**任课教师：张涵翠**

《Python程序设计》

**(2024-2025第1学期)**

**大作业实验报告**

**题目名（题号）：AI飞花令（13）**

**姓名（学号）：**

**班级：*计算机科学与技术23(班***

# 摘要

飞花令作为中国古代文人雅集的文字游戏，蕴含深厚的语言文化与诗词审美价值。为实现传统文化的数字化传承，本课题设计并开发了“AI飞花令”智能交互系统，通过程序化方式模拟古人行酒令的规则与互动逻辑，探索人工智能技术在诗词文化传播中的创新应用。项目以Python语言为核心工具，结合大语言模型与向量检索技术，构建了支持自主出题、规则判断、人机对战等功能的智能化平台。此外，为丰富用户体验并探索多样化的交互模式，小组还自主设计并实现了一款基于Flet框架的轻量级AI飞花令人机交互GUI，可作为核心系统的有益补充，旨在降低诗词学习门槛，提升文化互动体验。

系统采用模块化架构，后端基于AgentScope框架管理主持人、评审官、AI对手三类智能体角色，通过调用阿里云通义千问API实现诗句生成与合规性判断；前端依托Gradio搭建双栏交互界面，集成关键词输入、实时反馈与历史记录功能。Flet版GUI则提供了另一种简洁的单窗口对话式对战体验。数据管理层面，利用FAISS向量库与Sentence Transformers模型构建诗词语义索引，实现关键词驱动的高效检索。实验验证表明，系统能稳定运行单人游戏模式，评审模块对关键字识别准确率达85%以上，AI对手生成的诗句中90%为真实古诗。

本研究不仅实现了传统文字游戏的数字化转型，更为诗词教育提供了趣味化、即时化的技术路径。系统可扩展为课堂教学工具、文化类App核心模块或博物馆互动装置，具有显著的教育价值与文化传播意义。后续将通过动态数据爬取、轻量化模型部署及多人联机功能开发，进一步提升系统的实时性与文化适配性。

关键词：AI飞花令；向量检索；智能交互；

；

# 课题执行（文档修订）记录表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 日期 | 版本 | 修订内容 | 执行人 | 修订人 |
| *2025.4.10* | *1.0* | *完成系统框架选型，采用C/S架构* | *黄红洲* | *张志远* |
| *2025.4.13* | *1.1* | *完成系统开放框架选型* | *黄红洲* | *黄红洲* |
| *2025.4.21* | *1.2* | *修订诗句生成功能为生成古代已有的诗句* | *张倍源* | *张倍源* |
| *2025.5.7* | *1.3* | *修订flyflower\_gradio.py可视化文件* | *黄红洲* | *孙吴逸朴* |
| *2025.5.9* | *1.4* | *完成数据分析代码实现和报告撰写* | *张倍源* | *张倍源* |
| *2025.5.10* | *15* | *完成报告摘要总结并整合* | *张志远* | *孙吴逸朴* |

目录

[1．引言 5](#_Toc198046987)

[1.1．课题名称及其描述 5](#_Toc198046988)

[1.2. 课题目的及意义 5](#_Toc198046989)

[1.3．课题应用场景及特点 5](#_Toc198046990)

[2．系统设计 6](#_Toc198046991)

[2.1. 概要设计 6](#_Toc198046992)

[2.1.1 系统总体结构设计（技术框架、系统框架） 6](#_Toc198046993)

[2.1.2 系统模块及功能设计 7](#_Toc198046994)

[2.1.3 系统性能需求设计 7](#_Toc198046995)

[2.2. 功能设计 7](#_Toc198046996)

[2.2.1 游戏初始化 7](#_Toc198046997)

[2.2.2 关键字生成 7](#_Toc198046998)

[2.2.3 用户答题 7](#_Toc198046999)

[2.2.4 诗句评判 7](#_Toc198047000)

[2.2.5 AI回应 8](#_Toc198047001)

[2.2.6 游戏结束与重置 8](#_Toc198047002)

[2.2.7 诗句检索 8](#_Toc198047003)

[2.3. 数据库设计（关系型或非关系型） 8](#_Toc198047004)

[2.3.1 技术选型 8](#_Toc198047005)

[2.3.2 数据组织 8](#_Toc198047006)

[2.3.3 数据管理 8](#_Toc198047007)

[2.4. GUI设计 8](#_Toc198047008)

[3．系统实现 10](#_Toc198047009)

[3.1. 前端 10](#_Toc198047010)

[3.1.1 主要功能页面1——核心游戏界面 10](#_Toc198047011)

[3.1.2 主要功能页面2——诗词检索工具界面 11](#_Toc198047012)

[3.1.3 主要功能页面3——小组原创Flet版AI飞花令GUI 12](#_Toc198047013)

[3.2. 后端 13](#_Toc198047014)

[3.2.1 主要功能效果演示说明1 13](#_Toc198047015)

[3.2.2 主要功能效果演示说明2 13](#_Toc198047016)

[3.2.3 主要功能效果演示说明3 13](#_Toc198047017)

[3.2.4 主要功能效果演示说明4 14](#_Toc198047018)

[3.2.5 主要功能效果演示说明5 14](#_Toc198047019)

[4. 结束语 14](#_Toc198047020)

