess Training Data Size
    a. Rule of Thumb for Fine-Tuning
        i. Min of 1K Q&A pairs, recommended data size is about 50K+. Training data should cover major edge cases and diverse examples.
    b. If the data is insufficient,
        i. Use a few-shot learning with LangChain's prompts
        ii. Augment data via synthetic generation.

The Dataset should be more streamlined, processed and scraped from papers or verified/approved web pages in the future.