code.py 的说明

Code Dependencies

```
langchain==0.0.340
faiss-cpu==1.7.4
openai==1.3.6
sentence-transformers==2.2.2
```

Code Structure

1.向量数据库 Setup (`get_db()`)

```
def get_db(data_path, db_path, flag):
    """
    由 JSON data 制作 FAISS 向量数据库

    参数:
        data_path: JSON data file
        db_path: 向量数据的目录位置
        flag: 使用 'articles' or 'charges' 来表明数据

    Returns:
        生成 FAISS 向量数据库
    """
```

2. Load 已有的数据库 (`load_vector_db()`)

```
def load_vector_db(db_path, embedding_name="bge"):
    """
    Loads 已有的 FAISS 向量数据库

    Args:
        db_path: 向量数据的目录位置
        embedding_name: 使用bge模型

    Returns:
      返回选定的向量数据库
    """
```

3. Charge Prediction (`predict_charges()`)

```
def predict_charges(fact, defendants, top_k=3):
    """
    基于 qwen-max 根据 prompt 进行预测并记录结果

    Args:
        fact: 案件事实
        defendants: 被告人的名字列表
        top_k: rag检索top k=3个

    Returns:
        返回 JSON 的结果
    """
```

4. 主执行流程

```
if __name__ == "__main__":
    # 1. 加载训练数据
    # 2. 处理每个案件
    # 3. 生成预测结果
    # 4. 结果保存在jsonl
```

核心组件

1. **嵌入模型**：使用BAAI/bge-large-zh-v1.5中文文本嵌入
2. **向量存储**：FAISS用于高效相似性搜索
3. **大语言模型集成**：通过OpenAI兼容API调用Qwen-Max
4. **数据流程**：
    - 加载法律条文和罪名数据
    - 创建向量表示
    - 检索相关法律上下文
    - 生成罪名预测

输入/输出格式

输入：

```
{"fact": "案件描述", "defendants": ["被告人1", "被告人2"]}
```

输出：

```
{"被告人": ["唐某B", "李某某"],"罪行": ["故意毁坏财物罪", "故意毁坏财物罪"],"刑期(月)": [6, 6]}
```