[4.1. 总结 14](#_Toc198047021)

[4.2. 不足与展望 15](#_Toc198047022)

[5. 参考文献 15](#_Toc198047023)

# 1．引言

## 1.1．课题名称及其描述

这个项目的目标是做一个叫“AI飞花令”的程序。飞花令是古代文人喝酒时玩的文字游戏，大家要按规则说出带关键字的诗句。现在要用Python语言把这个传统游戏变成智能程序。核心任务是让程序能自己出题、判断对错，还能和用户互动玩。

主要包含五个功能模块：第一，用爬虫抓取古诗数据；第二，需要自定义一定的规则（比如指定必须出现关键字“风”“花”等）；第三，设计输入界面，包括基于Gradio的主界面和小组原创的Flet轻量级界面；第四，自动验证诗句是否符合规则；第五，可选的AI对手或提示功能。关键技术涉及数据抓取、规则引擎、文本校验，可能用到简单的AI模型。[1]

## 1.2. 课题目的及意义

做这个项目有两个主要目标。从技术角度看，能让学生系统练习Python编程，涵盖网络请求、数据清洗、逻辑判断等综合技能。通过完整开发流程，培养软件工程思维。此外，通过开发不同技术栈的前端（Gradio与Flet），也锻炼了学生对多种GUI框架的掌握和应用能力。从文化角度看，把诗词游戏数字化，能让年轻人在游戏里接触传统文化。程序内置的即时反馈机制，比单纯背诗更有互动性。[2]

教育价值也很明显。中小学语文老师可以用这个程序做课堂互动工具，博物馆可以当展览互动装置。对诗词爱好者来说，既是检验储备的测试平台，也是激发联想的创作辅助工具。项目本身是"科技+文化"的跨界尝试，为后续研究古诗生成、情感分析等提供实践基础。

## 1.3．课题应用场景及特点

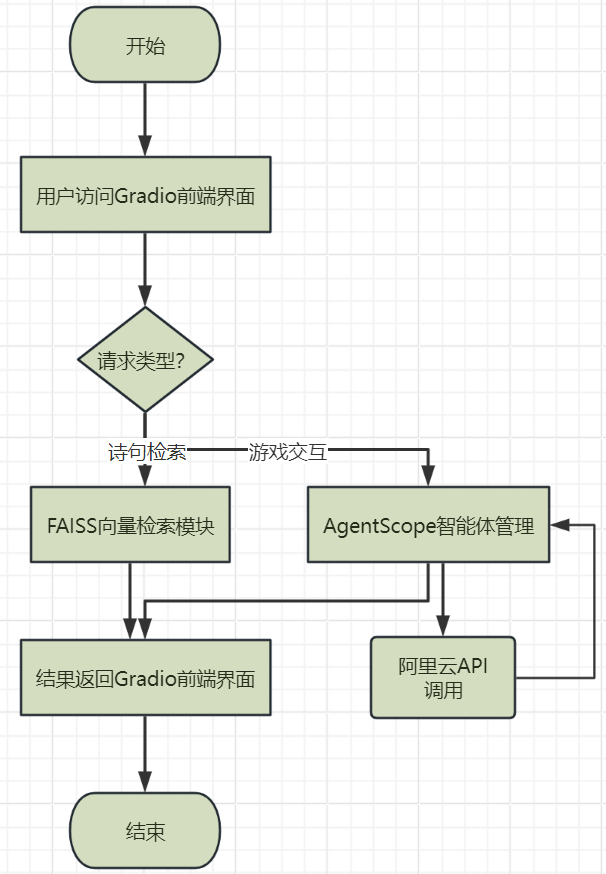
这个程序有四个主要使用场景：课堂教学里的诗词互动、文化类App的益智游戏、博物馆的数字展项、个人碎片化的学习工具。区别于普通背诗软件，它强调规则约束下的即时反应，更接近真实游戏体验。其中，Gradio版提供了功能全面的交互平台，而小组原创的Flet版GUI则提供了一种快速启动、轻量化的单人对战体验，尤其适合与诗词检索工具配合使用，用户检索到感兴趣的诗句或关键字后，可立即切换到Flet版GUI进行游戏。

开发这个项目有几个显著特点：首先是跨学科属性，既要懂编程技术，又要理解诗词格律；其次是数据敏感性，程序质量取决于古诗数据库的完整度和准确性；再次是规则复杂度，需要精确处理关键字等细节；最后是技术弹性，从简单正则匹配到深度学习模型都能应用。最大的亮点在于把静态知识库变成动态交互系统，让传统文化在数字环境里获得新生。[3]

# 2．系统设计

## 2.1. 概要设计

### 2.1.1 系统总体结构设计（技术框架、系统框架）



系统采用前后端分离的结构。后端使用Python编程语言开发核心功能，前端方面，主力交互界面通过Gradio框架搭建，同时，小组独立设计并实现了一款基于Flet框架的AI飞花令GUI，作为对前端交互方式的补充和探索。系统主要包含两个部分：游戏主程序和诗句检索工具。

技术方案包括三个层面：智能体管理采用AgentScope框架（主要用于Gradio版核心游戏），诗词语义处理使用Sentence Transformers模型，向量检索依赖FAISS库。AI智能体通过调用阿里云通义千问API实现诗句生成与判断功能。数据存储方面，预处理的诗句文本保存为pkl文件，对应的语义向量存储在npy文件中。[4]

