法狗狗 LegalChat 对接文档

更新日期：2023.5.16

介绍

LegalChat 是法狗狗基于大语言模型技术研发的可行法律咨询机器人，目前已经导入常用法律法规、律师建议和判决案例。该咨询机器人具有理解用户意图、主动引导用户补充信息、解答用户法律问题等功能。接口使用 Websocket 协议传输，机器人回答问题时，会逐字流式返回咨询结果。由于该服务处于开发测试阶段，该接口的数据结构和用法可能随时变更，请您谅解。示例代码为 TypeScript+Vue3 代码，其他语言可以参考该结构。不是完整代码，无法直接运行。仅供您参考数据结构和对接方式。

建议采用的开发框架：

Web：TypeScript + Vue3 + Vite

安卓：Kotlin + Jetpack Compose 或其他有双向数据绑定功能的框架

IOS：Swift UI

桌面：Electron 等跨平台框架+Web 技术栈

注意事项

1. 一个问答一个 socket， 也就是说每次发起问题都需要新建一个 socket， 因为每次 socket 回答完成，后端都会把 socket 关闭
2. 一个话题有字数限制，如果超过了后端会发送 type 为 close 的事件，也就是话题结束事件，需要用户重新咨询

接口基本信息

token ：

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Ik9aSDJFNjlrZUROYzluS
EdfemRwaGxqSjJ1b2tHaW5SRml0ajdxTldOS3cifQ.eyJzdWIiOiI2NDQwOWJhYzl
hNjhhOWFkZTM0ODcxZWEiLCJhdWQiOiI2NDQwOTM0MzI0ZTAyNGEwZjI3YjFmZjUi
LCJzY29wZSI6InByb2ZpbGUgcGhvbmUgb2ZmbGluZV9hY2Nlc3Mgb3BlbmlkIGVtY
WlsIiwiaWF0IjoxNjgyNjczMzc4LCJleHAiOjE2ODM4ODI5NzgsImp0aSI6ImhMZU
1kQmhjaC0ybWlZZVZYd042YXVodWd6N0VUWjFlaDZiT0gwRjliRU4iLCJpc3MiOiJ
odHRwczovL2xlZ2FsLWNoYXQuYXV0aGluZy5jbi9vaWRjIn0.c6ZqYw3u2W_mLbIU
FC9TUPCnlPQxZWDbIpsqSZ0P8dkmf0wBATvoSSGyoOTh0CbhyUzRJGdaZxLaVvCxc
Vw4wNBiIP7i3P0VOgvBIzd4TZPubGxUA61yLJxdH_i9Ouo163tUsI6gjNgnU1ijAQ
ULsjDeAF5sz0p6eZJDWiqk8g1NxtuCdY4cV3Ru8wsF3JjRgyogatLOf3Z6MghBZrC
vk7zTuvRiW991M8rK00n1anW8jyQkPRqgVLnY7L7bcln-iFP-
5gMxDJXK_gvu45_RzBssGeEhP1qbb7M7Rc6P61agjmu666nJFLAgkME1xmHrPxva4
LuoYGm1srhc06TJ9w
```


wsUrl:

```
wss://service-1dbgn4h6-
1254426977.http://hk.apigw.tencentcs.com/release/faxiaokai/
```

JavaScript 实现参考

工具辅助

全局变量

```
process_total=0 //总进度
process_current=0 //当前进度
chat_id= nanoid(10) //话题 id
historyMsg = [{
  role:'assistant',
  content:'您好！我是 xxxx，很高兴为您服务。'
}] // 历史记录 role 取值 assistant 或者 user 其中 user 是用户问题类型 另外一个是答案类型
stream=‘’ 流式字符显示
optionAnswerId= '' // 推荐问题 id
recommend_list=[] //推荐问题列表
loading=false //是否正在回复
```

生成 chat_id

```
//生成随机字符串
function nanoid(len) {
  len = len || 10 //默认长度为 10
  var str = ''
  var chars =
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789'
//字符集
  var charsLen = chars.length
  for (var i = 0; i < len; i++) {
    str += chars.charAt(Math.floor(Math.random() * charsLen)) //
随机选取一个字符
  }
  return str
}
```

初始化 websocket

