# 大作业：基于文心系列模型的地下城冒险游戏聊天机器人

# 大作业项目介绍

1.1项目背景&模型介绍

ERNIE-4.0-8K是百度自研的旗舰级超大规模⼤语⾔模型，相较ERNIE 3.5实现了模型能力全面升级，广泛适用于各领域复杂任务场景。在数据分析场景，ERNIE-4.0-8K（或称为ERNIE-Bot-8K）表现出优异的性能。根据Kyligence的评测结果，该模型在指标计算、分析洞察等方面均有出色表现，是目前评测过最接近OpenAI GPT的国产大模型。

1.2项目内容

本项目即采用该模型，通过API的调用实现本地的窗口对话，在本地存储对话历史信息以及游戏基本设定，并设计可视化界面。

1.3游戏内容

一个地下城文字冒险游戏，玩家共有四个职业可以选择，并有生命值、法力值、精神力量等客观属性，共设有战斗、事件、宝箱三种类型房间。用户需与管理员通过对话控制玩家进行杀敌和闯关，通过三层地下城，并击败最终Boss。

# 大作业程序内容

### API调用

1. # 初始化认证信息
2. os.environ["QIANFAN\_ACCESS\_KEY"] = "ALTAKGlLAqn33zQlysLMTsgf5J"
3. os.environ["QIANFAN\_SECRET\_KEY"] = "e07e00771c6d46088b3b87c9061d0349"
4. chat\_comp = qianfan.ChatCompletion()

API 调用模块负责与Qianfan的ChatCompletion模型进行交互。程序初始化时，首先设置了Qianfan API的认证信息，通过os.environ来设定访问密钥和秘密密钥；

1. resp = chat\_comp.do(model="ERNIE-4.0-8K", messages=messages)
2. assistant\_reply = resp["body"]["result"]

随后，使用qianfan.ChatCompletion()初始化聊天组件chat\_comp。在获取到用户的输入后，程序会将其添加到对话历史messages中，并通过调用chat\_comp.do与Qianfan的生成模型ERNIE-4.0-8K进行交互：生成的结果将作为系统管理员的回复，并呈现在界面中。

### 可视化界面

可视化界面模块基于PyQt5框架开发，提供了一个简单但直观的用户界面。

#### 文本显示框

1. font = QFont()
2. font.setPointSize(14)
3. self.text\_display.setFont(font)
4. self.text\_display.setStyleSheet(
5. "color: white; "
6. "border-image: url('./background.jpg'); "
7. )

用于显示系统管理员与用户之间的对话，并设置为只读，防止用户修改历史记录。文本显示框的字体大小为14pt，并且设置了字体颜色为白色，同时应用了背景图片。

#### 输入框

1. self.input\_box.setFont(font)
2. self.input\_box.setStyleSheet("background-color: white; color: black;")

用户可以在输入框中键入文本进行互动。输入框的字体与文本显示框一致，并且设置了白色背景和黑色字体，便于用户阅读和输入。

#### 发送键

1. self.send\_button = QPushButton("发送")
2. self.send\_button.clicked.connect(self.send\_message)
3. layout.addWidget(self.send\_button)

发送键与send\_message相绑定，用于发送信息：

1. def send\_message(self):
2. *# 获取用户输入*
3. user\_input = self.input\_box.text().strip()
4. if user\_input == "":
5. return
6. *# 显示用户输入*
7. self.text\_display.append(f"你: {user\_input}\n")
8. self.input\_box.clear()
9. *# 提示助手思考中*
10. self.text\_display.append("助手: 正在思考...\n")
11. *# 禁用输入框*
12. self.input\_box.setDisabled(True)
13. *# 退出条件*
14. if user\_input.lower() in ["exit", "quit", "退出"]:
15. self.text\_display.append("对话结束。")
16. return
17. *# API 调用放到后台线程中，避免阻塞主线程*
18. QtCore.QTimer.singleShot(100, lambda: self.get\_response(user\_input))