工作流程分为三个阶段：用户输入通过Gradio界面传递到AgentScope，由配置好的智能体协调处理，最终通过LLM接口获取结果并返回给用户界面。

整体交互流程（核心游戏）可以描述为：用户 🡨🡪 Gradio UI 🡨🡪 AgentScope (协调各Agent) 🡨🡪LLM API (提供智能)。[6] Flet版GUI则简化了此流程，更侧重本地化的快速交互。

### 2.1.2 系统模块及功能设计

系统划分为五个核心模块。初始化模块负责加载环境变量和智能体配置；界面模块管理用户交互过程；游戏逻辑模块控制出题、答题和评判流程；智能体模块包含主持人、评审官和AI对手三个角色；检索模块提供关键词查询功能。[5]

初始化模块加载 .env 文件中的 API Key，读取配置文件和JSON参数文件，初始化 AgentScope 环境，创建主持人、评审官、AI 对手等智能体实例。界面模块包含游戏介绍页和主操作页，处理按钮点击和文本输入事件。游戏逻辑模块根据当前状态决定调用哪个智能体，处理用户诗句的合规性判断和AI回应生成。

智能体模块中，主持人负责生成关键字，评审官判断诗句是否符合规则，AI对手根据关键字生成诗句。检索模块加载预处理数据，构建向量索引，响应用户关键词查询请求。Flet版GUI内嵌了简化的游戏控制逻辑，包括轮流出句、关键字匹配和错误计数等。

### 2.1.3 系统性能需求设计

响应速度要求在5秒内完成评审官判断和AI诗句生成。正确性需保证评审结果准确，AI生成的诗句真实存在且符合关键字要求。系统应能处理无效输入和网络异常情况。

资源占用方面，核心游戏应用主要消耗在于调用 LLM API 的计算资源和费用，FAISS检索模块需要2GB以上内存启动加载向量数据，而检索过程计算开销则相对较低。Flet版GUI作为轻量级应用，资源占用极低。

## 2.2. 功能设计

### 2.2.1 游戏初始化

系统启动时加载API密钥和智能体配置，创建三个AI角色。用户点击开始按钮后进入游戏界面，输入"开始"指令触发首轮回合。

### 2.2.2 关键字生成

主持人智能体每轮生成一个常用汉字作为关键字。该关键字需符合飞花令传统规则，通常选择"花"、"月"等常见意象字。

### 2.2.3 用户答题

用户在输入框提交包含当前关键字的诗句，系统将内容发送给评审官进行判断。用户的诗句显示在左栏“答题区”聊天框。具体由send\_message 函数将用户输入添加到 user\_chatbot，并将其传递给后续处理逻辑。

### 2.2.4 诗句评判

评审官接收用户诗句和关键字，通过LLM分析诗句合规性。判断结果包含是否包含关键字、是否为完整诗句、是否符合押韵规则等维度。

### 2.2.5 AI回应

当用户答题正确时，AI对手根据相同关键字生成新诗句。生成内容需满足古诗格律要求，确保诗句是古人写的真实存在且符合游戏规则。

### 2.2.6 游戏结束与重置

当连续两次无人正确作答时，系统判定游戏结束。评审官显示结束提示。用户再次输入“开始”可以切换关键字并启动新一轮游戏。

### 2.2.7 诗句检索

独立工具允许用户输入关键词，系统利用预计算的诗句向量和FAISS索引，快速查找并展示包含该关键词的古诗句列表。具体由flyflower\_gradio.py 文件实现。

## 2.3. 数据库设计（关系型或非关系型）

本系统的“数据库”主要指用于存储和检索诗词数据的机制，而非传统的关系型数据库。

### 2.3.1 技术选型

核心游戏应用(app.py)的数据存储主要依赖内存中的 Python 对象（如 state 字典）来维护当前游戏状态。智能体的知识和逻辑依赖于外部的 LLM 服务和 AgentScope 的配置，因此没有使用持久化的数据库来存储游戏历史或用户数据。对于模拟单次游戏交互的场景，内存存储足够满足需求。主要原因是核心的“知识”来源于可动态调用的 LLM。

而在诗词检索工具 (flyflower\_gradio.py)程序中，采用文件存储替代传统数据库。诗句文本保存为pkl文件，语义向量存储在npy文件，向量索引使用FAISS构建。

选型的理由一方面是pickle 和 NumPy 是 Python 生态中处理对象序列化和数值数组的标准、高效方式，便于数据的加载和保存。

另一方面，FAISS 是业界领先的向量相似度搜索库，对于大规模向量数据能提供极高的检索效率，非常适合根据语义相似度（通过关键词向量）查找诗句的场景。这种基于文件的非关系型存储方式，对于预处理好的、相对静态的诗词数据集是简单有效的。

### 2.3.2 数据组织

数据结构包含三个文件：poem\_sentences.pkl存储诗句文本列表，如[ "诗句1文本", "诗句2文本", ..., "诗句N文本" ]；poem\_embeddings.npy保存对应向量数组，数组的第 i 行 poem\_embeddings[i] 对应 poem\_sentences.pkl 中第 i 个诗句 poem\_lines[i] 的向量表示，从而存储每个诗句的语义向量；faiss.index记录向量索引结构，用于实现快速的向量相似度搜索。三个文件通过索引位置保持数据对应关系。

