My AI+Education Application Demo presentation

This document includes a demo video and presentation of: *AI-Powered Math Exam Analysis*, an application that takes a student's exam results as input and outputs:

1. **An analysis of their strengths and weaknesses across different knowledge modules.**
2. **Targeted practice questions addressing the student's knowledge gaps**

Demo video on Github: Link

Backup demo video on Tencent Doc: Link

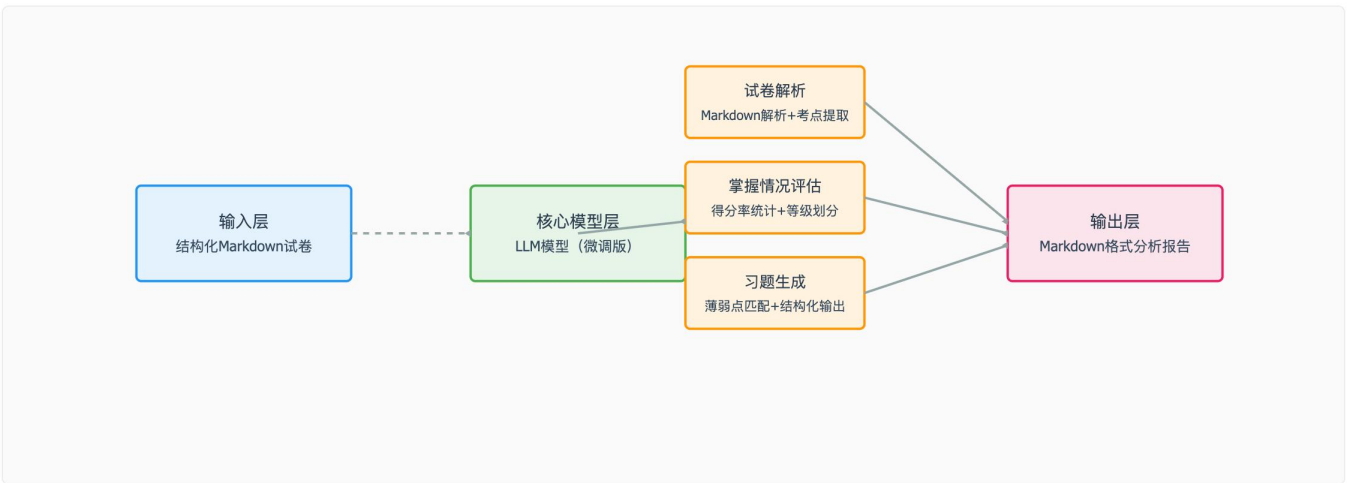
Presentation:

Cursor适配LLM试卷分析与习题生成Demo - 田小红

版本：v1.0 | 核心：轻量化LLM模型封装与Cursor集成

1. 核心功能模块

1.1 功能架构



2. 模型选型与优化策略

2.1 模型选型

选择公有云旗舰模型**通义千问3.0**作为核心LLM模型。通义千问3.0通过API接入方式部署，具有以下优势：

- **精度高**：作为旗舰级大语言模型，在试卷解析、知识模块匹配、主观题评分等任务上表现优异
- **无需微调**：模型本身具备强大的理解与生成能力，可直接应用于试卷分析与习题生成场景
- **维护成本低**：无需本地GPU硬件支持，无需模型训练与部署维护，降低技术门槛
- **部署便捷**：通过API密钥配置即可接入Cursor编辑器，实现快速集成

系统通过Cursor API Integration方式接入通义千问3.0，用户在Cursor编辑器中输入指令后，系统自动调用通义千问3.0 API完成试卷分析与习题生成任务。

2.2 模型微调优化

微调目标：优化Markdown格式解析准确率、知识模块匹配精度、主观题评分一致性、习题生成相关性

数据集设计：

```
{ "dataset_type": "结构化试卷分析数据集", "data_scale": "5k题目 + 2万作答样本", "data_fields": [ "markdown试卷文本", "知识模块标注（三级标签）", "主观题评分标准", "薄弱点分析示例", "针对性习题示例" ], "quality_requirement": "标注准确率≥98%" }
```

4. 核心技术实现逻辑

4.1 试卷解析逻辑

模型通过Prompt指令实现Markdown结构化解析，核心逻辑如下：

```
// 核心解析流程（伪代码） function parseMarkdownExam(examMarkdown) { // 1. 结构化字段提取 const questions = extractQuestions(examMarkdown, [ '题目编号', '题型', '分值', '题干', '学生答案', '参考答案' ]); // 2. 考点提取与模块匹配 questions.forEach(question => { const coreKnowledge = llm.extractCoreKnowledge(question.stem); question.knowledgeModules = llm.matchKnowledgeModules(coreKnowledge, knowledgeTree); }); // 3. 自动评分 questions.forEach(question => { if (question.type === '客观题') { question.score = question.studentAnswer === question.referenceAnswer ? question.fullScore : 0; } else { // 主观题LLM评分 const scoreResult = llm.gradeSubjectiveAnswer( question.stem, question.studentAnswer, question.referenceAnswer, question.fullScore ); question.score = scoreResult.score; question.deductionReason = scoreResult.reason; } }); return questions; }
```

