教育平台智能助手模型选型报告

1. 评估概述

1.1 背景与目标

随着教育场景对智能问答与个性化辅导需求的快速增长，选择一款高效、稳定且成本可控的中文大模型成为平台建设的关键。本次评测旨在**定量**与**定性**地对比主流中文 LLM，在统一的 RAG（Retrieval-Augmented Generation）流水线下，找出最契合我校教学平台的模型方案。

1.2 受测模型

模型	版本 / 来源	关键词	备注
ChatGLM3-6B	智谱 AI API	轻量、响应快	
DeepSeek-V3	DeepSeek API	高质量长文本	
Qwen-7B-Chat	阿里云 DashScope API	代码示例丰富	
Yi-6B-Chat	01.AI API	高 Source Recall	

> *Baichuan2-13B-Chat* 因连续出现 404 与 429 错误，数据缺失，已在本轮分析中排除。

1.3 测试数据与流程概览

- **数据集**：200 条来自《嵌入式 Linux 开发实践教程》的真实教学问答（概念 60% / 实操 40%）。
- **统一 RAG**：FAISS + 余弦召回 Top-k=4，确保检索阶段一致。
- **计量指标**：BERTScore-F1、Rouge-L、Source Recall、平均延迟、输出 Token 数。
- **执行环境**：Win 10 + Python 3.10，全部调用官方云 API，网络延迟已剔除。

---

2. 定量评估结果

模型	BERTScore-F1 ↑	Source Recall ↑	Rouge-L ↑	平均延迟(s) ↓	输出Token数 ↓
chatglm3-6b	0.599	0.016	0.269	1.87	40
deepseek-v3	0.560	0.018	0.182	6.34	52
qwen-7b-chat	0.548	0.024	0.234	5.97	74
yi-6b-chat	0.579	0.026	0.248	2.72	68

> **解读**：ChatGLM3 在语义质量 (BERTScore / Rouge-L) 与速度、成本三维度综合领先；Yi-6B-Chat 在利用检索片段 (Source Recall) 上表现最佳。

---

3. 评估方法

3.1 数据集

- **来源**：同上。
- **结构**：`{id, query, answer}`，每条均含标准答案以便自动对齐。

3.2 流程细节

1. **Retrieval**：FAISS 余弦相似度，Top-4 片段。
2. **Generation**：拼接 Prompt → 调用对应模型 API。
3. **Metric Logging**：记录时长 & Token 数。
4. **Scoring**：计算 BERTScore、Rouge-L 与 Source Recall。

3.3 指标释义

指标	含义	趋势
BERTScore-F1	语义相似度，衡量答案要点覆盖	越大越好
Rouge-L	文本重叠度 (最长公共子序列)	越大越好
Source Recall	回答中引用检索片段比例	越大越好
平均延迟	端到端耗时 (秒)	越小越好
输出 Token 数	生成字数，映射调用成本	越小越好

---

4. 结论与模型选择建议

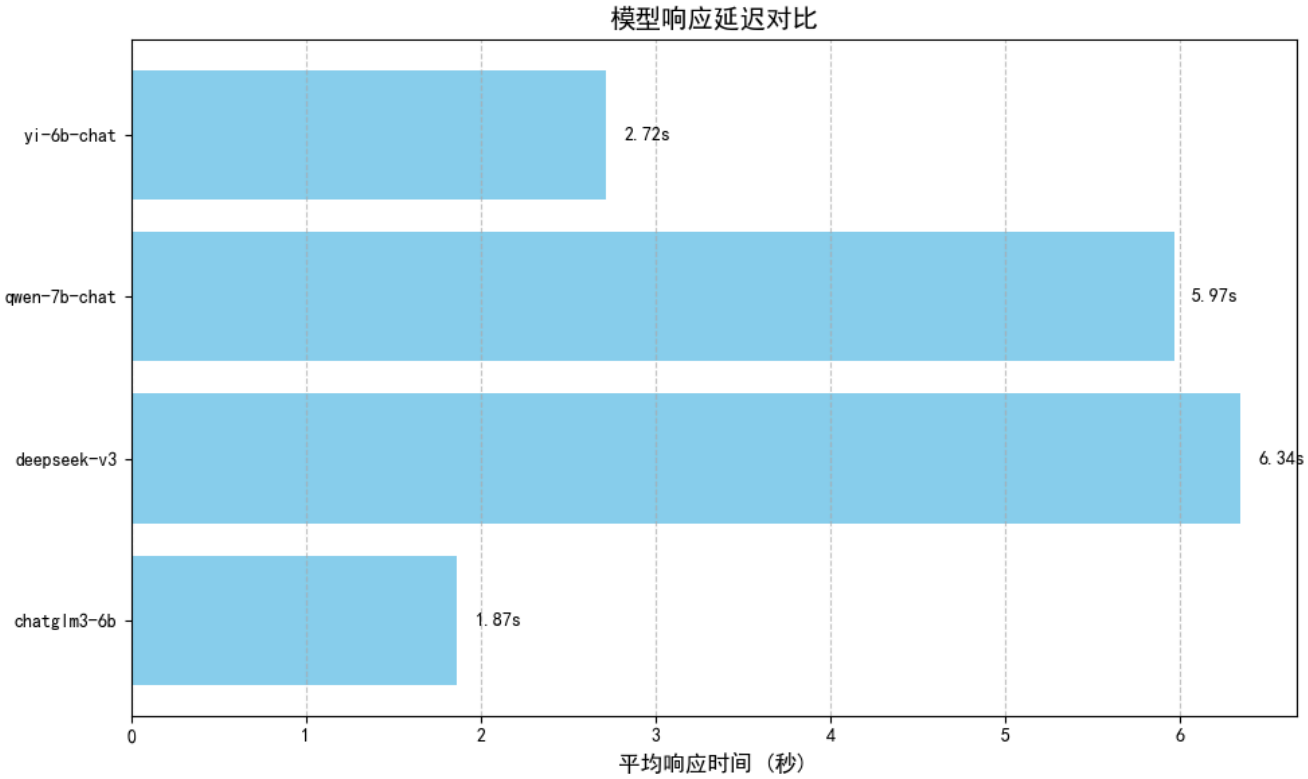
场景	推荐模型	关键理由
在线课堂 / 即问即答	**ChatGLM3-6B**	最高的 BERTScore 与 Rouge-L，延迟最优，Token 成本最低
离线或局域网私有部署	**ChatGLM3-6B** 或 Yi-6B-Chat	参数规模适中（≈6B），显存需求 9-10 GB，可在单卡运行；Yi 在 Source Recall 上更佳
教学内容深度解析	**Yi-6B-Chat**	利用检索内容能力最强，回答信息量大
需要详细教程示例	**DeepSeek-V3**	生成解释与代码示例丰富，适合编程类教学