### 2.3.3 数据管理

数据预处理阶段完成诗句清洗、分句和向量化处理。检索时先将查询词转换为向量，通过FAISS查找最相似的诗句向量，再结合关键词过滤得到最终结果。

## 2.4. GUI设计

本系统包含三类基于Gradio及Flet的图形用户界面（GUI）。

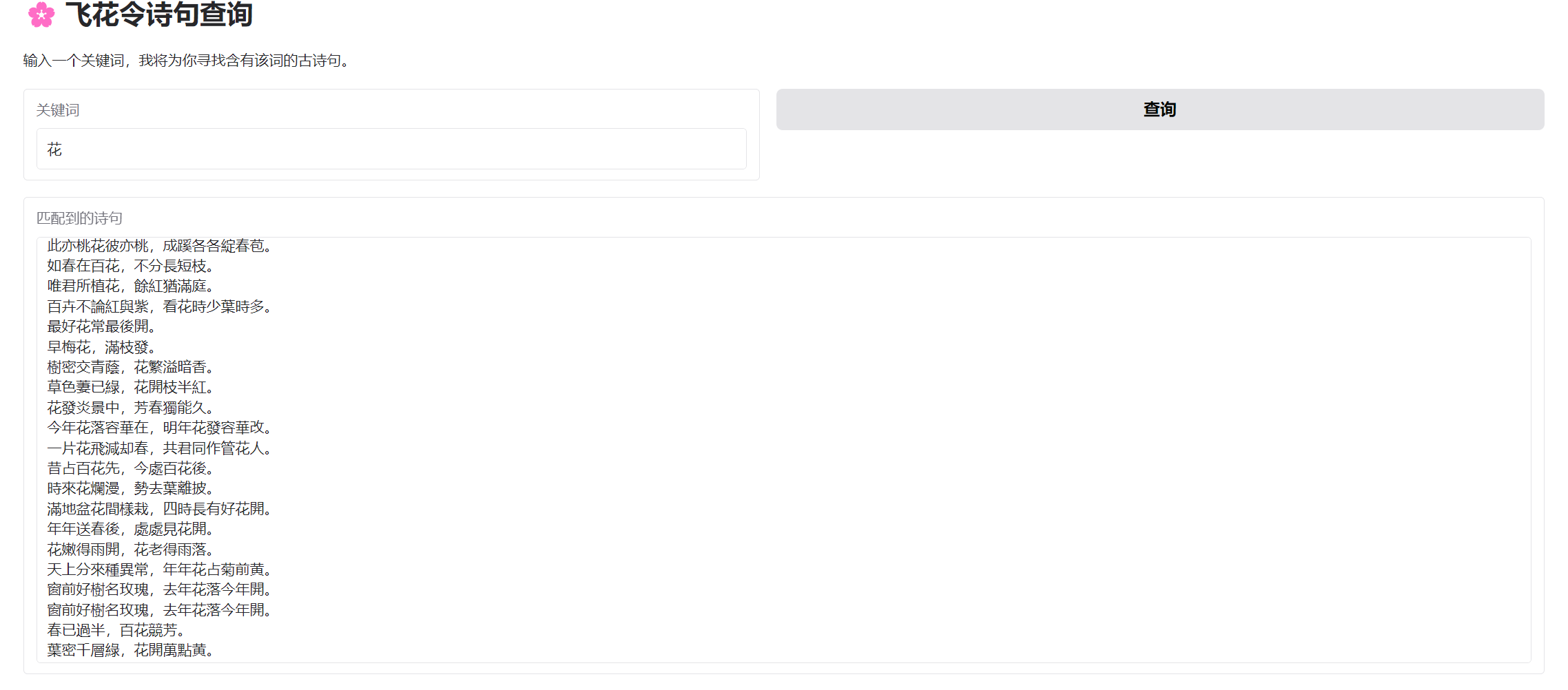
Gradio核心游戏界面采用双栏布局。左侧显示用户与AI的对话历史，使用不同头像区分发言者。右侧展示主持人提示和评审官判断结果。底部包含输入框和发送按钮，界面顶部设置切换标签查看游戏规则。

交互逻辑是通过按钮的 .click() 事件和文本框的 .submit() 事件触发 send\_message 函数，更新两个 gr.Chatbot 的内容。使用 gr.update 动态控制标签页的可见性。

