测试 wps 格式文件读取

1	1	3	3	4	
1	1	3	3	4	
1	1	3	3	4	
1	1	3	3	4	
1	1	3	3	4	6

机体

一款轻量 Agent 桌面端拘束器（本地/远端统一 OpenAI 兼容接口）

\[中文 | [English](README_en.md) \]

快速开始

1. 下载机体
   - https://pan.baidu.com/s/18NOUMjaJIZsV_Z0toOzGBg?pwd=body
   - 包括 eva 程序前端和 EVA_BACKEND 后端。
   - EVA_BACKEND 后端结构：`EVA_BACKEND/<架构>/<系统>/<设备>/<项目>/`（例如`EVA_BACKEND/x86_64/win/cuda/llama.cpp/llama-server.exe`）。也可以自己编译相应后端，按照这个结构摆放，机体运行时会自动识别。

2. 下载模型
   - https://pan.baidu.com/s/18NOUMjaJIZsV_Z0toOzGBg?pwd=body
   - EVA_MODELS 模型目录：

- `EVA_MODELS/llm`：大语言模型（主要），选一个小于你的设备内存/显存大小的模型
- `EVA_MODELS/multimodal`：多模态模型（非必要）
- `EVA_MODELS/embedding`：嵌入模型（非必要）
- `EVA_MODELS/speech2text`：Whisper 模型（非必要）
- `EVA_MODELS/text2speech`：tts.cpp 模型（非必要）
- `EVA_MODELS/text2image`：SD/Flux/Qwen-image/Qwen-image-edit/ 模型（非必要）
- 也可以前往 https://hf-mirror.com 搜索，机体支持几乎所有开源大语言模型

3. 启动
- windows 双击打开 eva.exe，linux 运行 eva.appimage（运行前赋予程序可执行权限 ）
- liunx 下需要给 eva 和后端可执行权限，进入 eva 的文件夹，终端运行 chmod -R a+x .
- 运行时确保机体程序与后端在同一目录

4. 装载！
- 点击装载按钮，选择一个 gguf 模型载入内存

5. 加速！
- 如果有显卡的话，点击设置按钮，为模型选择合适的推理设备（vulkan/cuda），并将 gpu 负载层数调到最大

6. 发送！
- 在输入区输入聊天内容，点击发送

功能
基础功能
<details>
<summary> 两种模式 </summary>

1. 本地模式：选择 gguf 模型后，启动本地的 `llama-server` 程序并默认开放端口 8080，也可以网页访问。

2. 链接模式：填写 `endpoint/key/model` 切换到远端模型，使用 OpenAI 兼容接口（`/v1/chat/completions`）。

</details>
<details>
<summary> 两种状态 </summary>

1. 对话状态
- 在输入区输入聊天内容，模型进行回复
- 可以使用挂载的工具
- 可以按 f1 截图，按 f2 进行录音，截图和录音会发送给多模态或 whisper 模型进行相应处理

2. 补完状态
   - 在输出区键入任意文字，模型对其进行补完
</details>
<details>
<summary> 六个工具 </summary>

```txt
   在系统指令中附加“工具协议”，指导模型以 <tool_call>JSON</tool_call> 发起调用；
   推理结束后自动解析工具请求，执行并把结果以 "tool_response: ..." 继续发送，直至没有新请求。
```

1. 计算器
   - 模型输出计算公式给计算器工具，工具将返回计算结果
   - 例如：计算 888*999
   - 调用难度：⭐

2. 鼠标键盘
   - 模型输出行动序列来控制用户的鼠标和键盘，需要模型拥有视觉才能完成定位
   - 例如：帮我自动在冒险岛里搬砖
   - 调用难度：⭐⭐⭐⭐⭐

3. 软件工程师
   - 类似 Cline 的自动化工具执行链（execute_command/read_file/write_file/replace_in_file/edit_in_file/list_files/search_content/MCP…）。
   - 例如：帮我构建一个 cmake qt 的初始项目
   - 调用难度：⭐⭐⭐⭐⭐

4. 知识库
   - 模型输出查询文本给知识库工具，工具将返回三条最相关的已嵌入知识
   - 要求：先在“增殖-知识库”上传文本并构建（启动嵌入服务 → /v1/embeddings 逐段入库）。
   - 例如：请问机体有哪些功能？
   - 调用难度：⭐⭐⭐

5. 文生图
   - 模型输出绘画提示词给文生图工具，工具将返回绘制好的图像
   - 要求：用户需要先在增殖窗口配置文生图的模型路径，支持 sd 和 flux 模型
   - 例如：画一个女孩
   - 调用难度：⭐⭐