```
// ws_url: socket 地址
// token: token 值用来鉴权
// chat_id: 话题 id
// msg: 用户问题
function initWebsokcet(chat_id,msg){
  const ws_url = 'xxxx'
  const token = 'xxxx'
  const ws = new WebSocket(ws_url)
  ws.onopen = () => {
    ws.send(token)
    ws.send(chat_id)
    ws.send(msg)
  }
  ws.onmessage = (data: any) => {
    let event: any = JSON.parse(data.data)
    add_message(event)
  }
  ws.onclose = ()=>{
    loading = false
  }
  ws.onerror = (err) => {
    loading = false
    console.log(err)
    console.error('服务器连接失败')
  }
}
```

核心业务流程

用户发送问题

```
function sendQes (msg){
  if(loading) return
  loading = true
  recommend_list =[] //清空推荐问题
  optionAnswerId=nanoid(10)
  //添加历史记录
  historyMsg.push({
    role:'user',
    content:'msg',
  })
  //每次发送消息都需要重新建议 socket
  initWebsokcet（chat_id,msg）
}
```

用户发送推荐问题

```
function sendrecommendQes (msg){
  optionAnswerId=nanoid(10)
  recommend_list =[] //清空推荐问题
  if（msg==='重新咨询）{
  chatId =nanoid(10) //开启新话题
  stream=''
  historyMsg = [
    {
      role: 'assistant',
      content: '您好！我是 xxx，很高兴为您服务。',
    }]
  return
}
if(loading) return
loading = true

//添加历史记录
historyMsg.push({
  role:'user',
  content:'msg',
})
//每次发送消息都需要重新建立 socket
initWebsokcet（chat_id,msg）
}
```

接收答案并添加到前端聊天记录中

```
function add_message(answer){
  switch (answer.type) {
    case 'close' :
      recommend_list = ['重新咨询']
      loading = false
    case 'char':
      process_current = 0
      process_total = 0
      const content = answer.content //回答内容
      if((content !== '[换行]' && content !== '[结束]') return
stream+=content
      if(content === '[换行]')
        return historyMsg.push({role: 'assistant',content:
stream}
                                //结束
                                loading = false
      historyMsg.push({role: 'assistant', content: stream})
      const current_stage_id = optionAnswerId = nanoid(10) //当前
推荐问题 id



      //获取推荐问题接口
      const last_message ={ role: 'assistant', content: stream }
      const BASE_URL_HK= 'https://service-1dbgn4h6-
1254426977.hk.apigw.tencentcs.com/release/predict'
      stream = ''
      fetch(BASE_URL_HK, {
        method: 'POST',
        body: JSON.stringify({ history: last_message }),
      })
        .then((res) => {
          return res.json()
        })
        .then((data: any) => {
          //如果不是当前推荐问题直接抛弃
          if (optionAnswerId == current_stage_id)
recommend_list.value = data['data']
        })
      break;
    case 'process': //分析进度
```

已过期的说明

建议

建议采用的数据结构和变量：

```
class MessageItem {
   role: string
   content: string
   (role:string,content:string){
      this.role = role
      this.content = ""
   }
}