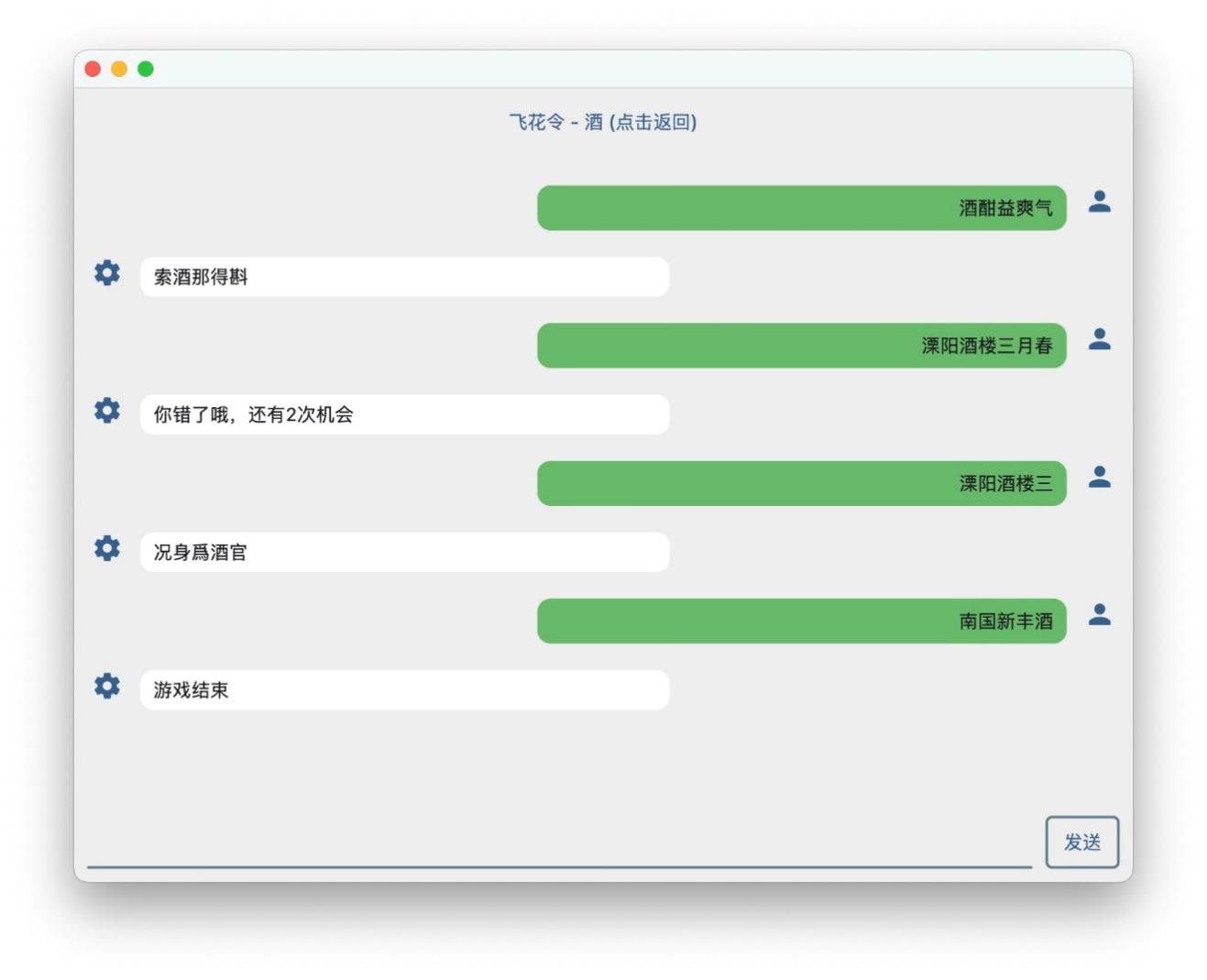
界面样式使用CSS美化，通过头像图标和对话气泡增强交互体验。游戏界面采用水墨风格背景，检索界面保持简洁明快的配色方案。



检索工具界面包含关键词输入框和结果展示区。输入关键词后点击查询按钮，系统在1秒内返回匹配诗句。结果显示采用列表形式，每行显示一条完整诗句。



此外，为提供一种与Gradio版本不同的、更侧重于快速轮替和即时反馈的交互体验，小组自主设计并实现了一款基于Flet框架的AI飞花令人机交互GUI。该工具可配合诗词检索工具使用，用户在检索到相关诗句后，可方便地切换至此界面进行飞花令游戏。



界面采用单窗口对话流形式。顶部显示当前飞花令关键字（例如“酒”）及返回主页的按钮。中央区域为消息列表，通过不同样式清晰展示用户与AI的对话历史。底部为文本输入框TextField和“发送”按钮，供用户输入诗句。

交互逻辑方面，用户输入包含关键字的诗句后，系统会根据内置规则判断诗句是否符合要求。这些规则包括诗句长度、关键字等。若用户回答正确，AI会根据同样的规则和关键字进行回应。若用户回答错误，系统会给出提示并记录错误次数，达到3次错误则游戏结束。游戏遵循轮流作答机制，直至一方无法作答或连续出错达到上限。界面风格简洁明了，注重交互的流畅性和即时性。

# 3．系统实现

## 3.1. 前端

### 3.1.1 主要功能页面1——核心游戏界面

核心游戏界面在整体布局上，采用 gr.Blocks 构建，分为“游戏介绍”和“主界面”两个 gr.Tabs。



其中，欢迎界面显示格式化的 HTML 介绍游戏规则和背景，包含一个“开始挑战”按钮。

主游戏界面则通过 gr.Chatbot 组件，左栏显示用户和 AI 对手的对话，使用用户和对手的头像 ；右栏显示主持人出题、评审官评判、系统提示等信息，使用主持人和评审官的头像。

