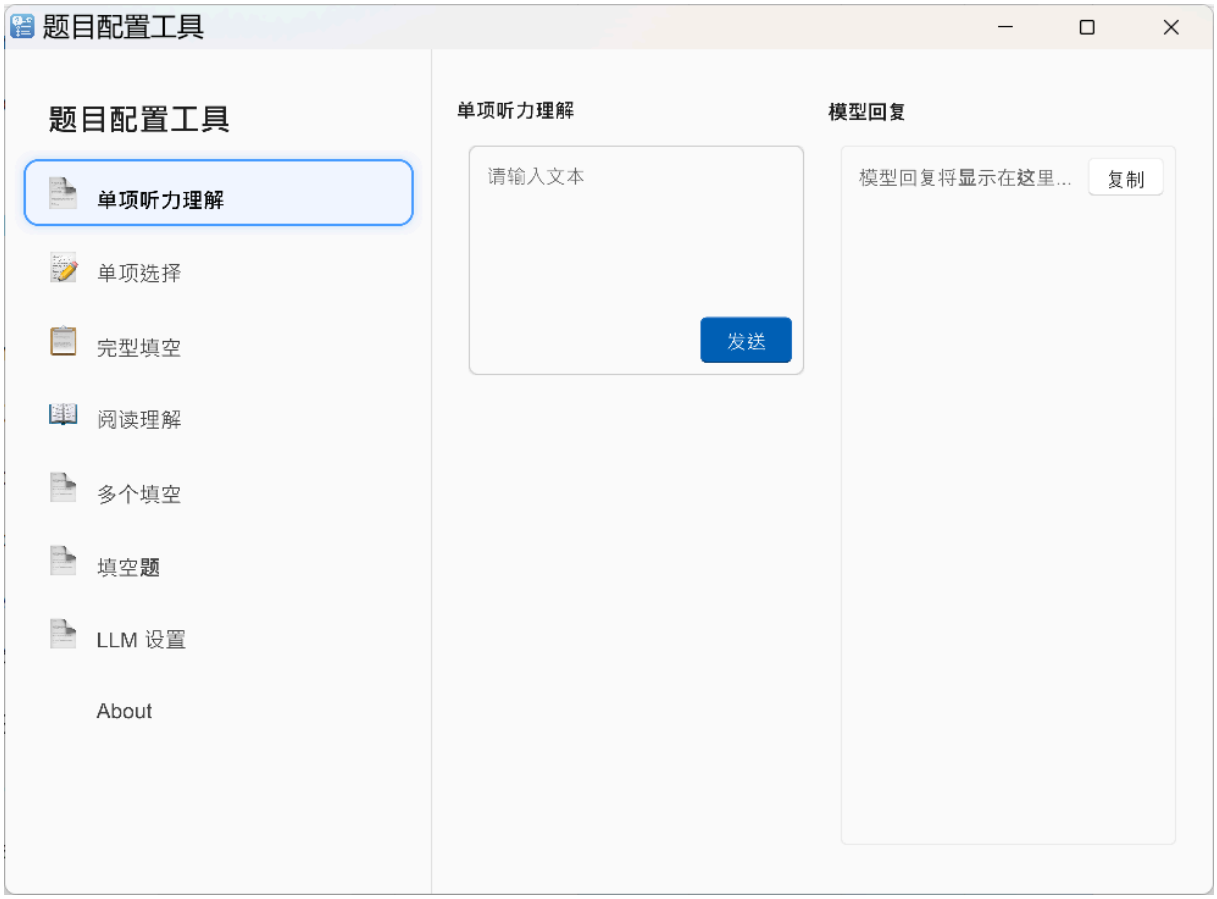
Question Tool 智能题目录入工具介绍

目录

1. 项目概述

1.1. 产品简介



Question Tool 是一个基于 Rust 和 Slint UI 框架开发的智能题目录入工具，专为教育机构和内容创作者设计。通过集成多种大语言模型后端（OpenAI GPT、GitHub Models 等），实现从剪贴板内容自动生成标准化题目，并通过 JavaScript 自动化脚本完成网页表单的批量录入。