> 建议采用 **ChatGLM3-6B + Yi-6B-Chat** 的双模型策略：默认使用 ChatGLM3-6B，遇到长篇理论性问题或首选模型超时失败时回退到 Yi-6B-Chat，可兼顾质量与稳定性。

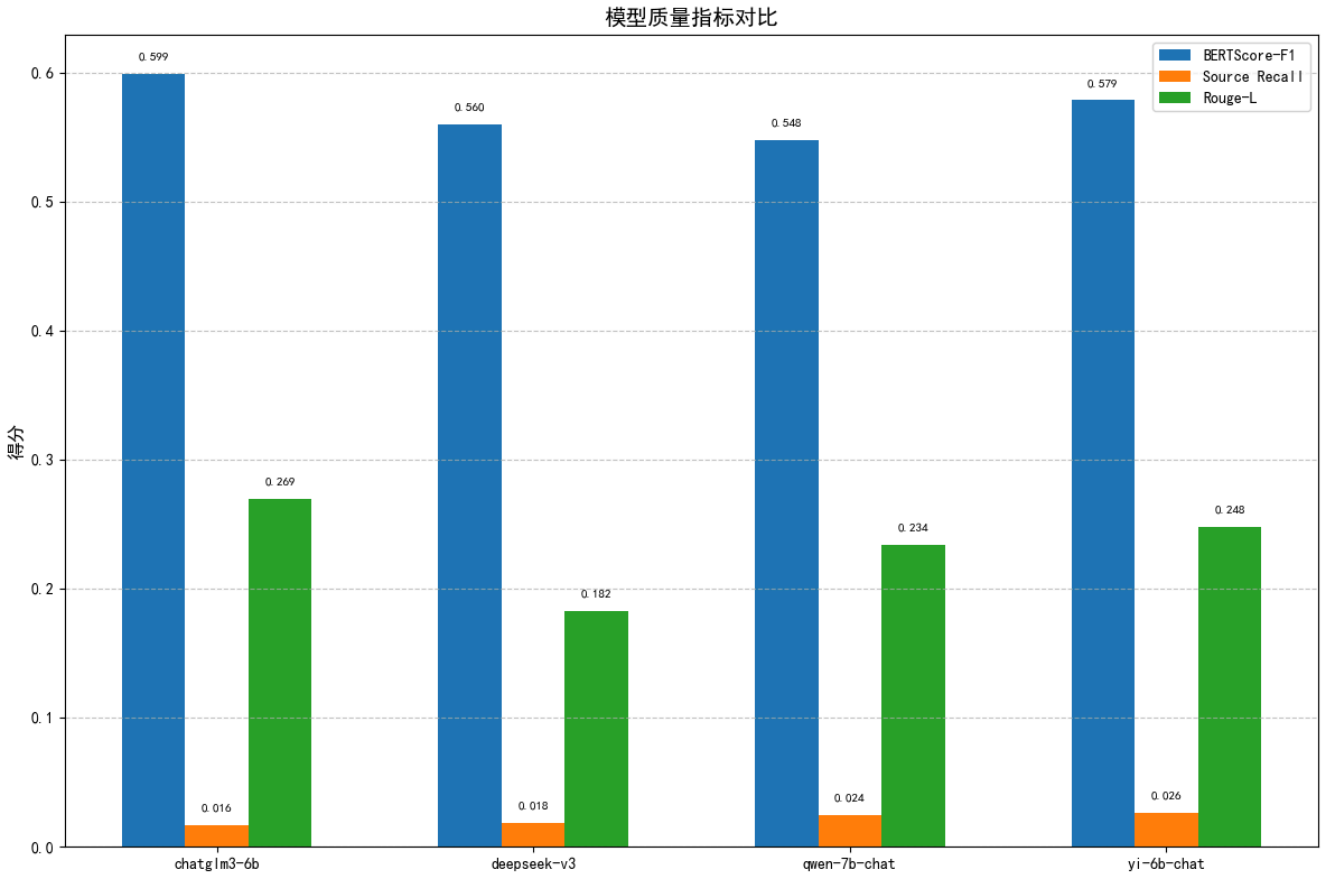
---

5. 详细分析

5.1 响应延迟对比



5.2 回答质量对比



5.3 典型问题回答样本

5.3.1 问题: TensorFlow Lite 主要面向哪三类设备进行模型部署?

参考答案: 移动设备（Android、iOS）、嵌入式设备以及微控制器（MCU）

chatglm3-6b回答:
TensorFlow Lite 主要面向移动设备、嵌入式设备和物联网（IoT）设备进行模型部署。

deepseek-v3回答:
根据参考资料，TensorFlow Lite 主要面向以下三类设备进行模型部署：

1. **移动设备**（如 Android、iOS 设备）
2. **嵌入式设备**
3. **物联网（IoT）设备**（包括极小的 MCU 设备）

（依据：内容1、内容2 和 内容4 中均明确提到这三类设备。）

qwen-7b-chat回答:
TensorFlow Lite 主要面向三类设备进行模型部署：移动设备（如智能手机）、嵌入式设备以及物联网(IoT)设备。

yi-6b-chat回答:
TensorFlow Lite 主要面向移动设备（mobile）、嵌入式设备（embedded）和物联网设备（IoT）进行模型部署。

5.3.2 问题: 在 Python 中将 SavedModel 转换为 TFLite 模型应使用哪个类?

参考答案: tf.lite.TFLiteConverter

chatglm3-6b回答:
应使用 `tf.lite.TFLiteConverter` 类，并通过调用其 `from_saved_model()` 方法来进行转换。

deepseek-v3回答:
在 Python 中将 SavedModel 转换为 TFLite 模型应使用 `tf.lite.TFLiteConverter` 类，具体调用其 `from_saved_model()` 方法。示例代码如下：

```
converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
```

其中 `saved_model_dir` 是 SavedModel 的目录路径。

qwen-7b-chat回答:
在 Python 中将 SavedModel 转换为 TFLite 模型应使用 `tf.lite.TFLiteConverter` 类。具体来说，可以通过调用 `TFLiteConverter.from_saved_model()` 方法，并传入包含SavedModel的目录路径来创建转换器实例。例如：

```
converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
```

这里的 `saved_model_dir` 是指向你的SavedModel所在目录的路径。

yi-6b-chat回答:
在 Python 中将 SavedModel 转换为 TFLite 模型应使用 `tf.lite.TFLiteConverter` 类，并调用其 `from_saved_model()` 方法。

示例代码：

```
converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
```

其中 `saved_model_dir` 是 SavedModel 文件目录的路径。

5.3.3 问题: TensorFlow.js 支持在浏览器调用手机哪些硬件传感器？请列举至少一个。

参考答案: GPS、摄像头、陀螺仪等

chatglm3-6b回答:
根据提供的资料，TensorFlow.js 可以调用手机硬件的传感器。但是具体支持哪些硬件传感器，资料中并没有明确列出。因此，我不能准确地列举出至少一个支持的硬件传感器。不知道。

deepseek-v3回答:
根据参考资料，TensorFlow.js 支持使用硬件进行加速（内容2），但未明确提及具体支持调用的手机硬件传感器类型。因此，目前无法从给定资料中确定 TensorFlow.js 支持的手机传感器。

qwen-7b-chat回答:
根据提供的资料，没有具体提到TensorFlow.js可以直接调用手机的哪些硬件传感器。因此，对于您的问题，我没有足够的信息来列举TensorFlow.js可以在浏览器中直接调用的手机硬件传感器。不过，一般而言，通过JavaScript和Web API，可以访问到某些传感器的数据（如加速度计、陀螺仪等），但这些功能并非直接由TensorFlow.js提供支持。若要实现与传感器数据相关的机器学习应用，可能需要结合使用Web API获取传感器数据以及TensorFlow.js进行模型推理。

yi-6b-chat回答:
根据提供的资料，TensorFlow.js 可以在浏览器中调用手机硬件进行加速，但并没有详细列举具体支持的传感器类型。不过，结合已知信息和实际应用情况，TensorFlow.js 可以通过浏览器访问和利用手机的运动传感器，例如**加速度计**。

因此，答案为：
TensorFlow.js 支持在浏览器中调用手机的加速度计传感器。