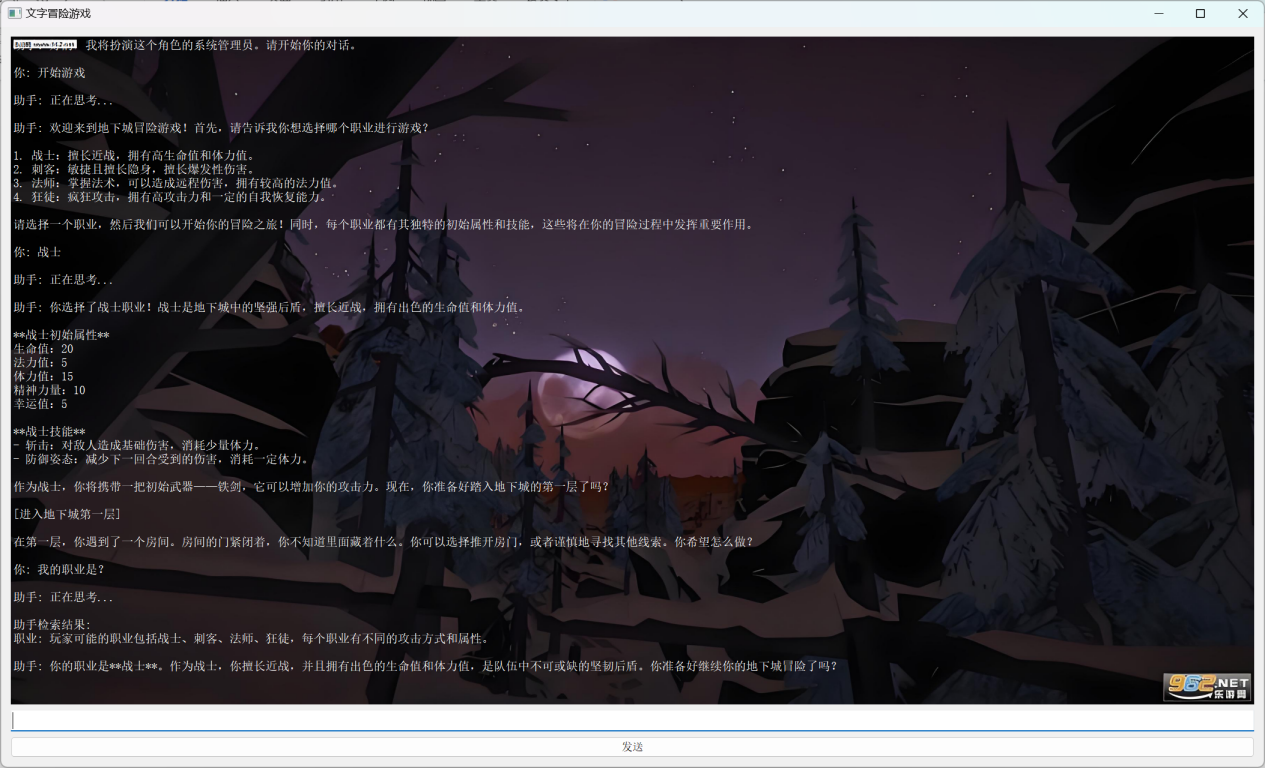
该函数获取用户输入，排除空白输入后进行显示，暂时禁用输入框并提示正在思考，并针对结束命令进行判断，若为exit等则直接退出；若为一般对话，则在后台运行API调用。

#### 检索增强与回复模块

1. def get\_response(self, user\_input):
2. *# 检索增强：先从知识库中检索相关信息*
3. relevant\_info = retrieve\_relevant\_info(user\_input)
4. if relevant\_info != "未找到相关信息。":
5. self.text\_display.append(f"助手检索结果:\n{relevant\_info}\n")
6. try:
7. *# 添加用户输入到对话历史*
8. messages.append({"role": "user", "content": user\_input})
9. *# 调用生成接口前，确保最后一个是用户的消息（偶数索引）*
10. resp = chat\_comp.do(model="ERNIE-4.0-8K", messages=messages)
11. assistant\_reply = resp["body"]["result"]
12. except Exception as e:
13. assistant\_reply = f"发生错误: {str(e)}"
14. *# 显示助手回复*
15. self.text\_display.append(f"助手: {assistant\_reply}\n")
16. *# 添加助手回复到对话历史*
17. messages.append({"role": "assistant", "content": assistant\_reply})
18. *# 启用输入框*
19. self.input\_box.setDisabled(False)
20. *# 滚动到最新消息*
21. self.text\_display.moveCursor(QtGui.QTextCursor.End)

该模块首先调用retrieve\_relevant\_info在预置的检索模块中检索用户查询的信息，如“战斗”等关键词的解释。然后在存储对话信息的message中加入用户输入，调用API中的接口模块获取回答信息，输出至显示框并添加到对话历史信息并解锁输入框。

# 效果展示



上图为部分对话。

# 参考文献

[1] https://cloud.baidu.com/doc/WENXINWORKSHOP/s/clntwmv7t