1.2. 核心价值主张

1.2.1. 效率革命

- 手动录入效率提升 **20** 倍
- 每周节省 **10+** 小时 工作时间

1.2.2. 智能化生成

- AI 驱动的内容生成，确保专业质量
- 六种题型全覆盖，标准化输出
- 从内容识别到网页填充的完整自动化流程

2. 功能特性

2.1. 支持的题目类型

Question Tool 目前支持以下六种标准化题目类型：

2.1.1. 单选题

Single Choice

1. 自动生成选项 A、B、C、D
2. 智能答案标记和解析
3. 符合标准化考试格式

2.1.2. 完形填空

Cloze Test

1. 文章挖空处理
2. 选项匹配和语法分析
3. 自动生成标注

2.1.3. 单项听力理解

Listening Single

1. 音频材料描述生成
2. 口语化表达识别
3. 情景对话分析

2.1.4. 阅读理解

Reading Comprehension

1. 文章段落格式化处理
2. 多题目组合生成
3. 考点分析和答题技巧

2.1.5. 听力复合题

Listening Compound

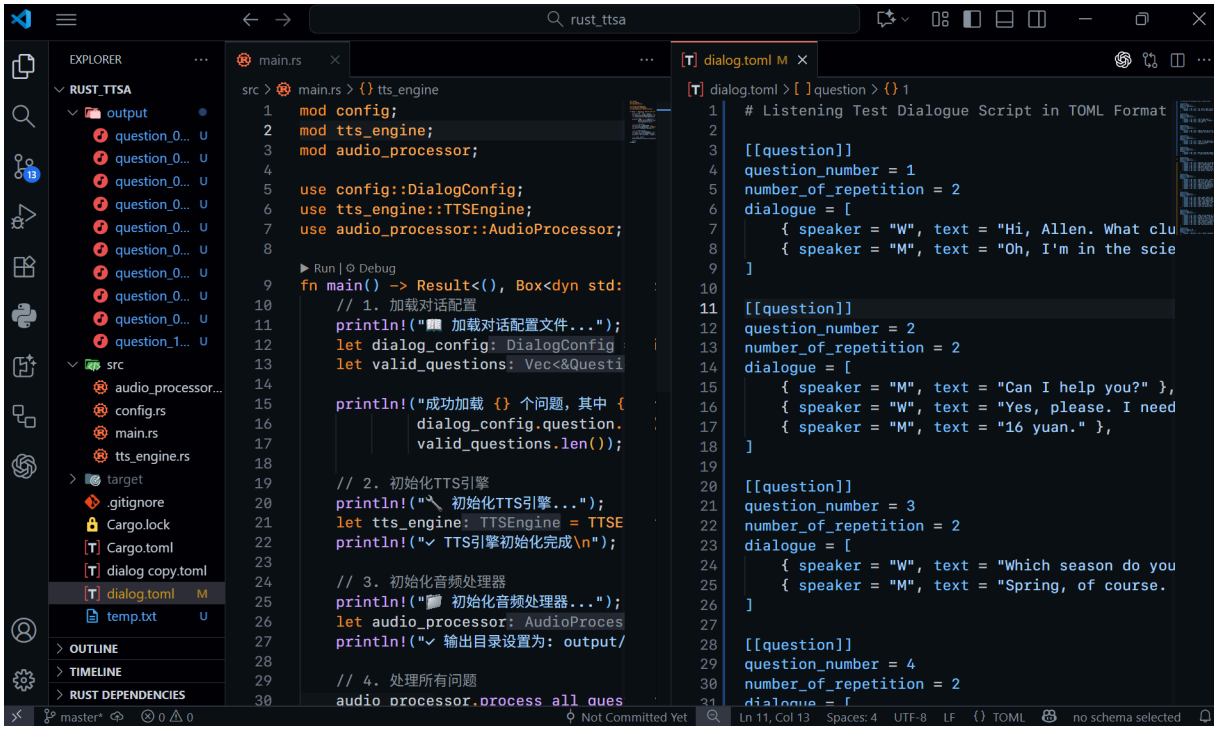
1. 长对话材料处理
2. 智能答案标记和解析
3. 复杂逻辑关系分析

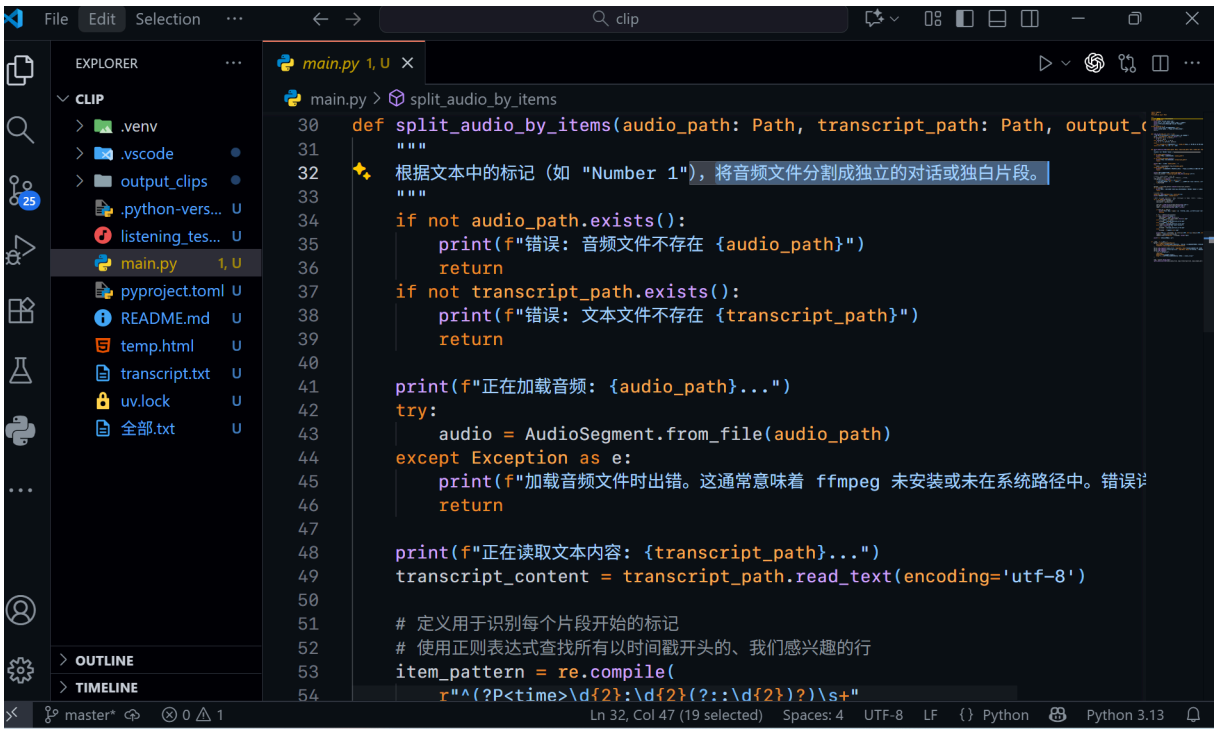
2.1.6. 多项填空

Multi Blank Filling

1. 知识点填空
2. 概念定义匹配
3. 智能答案标记和解析

以及听力音频的生成和处理。（待整合，代码已完成） 用户只需将文本复制到剪贴板，Question Tool 即可自动识别对话类型并自动生成标准的 Toml 文件，进而生成全部音频文件。





