7.用到的工具

1.FastAPI

FastAPI 是一个用于构建 API 的现代、快速（高性能）的 web 框架，使用 Python 3.8+ 并基于标准的 Python 类型提示。

关键特性：

- **快速**：可与 **NodeJS** 和 **Go** 并肩的极高性能（归功于 Starlette 和 Pydantic）。最快的 Python web 框架之一。
- **高效编码**：提高功能开发速度约 200% 至 300%。*
- **更少 bug**：减少约 40% 的人为（开发者）导致错误。*
- **智能**：极佳的编辑器支持。处处皆可自动补全，减少调试时间。
- **简单**：设计的易于使用和学习，阅读文档的时间更短。
- **简短**：使代码重复最小化。通过不同的参数声明实现丰富功能。bug 更少。
- **健壮**：生产可用级别的代码。还有自动生成的交互式文档。
- **标准化**：基于（并完全兼容）API 的相关开放标准：OpenAPI (以前被称为 Swagger) 和 JSON Schema。

Python

```python
# 处理POST请求的端点
@app.post("/")
async def create_item(request: Request):
    global model, tokenizer  # 声明全局变量以便在函数内部使用模型和分词器
    json_post_raw = await request.json()  # 获取POST请求的JSON数据
    json_post = json.dumps(json_post_raw)  # 将JSON数据转换为字符串
    json_post_list = json.loads(json_post)  # 将字符串转换为Python对象
    prompt = json_post_list.get('prompt')  # 获取请求中的提示
    history = json_post_list.get('history')  # 获取请求中的历史记录
    max_length = json_post_list.get('max_length')  # 获取请求中的最大长度
    top_p = json_post_list.get('top_p')  # 获取请求中的top_p参数
```

```
    temperature = json_post_list.get('temperature')  # 获取请求中的温度参数
    # 调用模型进行对话生成
    response, history = model.chat(
        tokenizer,
        prompt,
        history=history,
        max_length=max_length if max_length else 2048,  # 如果未提供最大长度，默认使用2048
        top_p=top_p if top_p else 0.7,  # 如果未提供top_p参数，默认使用0.7
        temperature=temperature if temperature else 0.95  # 如果未提供温度参数，默认使用0.95
    )
    now = datetime.datetime.now()  # 获取当前时间
    time = now.strftime("%Y-%m-%d %H:%M:%S")  # 格式化时间为字符串
    # 构建响应JSON
    answer = {
        "result": response,
        "history": history,
        "status_code": 200,
        "time": time
    }
    # 构建日志信息
    log = "[" + time + "] " + '", prompt:"' + prompt + '", respons
e:"' + repr(response) + '"'
    print(log)  # 打印日志
    torch_gc()  # 执行GPU内存清理
    return answer  # 返回响应
```

2.Kor

`Kor` 是个支持 `LangChain` 的文本抽取库，可以把文本抽取成 `json` 格式。简单使用一下 `Kor`，首先用 `langchain` 的 `LLM` 模块重新封装一下，`langchian` 中的 `ChatOpenAI` 类。

Python

```
from kor.extraction import create_extraction_chain
from kor.nodes import Object, Text, Number
from langchain.chat_models import ChatOpenAI
```

2.1 Schema

Kor 要求您使用一些可选示例指定要解析的内容的架构。

Python

```
# kor示例
schema = Object(
    id="script",
    description="Adapted from the novel into script",
    attributes=[
        Text(
            id="role",
            description="The character who is speaking",
        ),
        Text(
            id="dialogue",
            description="The dialogue spoken by the characters in th
e sentence",
        )
    ],
    examples=[
        (
            '''
            龙王说:“再也没有比这更重的兵器了。”悟空不信，和龙王吵了起来，龙婆给龙王说:“大禹治水时，测定海水深浅的神珍铁最近总是放光，就把这给他，管他能不能用，打发他走算了。”龙王听后告诉悟空:“这宝物太重了，你自己去取吧！”
            ''',
            [
                {"role": "龙王", "dialogue": "再也没有比这更重的兵器
了。"},
                {"role": "龙婆", "dialogue": "大禹治水时，测定海水深浅的神珍铁最近总是放光，就把这给他，管他能不能用，打发他走算了。”龙王听后告诉悟空:“这宝物太重了，你自己去取吧！"},
            ],
        ),
        (
            '''
```

```
            悟空见八戒这么长时间不回来，就拔根毫毛变成自己，陪着师父和沙僧，真身驾云来到山凹里，见八戒和妖精正在交战，便高声叫道:“八戒别慌，老孙来了！”八戒一听，来了精神，没几下，就把那群妖怪打败了。
            ''',
            [
                {"role": "悟空", "dialogue": "八戒别慌，老孙来了！"},
            ],
        )
    ],
    many=True,
)
```

2.2 Langchain

```python
from langchain.llms import OpenAI

llm = ChatOpenAI(
    model_name="gpt-3.5-turbo",
    temperature=0,
    max_tokens=2000,
    frequency_penalty=0,
    presence_penalty=0,
    top_p=1.0,
)

chain = create_extraction_chain(llm, schema)
```

2.3 Extract

定义了链和模式后，我们就可以提取数据了。

```python
run(chain, chunk_list[i])
```