数据加载，分割与向量化入库

1.数据load

根据实验文档说明，使用 `langchain.document_loaders` 中的 `CSVLoader` 来加载的数据

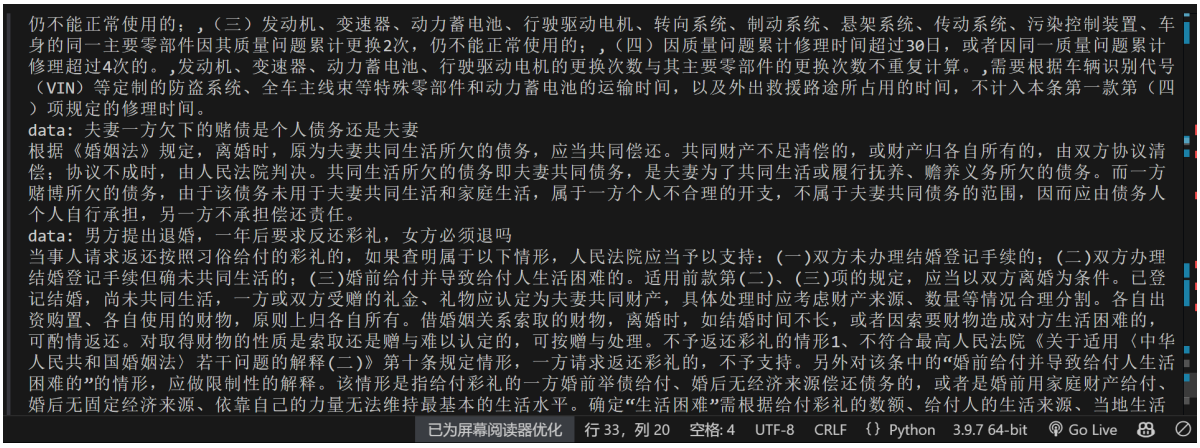
```python
from langchain_community.document_loaders import CSVLoader

# Load CSV data
file_path = "law_data_3k.csv"
loader = CSVLoader(file_path=file_path)
data = loader.load()
```

load之后观察data的数据情况

```
Total number of records: 3013
Processed 1000 records.
Processed 2000 records.
Processed 3000 records.
```

共计3013组数据，而观察law_data_3k.csv实际共有近6000行，这是因为load操作把2400余条问答的问题和回答两项记在一起了，：



如上图，一个data块里包括了问题和解答，而在csv里他们属于不同的行。

2.数据分割

数据分割参考 `langchain.text_splitter` 中的 `CharacterTextSplitter`,其中分割参数有下面这些选项:

```python
# Initialize text splitter
text_splitter = CharacterTextSplitter(
    separator="\n\n",        # Use paragraph separator
    chunk_size=1000,         # Set chunk size
    chunk_overlap=100,      # Set overlap between chunks to 100 characters
    length_function=len,    # Use len to calculate text length
    is_separator_regex=False,
)
```

测试发现当分割符选择 `\n\n` 时，数据块分割后数量不变，说明文本里并没有 `\n\n`,这个不能起到好的分割作用，而选择 `\n`,的情况下块的数量会有所增长。

关于块大小的问题，选择不同 `chunk size` 的块进行测试，所得分块数量如下:

> chunk = 1000

```
Total number of records: 3013
Processed 1000 records.
Processed 2000 records.
Processed 3000 records.
Total number of chunks: 3261
```

> chunk = 500

```
Total number of records: 3013
Processed 1000 records.
Processed 2000 records.
Processed 3000 records.
Total number of chunks: 4254
```

> chunk = 300

```
Total number of records: 3013
Processed 1000 records.
Processed 2000 records.
Processed 3000 records.
Total number of chunks: 4254
```

> chunk = 200

```
Total number of records: 3013
Processed 1000 records.
Processed 2000 records.
Processed 3000 records.
Total number of chunks: 4921
```

> chunk = 150

```
Total number of records: 3013
Processed 1000 records.
Processed 2000 records.
Processed 3000 records.
Total number of chunks: 5092
```

分块100时会有提示分块太小，根据`csv`数据分布，个人觉得选则`200`块大小比较合适.

3. 向量化处理与入库

选取实验文档里的`m3e-base`嵌入模型构建向量，把这个模型部署在本地后封装在`MyEmbeddingFunction`里作为参数传给`chroma`，`chroma`将`document`里的数据向量化后存放在建立的数据库里：

```
class MyEmbeddingFunction(EmbeddingFunction):
     def embed_documents(self, texts: Documents) -> Embeddings:
        # 使用 tqdm 显示嵌入进度条
        embeddings = [
            model.encode(text, convert_to_numpy=True) for text in
tqdm(texts, desc="Embedding Texts", unit="text")
        ]
        # 返回嵌入结果
        return [embedding.tolist() for embedding in embeddings]

# Load pre-trained SentenceTransformer model
model = SentenceTransformer('D:/2024 fall/web/lab/实验三/m3e-base')

```

```
try:
    # Chroma expects an embedding function, passing MyEmbeddingFunction as
the embedding function
    persist_directory = "./chroma_data_200"
    db = Chroma.from_documents(
        documents,
        MyEmbeddingFunction(),
        persist_directory=persist_directory
    )
    print("Chroma database saved")
except Exception as e:
    print(f"Error saving Chroma database: {e}")

```

最后运行结果

chunk= 200 的运行结果以及源代码在这里。(附上chunk =300 的结果，以便后续选择)

链接：https://rec.ustc.edu.cn/share/e1899490-d0ce-11ef-ab47-e9e1edc2baf9

密码：1958