const history: Ref<Array<MessageItem>> = ref([]) // 聊天记录
const input = ref("") // 输入框
const stream = ref("") // 流式回复
const process_current = ref(0) // 法典阅读进度
const process_total = ref(0) // 法典长度
const recommend_list: Ref<Array<string>> = ref([]) // 推荐回复列表
const is_closed = ref(false) // 长链接是否关闭
const loading = ref(true) // 是否加载中
var stage_id = "" // 当前对话轮次 id,用于丢弃过期请求
```

接口说明（已过期，部分可参考）

WebSocket 接口

请求地址： wss://service-1dbgn4h6-1254426977.hk.apigw.tencentcs.com/release/faxiaokai/

鉴权方式： 建立 WebSocket 连接时，发送 Access Token 作为首条消息。

调试试用 Token：

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Ik9aSDJFNjlrZUROYzluSEdfemRwaGxqSjJ1b2tHaW5SRml0ajdxTldOS3cifQ.eyJzdWIiOiI2NDQwOWJhYzlhNjhhOWFkZTM0ODcxZWEiLCJhdWQiOiI2NDQwOTM0MzI0ZTAyNGEwZjI3YjFmZjUiLCJzY29wZSI6InByb2ZpbGUgcGhvbmUgb2ZmbGluZV9hY2Nlc3Mgb3BlbmlkIGVtYWlsIiwiaWF0IjoxNjgyNjczMzc4LCJleHAiOjE2ODM4ODI5NzgsImp0aSI6ImhMZU1kQmhjaC0ybWlZZVZYd042YXVodWd6N0VUWjFlaDZiT0gwRjliRU4iLCJpc3MiOiJodHRwczovL2xlZ2FsLWNoYXQuYXV0aGluZy5jbi9vaWRjIn0.c6ZqYw3u2W_mLbIUFC9TUPCnlPQxZWDbIpsqSZ0P8dkmf0wBATvoSSGyoOTh0CbhyUzRJGdaZxLaVvCxcVw4wNBiIP7i3P0VOgvBIzd4TZPubGxUA61yLJxdH_i9Ouo163tUsI6gjNgnU1ijAQULsjDeAF5sz0p6eZJDWiqk8g1NxtuCdY4cV3Ru8wsF3JjRgyogatLOf3Z6MghBZrCvk7zTuvRiW991M8rK00n1anW8jyQkPRqgVLnY7L7bcln-iFP-5gMxDJXK_gvu45_RzBssGeEhP1qbb7M7Rc6P61agjmu666nJFLAgkME1xmHrPxva4LuoYGm1srhc06TJ9w

鉴权示例代码：

```
const ws = new WebSocket("wss://service-1dbgn4h6-1254426977.hk.apigw.tencentcs.com/release/")
ws.onopen = () => {
```

```
  ws.send(`${上文中给出的试用 token}`)
}
```

传入数据：直接向 WebSocket 发送用户问题

示例代码(TypeScript + Vue3)：

```
ws.send(input.value)
```

接收数据示例：

1、字符

正常字符：{"type": "char", "content": "\u60a8"}

结束标识：{"type": "char", "content": "[\u7ed3\u675f]"}

收到字符后，应将收到的内容加入 stream 向用户展示，直到收到 [结束] 标识时将 stream 放到 history 中。

2、进度条

{"type": "process", "current": 1, "total": 8}

示例代码(TypeScript + Vue3)：

```
ws.onmessage = (data: any) => {
  let event: any = JSON.parse(data.data)
  switch (event.type) {
    case "char":
      process_total.value = 0
      if (event.content == '[结束]') {
        stage_id = nanoid(10)
        let predict_history = []
        if (history.value.length > 1) {
          let last_ask = ""
          for (let item of history.value[history.value.length - 1].content.keys()) last_ask = item
          predict_history.push({"role": "user", "content": last_ask})
          predict_history.push({"role": "assistant", "content": stream.value})
          predictQuestion(predict_history, stage_id) // 该方法实现见下一个接口
        }
        loading.value = false
                history.value = history.value.concat([new MessageItem("robot", stream.value)])
      } else stream.value += event.content
      break
    case "process":
      process_current.value = event.current
      process_total.value = event.total
      break
  }
```

```
}
```

Http 接口

请求地址：http://service-1dbgn4h6-1254426977.hk.apigw.tencentcs.com/release/predict

传入数据：json，传入参数为最后一条回复。形式如 {"history": {"role": "assistant", "content": "XXX"}}

示例代码(TypeScript + Vue3)：

```
function predictQuestion(last_message: any, current_stage_id: string) {
    if (is_closed.value) return
    fetch(`${BASE_URL_HK}/predict`, {
        method: "POST",
        body: JSON.stringify({"history": last_message})
    }).then((res) => {
        return res.json()
    }).then((data: any) => {
        if (stage_id == current_stage_id) recommend_list.value = data['data']
    })
}
```