4.2 掌握情况评估逻辑

基于解析后的题目数据，实现知识模块掌握情况量化评估：

```
// 掌握情况评估流程（伪代码） function evaluateMastery(questions) { // 1. 按知识模块分组统计 const moduleStats = groupByKnowledgeModule(questions); // 2. 计算得分率与等级 moduleStats.forEach(module => { module.totalScore = sum(module.questions.map(q => q.fullScore)); module.actualScore = sum(module.questions.map(q => q.score)); module.scoreRate = (module.actualScore / module.totalScore * 100).toFixed(1) + '%'; // 等级划分 if (module.scoreRate >= 85) module.masteryLevel = '优秀'; else if (module.scoreRate >= 70) module.masteryLevel = '良好'; else if (module.scoreRate >= 50) module.masteryLevel = '薄弱'; else module.masteryLevel = '极差'; }); // 3. 薄弱点深度分析 const weakModules = moduleStats.filter(m => m.masteryLevel === '薄弱' || m.masteryLevel === '极差'); weakModules.forEach(module => { module.weakPointAnalysis = llm.analyzeWeakPoints(module.questions); }); return { moduleStats, weakModules }; }
```

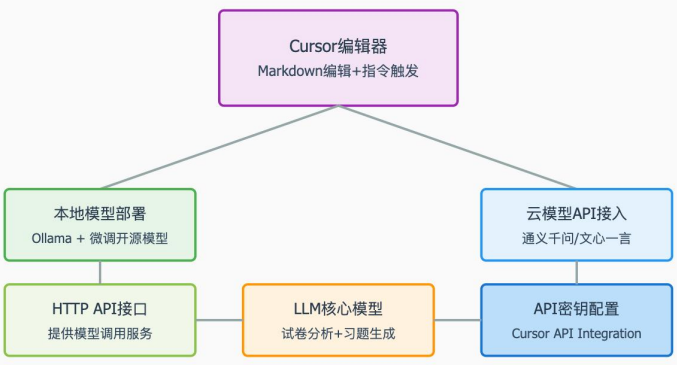
4.3 习题生成逻辑

基于薄弱模块生成个性化习题，核心逻辑包括考点匹配、难度控制、格式标准化：

```
// 习题生成流程（伪代码） function generatePracticeQuestions(weakModules, miniQuestionBank) { const practiceQuestions = []; weakModules.forEach(module => { // 提取核心考点 const corePoints = module.weakPointAnalysis.map(analysis => analysis.corePoint); // 按难度比例筛选/生成题目 const basicCount = Math.ceil(corePoints.length * 0.6); const advancedCount = Math.ceil(corePoints.length * 0.3); const extendCount = Math.ceil(corePoints.length * 0.1); // 基础题（直接筛选） const basicQuestions = miniQuestionBank.filter(q => corePoints.includes(q.corePoint) && q.difficulty === '基础' ).slice(0, basicCount); // 提升题（变式生成） const advancedQuestions = corePoints.slice(0, advancedCount).map(point => llm.generateVariantQuestion(point, '提升', miniQuestionBank) ); // 拓展题（跨模块综合） const extendQuestions = corePoints.slice(0, extendCount).map(point => llm.generateComprehensiveQuestion(point, module.knowledgeModule, miniQuestionBank) ); // 合并并补充解析 const moduleQuestions = [...basicQuestions, ...advancedQuestions, ...extendQuestions].map(q => ({ ...q, analysis: llm.generateQuestionAnalysis(q.stem, q.answer), targetWeakPoint: point })); practiceQuestions.push({ module: module.knowledgeModule, questions: moduleQuestions }); }); return practiceQuestions; }
```

5. Cursor编辑器集成实现

5.1 模型接入架构



5.2 指令设计与触发机制

核心指令规范：通过标准化指令触发模型功能，支持基础功能与自定义需求：

```
// 1. 基础功能指令（默认参数） "分析试卷：请解析上述Markdown试卷，输出知识模块掌握情况、薄弱点分析及个性化练习题。" // 2. 自定义难度指令 "分析试卷（调整难度）：请解析上述试卷，针对薄弱模块生成“基础题占比80%、提升题20%”的练习题。" // 3. 指定模块分析指令 "分析试卷（指定模块）：请重点分析“二次函数”模块的掌握情况，生成5道针对性练习题。" // 4. 输出格式自定义指令 "分析试卷（简化输出）：仅输出薄弱模块清单及对应的基础练习题，无需详细解析。"
```

触发机制：用户在Cursor的Markdown文件末尾输入指令后，Cursor通过自定义模型接口调用LLM，模型接收"试卷文本+指令"作为输入，输出结构化Markdown报告，直接插入到Cursor编辑器中。

> **技术要点**：指令设计采用"固定前缀+参数配置"的模式，便于模型解析用户需求；输出格式严格遵循Markdown规范，确保在Cursor中可直接编辑、复制、导出。