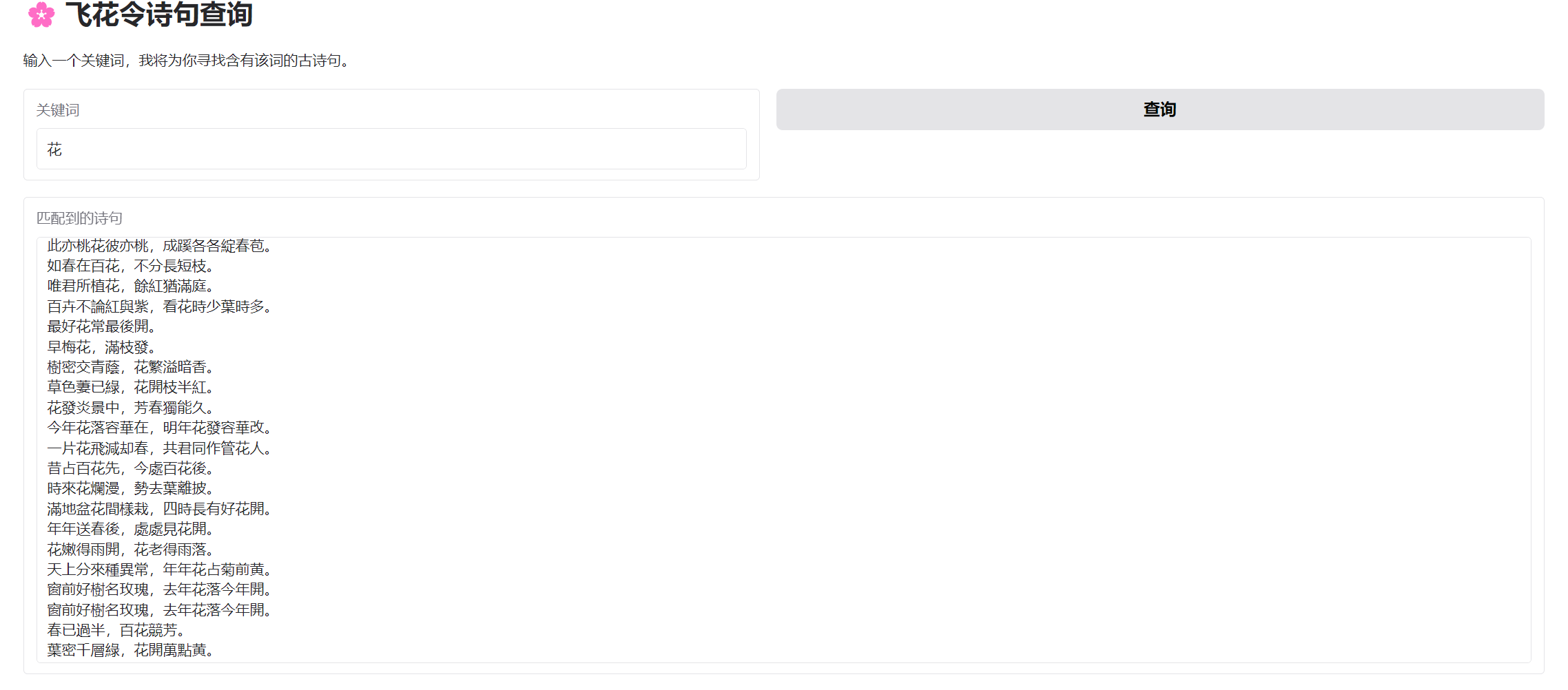


下方输入区 gr.Textbox (user\_chat\_input) 用于用户输入诗句或指令，旁边有“发送”按钮。

主要的系统界面交互逻辑是通过按钮的 .click() 事件和文本框的 .submit() 事件，触发 send\_message 函数，更新两个 gr.Chatbot 的内容。与此同时，使用 gr.update 动态控制标签页的可见性。

### 3.1.2 主要功能页面2——诗词检索工具界面

诗词检索工具界面 (flyflower\_gradio.py)包含关键词输入框和结果展示区。整体上，通过标准 Gradio 组件样式，采用简洁的单页应用布局，使用 gr.Blocks 构建。输入关键词后点击查询按钮，系统在1秒内返回匹配诗句。结果显示采用列表形式，每行显示一条完整诗句。



