祉来好！

我初步看了《使用说明1.0》， 且试用了以下 app 。

确实，登录等待时间是比较长的。（对于产品而言，这个等待时间是不太OK的。）我看了一下 server.py 的实现， 认为，可以改进一下总体程序的设计。

1. 主线程，一开始，就开足够的 10个用户对话线程(UserDialogThread) 。
2. UserDialogThread 线程，维护一个状态变量 State , State的取值可以是（stInit, stWaitConnect, stConnect，stLogin, stChat, stError）枚举类型的值。 初始值是 stInit .

运行的伪代码如下：

while True ：

if(self.State==stInit) : #初始状态

#加载 seleninum 驱动，并定位到 my.replika.ai 网页，并可以把 login的连接先点击展开准备好。

driver = webdriver.Firefox()

driver.get('https://my.replika.ai/')

……

self.State=stWaitConnect

elif(self.State==stWaitConnect) :

pass

elif (self.State==stConnect) : # 和App 已建立tcp网络连接状态

#socket 接收 app 端发送过来的 登录 用户名和密码， 如接收这些信息成功，self.State=stLogin

elif (self.State==stLogin) : #登录登录状态

#模拟网页进行登录，如登录成功，self.State=stChat , 否则，休眠sleep 几秒，（因为 状态维持在 stLogin ， 所以，休眠结束后，仍会重做 登录这个网页操作的。）

elif (self.State==stChat) : #进入对话状态

# socket 接收 app 端发送过来的聊天信息 ，汉译英，发送给 replika 网页后台， 接收replika 发送过来的聊天信息，英译汉，发送给app 端。

else (self.State==stError) : #进入错误处理状态

#如socket错误的，关闭和app的网络连接，self.State=stInit 状态， 等等。

1. UserDialigThread 类可以设计有一个 SetConnect(self, socket，ipAddr) 函数

def SetConnect(self, socket,ipAddr) :

if(self.State==stWaitConnect) :

self.socket=socket

self.ipAddr=ipAddr

self.State=stConnect

当主线程接受了客户端App的网络连接时，可以查看一下 哪个UserDialogThread 是在 stInit 状态的， 然后调用该 thread 的 SetConnect（） 函数。

最主要的，是请使用 状态驱动的方式 。

1. 参考：https://www.cnblogs.com/101718qiong/p/7390718.html

可以使用 implicitly\_wait() 方法就可以方便的实现智能等待 。

1. 对话聊天信息，需要写入聊天日志文件， 日志文件名，为“chat\_用户名.log” ， 记录聊天信息的发送和接收时间戳（timestamp），信息文本内容，包括翻译前后的中文信息和英文信息。
2. replika 的聊天信息 有带 \* \* 的， \* \*之间的文字表示的是一个 动作， 也就是 我们想象 replika 在做这个动作， 类似我们平时文字聊天的 “颜文字” 表示一些情感表情 一样。