2.2. 技术架构

Question Tool 系统架构

前端 ↔ 后端 ↔ 自动化脚本

前端界面

框架: Slint UI
Rust 原生 UI 框架

特性:
1. 现代化界面设计
2. 流畅的用户体验
3. 实时剪贴板监控
4. 内容预览功能

后端处理

语言: Rust
高性能、内存安全

组件:
1. Tokio 异步运行时
2. 多 LLM 后端支持
3. 剪贴板监控服务
4. 内容处理引擎

自动化脚本

语言: JavaScript
浏览器执行环境

功能:
1. DOM 智能操作
2. 表单自动填充
3. 用户事件模拟
4. 跨浏览器兼容

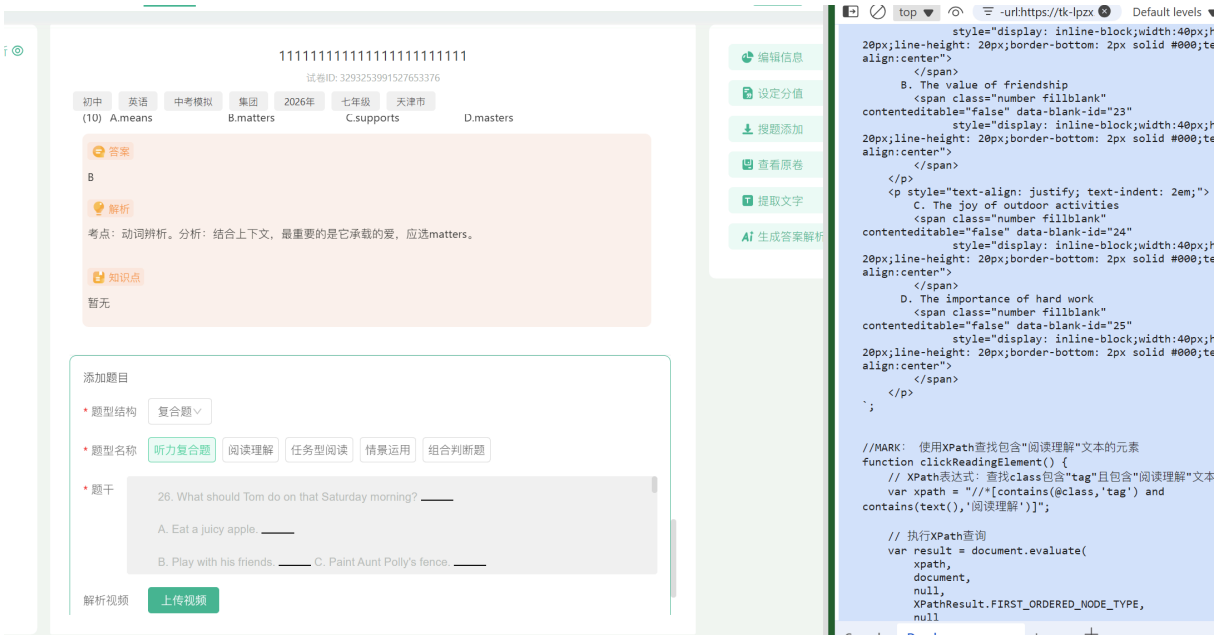


图 1 自动化脚本工作

3. 性能数据与优化

3.1. 效率对比分析

题目类型	手动录入时间	自动化处理时间	效率提升倍数
完形填空	20 分钟	50 秒	24×
阅读理解	15 分钟	40 秒	22.5×
单选题	8 分钟	35 秒	13.7×
听力题	12 分钟	120 秒	6×