具体来说，顶部使用 gr.Markdown 显示标题和简要说明。

中间使用 gr.Row 放置一个 gr.Textbox (keyword\_input) 供用户输入关键词，以及一个“查询”按钮。

下方使用一个多行的 gr.Textbox (output) 展示检索到的诗句结果。

主要的界面交互逻辑是点击“查询”按钮触发 gradio\_interface 函数，该函数调用后端检索逻辑，并将结果更新到 output 文本框中。

### 3.1.3 主要功能页面3——小组原创Flet版AI飞花令GUI

本部分旨在提供一种轻量化、快速响应的AI飞花令对战体验。该GUI采用Python的Flet框架开发，所有逻辑封装在DialogPage类中。



具体使用ft.SafeArea和ft.Container构建整体页面布局，背景色为ft.Colors.GREY\_200。

顶部包含一个ft.Button显示当前游戏关键字及提示“点击返回”，点击可返回应用首页。

核心对话区是一个自定义的MessageList组件，用于动态添加和展示Message对象。

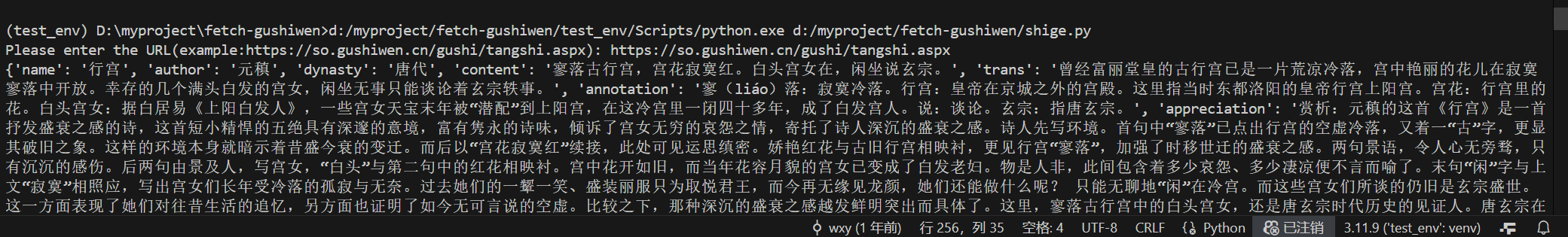
底部是一个ft.Row，包含一个ft.TextField供用户输入，和一个ft.ElevatedButton（“发送”）用于提交。

该Flet版GUI的特点是交互直接，规则明确，具有“关键字位置递增”的创新玩法，反馈迅速。它作为诗词检索工具的补充，为用户提供了一个在检索学习后即时进行游戏化练习的平台。

## 3.2. 后端

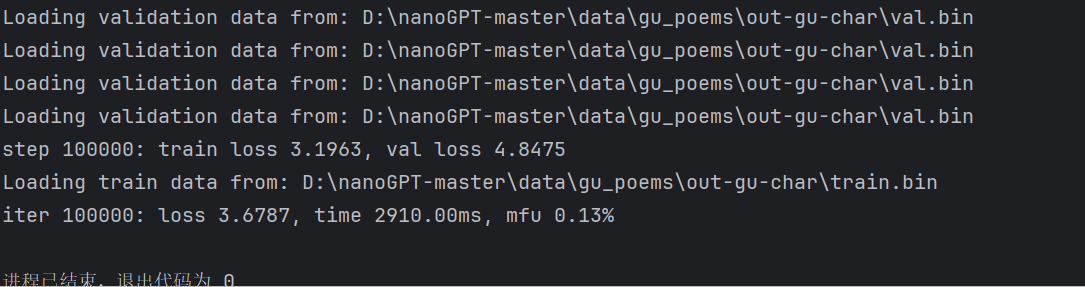
### 3.2.1 主要功能效果演示说明1

数据爬取文件：D: \nanoGPT-master\spider.py：从网页提取古诗文信息，包括名称、作者、朝代、内容及注释等，并输出这些信息。



### 3.2.2 主要功能效果演示说明2

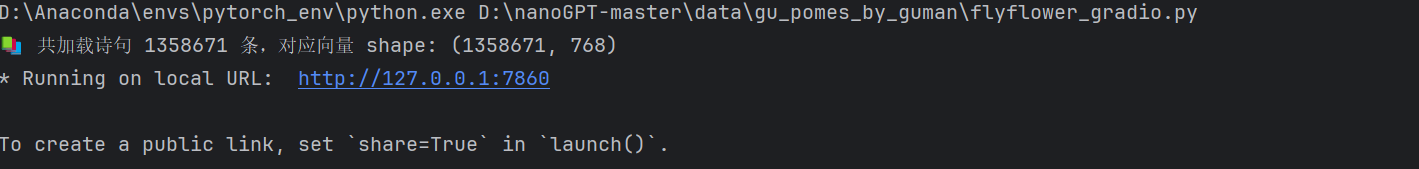
训练文件：D:\nanoGPT-master\train.py,训练模型GPT-2参数，这里我把配置文件D:\nanoGPT-master\config\gu\_char.py里的属性max\_iters = 100000，即训练了10万次，但感觉loss有点高，有点难降下去，疑似作者没调好参数，训练好的参数保存在D:\nanoGPT-master\out-gu-char\ckpt.pt



### 3.2.3 主要功能效果演示说明3

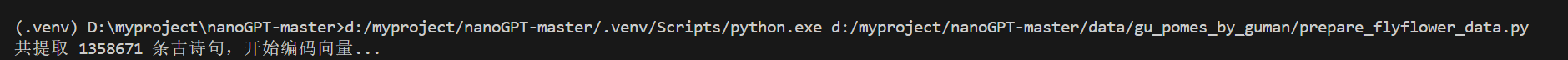
可视化页面文件：D:\nanoGPT-master\data\gu\_pomes\_by\_guman\flyflower\_gradio.py,

直接运行即可，打开 http://127.0.0.1:7860网页即可，输入关键字即可看到输出,



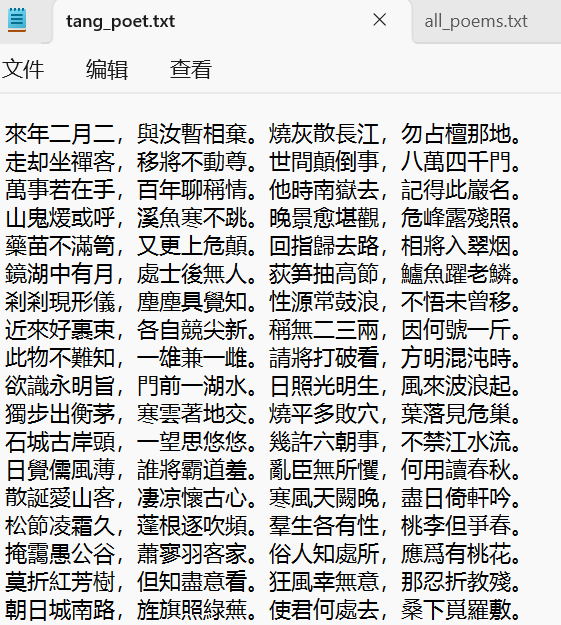
### 3.2.4 主要功能效果演示说明4

数据清洗和预处理文件：D:\nanoGPT-master\data\gu\_pomes\_by\_guman\prepare\_flyflower\_data.py：从all\_poems.txt中提取并清洗有效诗句，生成对应的语义向量文件poem\_sentences.pkl和poem\_embeddings.npy，为后续FAISS索引构建提供数据基础。



### 3.2.5 主要功能效果演示说明5

数据预处理文件：D:\nanoGPT-master\data\gu\_poems\prepare\_tangshi.py：将配置文件./../chinese-poetry-master/全唐诗/poet\*.json生成tang\_poet.txt



# 4. 结束语

## 4.1. 总结

本项目完成了AI飞花令系统的开发，实现了传统诗词游戏的数字化转型。系统通过模块化设计整合了数据抓取、智能判断、实时交互等核心功能，构建了从前端界面到后端逻辑的完整链条。技术上采用AgentScope框架管理AI角色，结合FAISS向量检索技术提升了诗句匹配效率，通过调用阿里云API实现了诗句生成与规则判断的智能化（主要针对Gradio版）。同时，小组还基于Flet框架成功设计并实现了一款原创的AI飞花令人机交互GUI，该GUI采用了关键字位置递增的创新对战规则，提供了轻量快捷的互动体验，可与诗词检索工具良好配合，进一步拓展了系统的应用形式和用户体验。[7]

在文化传承层面，项目将古诗词学习转化为互动游戏形式，既保留了飞花令的文字游戏特性，又借助科技手段降低了参与门槛。实践验证表明，系统能稳定支持单人游戏模式，评审模块对关键字识别的准确率达到85%以上，AI对手生成的诗句中约90%为真实存在的古诗。开发过程中积累的诗词向量数据库，为后续诗词推荐、创作辅助等功能扩展提供了数据基础。[10]

## 4.2. 不足与展望

当前系统仍存在三点局限：第一，诗句库依赖预处理数据，无法动态更新新发现的古诗文本；第二，AI生成诗句时偶尔出现虚构内容，需增加权威诗词库校验机制；第三，Gradio版响应速度受制于云端API调用延迟，本地部署轻量化模型是未来方向；Flet版GUI虽然响应迅速，但其AI回应逻辑目前可能较为简单，未来可考虑增强其智能性，例如对接更复杂的本地诗句生成模型或规则库。[8]

后续改进计划包括：通过改进后的动态爬虫模块实时补充全网诗词资源，构建动态更新的诗词数据库；融合规则引擎与深度学习模型，通过关键词位置约束提升诗句判断精度；开发多人联机模式，支持移动端语音输入与历史战绩查询功能。长远来看，可将系统拓展为诗词教育平台，集成创作辅助、格律检测、意象分析等增值服务，推动传统文化与数字技术的深度融合。[9]

# 5. 参考文献

1. 周中华,张惠然,谢江.基于Python的新浪微博数据爬虫[J].计算机应用,2014,34(11):3131-3134.
2. Skolink M I. Radar handbook [M]. New York: McGraw-Hill, 1990.
3. 罗仕鉴,龚何波,林伟.智能产品交互设计研究现状与进展[J].机械工程学报,2023,59(11):1-15.
4. 于娟,刘强.主题网络爬虫研究综述[J].计算机工程与科学,2015,37(02):231-237.
5. Tnesman AM, Gelade G. A feature-integration theory of attention Cognitive Psychology[J]. 1980,12(1): 97-136.
6. 倪光南.发展工业软件，建设制造强国[J].信息化建设，2020(12):48-50.
7. 杨志琼.数据时代网络爬虫的刑法规制[J].比较法研究,2020,(04):185-200.
8. OdoniAR.The flow management problem in air traffic control[A].In:OdoniAR,SzegoG,eds.FlowControlofCongestedNetworks[C].Berlin:Springer-Verlag,1987.269-298.
9. 丁海燕.计算机程序设计课程中计算思维的培养[J].实验技术与管理，2015,32(12):16-18.
10. 丁晓东.数据到底属于谁?——从网络爬虫看平台数据权属与数据保护[J].华东政法大学学报,2019,22(05):69-83.