表 1　对比

3.2. 系统性能优化

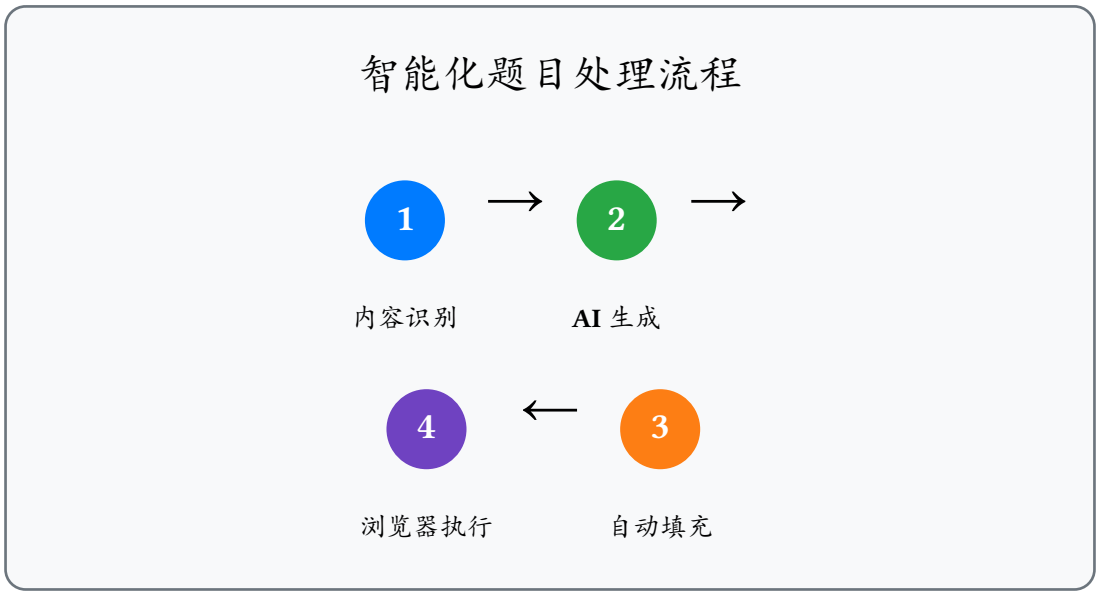
3.2.1. 内存管理优化

优化前:　349MB
优化后:　80MB
优化幅度:　77% ↓

3.2.2. 运行时性能

启动时间	< 2 秒
响应延迟	< 100ms
稳定性	7×24 小时
内存泄漏	零检出

4. 工作流程



4.1. 详细流程说明

4.1.1. 内容识别

1. 实时监控剪贴板内容
2. 自动识别图片中的文本 (OCR)
3. 智能分析内容类型和结构
4. 预处理和格式规范化

4.1.3. AI 生成

1. 根据内容类型选择合适的提示模板
2. 调用 LLM API 生成标准化题目
3. 确保输出格式符合平台要求
4. 多轮优化和内容校验

4.1.2. 自动填充

1. 生成对应的 JavaScript 自动化代码
2. 模拟用户操作填充网页表单
3. 处理各种表单控件和验证逻辑
4. 批量处理和错误重试机制

4.1.4. 浏览器执行

1. 在浏览器环境中执行自动化脚本
2. 兼容主流浏览器 (Chrome、Firefox、Edge)
3. 处理动态加载和异步内容
4. 提供执行日志和结果反馈

4.2. 开发背景

4.2.1. 项目起源

本项目由河南工业大学计算机科学与技术专业大三学生王浩然在郑州新东方录排实习期间开发。针对教育内容制作中的实际痛点，提升工作效率。

4.2.2. 解决的问题

传统工作流的挑战:

- 手动录入题目耗时长，容易出错
- 格式标准化要求严格，重复性工作量大
- 不同题型的处理逻辑复杂，难以标准化

技术解决方案:

- AI 驱动的内容生成，确保质量和一致性
- 自动化脚本处理重复性操作
- 模块化架构支持快速扩展新题型
- 内存优化确保长时间稳定运行

4.3. 技术创新

4.3.1. 多后端 LLM 支持

```
pub enum LLMBackend {
    OpenAI,
    GitHub,
    // 易于扩展新的 AI 服务
}
```

4.3.2. 智能提示模板系统

每种题型都有专门优化的提示模板，确保生成内容的专业性和准确性。

4.3.3. 内存管理优化

- 临时文件自动清理机制
- 图片对象智能释放
- 全局 Tokio Runtime 复用

4.3.4. 跨平台兼容性

基于 Rust 和 Slint 的技术栈确保在 Windows、macOS 和 Linux 上的一致体验。

4.4. 未来规划

4.4.1. 短期优化 **(v0.2.0)**

- 性能提升: 进一步优化代码执行速度
- **UI** 改进: 增强用户界面的交互体验
- 错误处理: 完善异常情况的处理和恢复机制

4.4.2. 中期发展 **(v1.0.0)**

- **API** 集成: 支持直接访问题库数据库
- 题型扩展: 增加更多专业领域的题目类型
- 多个学科支持: 扩展到数学、物理等多个学科领域
- 并发处理: 支持多任务并发执行，提高处理效率

4.5. 技术规格

4.5.1. 系统要求

- 操作系统: Windows 7+ / macOS 10.15+ / Ubuntu 18.04+
- 内存: 最低 4GB，推荐 8GB
- 存储: 至少 100MB 可用空间
- 网络: 稳定的互联网连接 (用于 AI API 调用)

4.5.2. 开发环境

- **Rust**: 1.75+
- **Node.js**: 16+ (用于构建脚本)
- **Git**: 版本控制和协作

4.6. 实际应用案例

4.6.1. 案例一：完形填空题目处理

场景描述： 处理一套标准英语完形填空题，包含 20 个空格，需要生成选项和标准答案。

处理流程：

1. 从剪贴板获取原始文章内容
2. AI 识别需要挖空的关键词位置
3. 识别选项
4. 自动生成考点分析和解题思路
5. 一键填充到网页表单中

效率对比：

- 传统方式：20 分钟（包括选项设计、格式调整、手动录入）
- 自动化处理：50 秒（AI 生成+自动填充）

- 效率提升：24 倍

4.6.2. 案例二：阅读理解题组

场景描述： 处理包含 5 道题目的阅读理解材料，涉及细节理解、推理判断等多种题型。

处理过程：

1. 从剪贴板获取原始文章内容
2. 创建题目
3. 每道题目配备详细的解析说明
4. 自动设置答案

效率提升：

- 传统耗时：15 分钟
- 自动化耗时：50 秒
- 时间节省：20 倍效率提升

5. 总结与展望

5.1. 项目价值与意义

Question Tool 代表了 教育技术领域 **AI** 应用 的一个成功实践案例。通过将前沿的语言模型技术与实际的教育内容制作需求相结合，成功创造了一个能够显著提升工作效率的实用工具。

5.1.1. 技术层面的突破

1. 性能优化: 24 倍效率提升，内存占用降低 77%
2. 架构设计: 模块化、可扩展的系统架构
3. 技术创新: Rust + AI + 自动化的结合
4. 跨平台兼容: 支持主流操作系统和浏览器

5.2. 致谢

感谢 郑州新东方 提供的实习机会和实际应用场景，

感谢指导老师和同事们的支持与建议，

感谢开源社区提供的优秀技术栈和工具。

文档版本: v1.0 | 更新日期: 2025 年 10 月 | 作者: 王浩然