6. MCP 工具
   - 通过 MCP 服务，获取到外部丰富的工具

- 说明：挂载工具后需要前往增殖窗口配置 MCP 服务
- 调用难度：★ ★ ★ ★ ★
</details>
<details>
<summary> 任意技能 </summary>

技能为机体在约定框架下引入的可插拔能力包。通过该机制，指挥员可在不改动主程序的情况下，为驾驶员注入特定场景的工作流程、模板与脚本指令。
具体可参考 EVA_SIKLLS 目录下的示例技能。

- **规范约束**：挂载软件工程师工具后，将技能文件夹压缩成 zip 文件，可拖入技能区进行导入。每个技能都必须提供 `SKILL.md`，文件以 YAML frontmatter 描述 `name`、`description`、`license` 等元数据，正文给出操作步骤、输入输出格式、注意事项。EVA 在解析时严格依赖这些字段，缺失将直接拒绝导入。
</details>

增强功能
<details>
<summary> 视觉 </summary>

- 介绍：在 本地模式 + 对话状态 下可以挂载视觉模型，视觉模型一般名称中带有 mmproj，并且只和特定的模型相匹配。挂载成功后用户可以选择图像进行预解码，来作为模型的上文

- 激活方法：在设置中右击“挂载视觉”选择 mmproj；拖拽/右击上传/按 F1 截图后，点击“发送”进行预解码，再进行问答。
</details>
<details>

<summary> 听觉 </summary>

- 介绍：借助 whisper.cpp 项目将用户的声音转为文本，也可以直接传入音频转为字幕文件
- 激活方法：右击状态区打开“增殖-声转文”，选择 whisper 模型路径；回到主界面按 F2 开始/结束录音，结束后自动转写回填输入框。
</details>
<details>

<summary> 语音 </summary>

- 介绍：借助 windows 系统的语音功能将模型输出的文本转为语音并自动播放，或者可以自己配置 tts.cpp 模型进行文转声
- 激活方法：右击状态区打开“增殖-文转声”，选择系统语音或加载 tts.cpp 模型并启动。
</details>

辅助功能

<details>
<summary> 模型量化 </summary>

- 可以右击状态区弹出增殖窗口，在模型量化选项卡中对未经量化的 fp32、fp16、bf16 的 gguf 模型进行量化

</details>
<details>
<summary> 自动监视 </summary>

- 本地对话状态下，挂载视觉后，可设置监视帧率；随后会自动附带最近 1 分钟的屏幕帧到下一次发送。
</details>

源码编译

<details>

<summary> 展开 </summary>

1. 配置环境
   - 安装编译器 windows 可以用 msvc 或 mingw，linux 需要 g++或 clang
   - 安装 Qt5.15 库 https://download.qt.io/
   - 安装 cmake https://cmake.org/

2. 克隆源代码

   ```bash
   git clone https://github.com/ylsdamxssjxxdd/eva.git
   ```

3. 编译
   ```bash
   cd eva
   cmake -B build -DBODY_PACK=OFF
   cmake --build build --config Release -j 8
   ```
   - BODY_PACK：是否需要打包的标志，若开启，windows 下将所有组件放置在 bin 目录下；linux 下将所有组件打包为一个 AppImage 文件，但是依赖 linuxdeploy 等工具需要自行配置

4. 后端准备
   - 从上游或第三方获取已编译的推理程序。
      - 也可以在 nerv 项目中获取所有三方源码自行编译 git clone

https://github.com/ylsdamxssjxxdd/nerv.git

- 按中央教条放置第三方程序：`EVA_BACKEND/<架构>/<系统>/<设备>/<项目>/`，例如：
- `EVA_BACKEND/x86_64/win/cuda/llama.cpp/llama-server(.exe)`
- 架构：`x86_64`、`x86_32`、`arm64`、`arm32`
- 系统：`win`、`linux`
- 设备：`cpu`、`cuda`、`vulkan`、`opencl`（可自定义扩展）
- 项目：如 `llama.cpp`、`whisper.cpp`、`stable-diffusion.cpp`
- 运行时 EVA 仅在本机同架构目录下枚举设备并查找可执行文件，并自动补全库搜索路径（Windows: PATH；Linux: LD_LIBRARY_PATH）。

5. 打包分发（解压即用）

- 将可执行（build/bin/eva[.exe]）、同级目录 `EVA_BACKEND/`、必要 thirdparty 与资源、以及可选 `EVA_MODELS/` 一并打包；
- 目录示例：
  - `EVA_BACKEND/<arch>/<os>/<device>/llama.cpp/llama-server(.exe)`
  - `EVA_BACKEND/<arch>/<os>/<device>/whisper.cpp/whisper-cli(.exe)`
  - `EVA_BACKEND/<arch>/<os>/<device>/llama-tts/llama-tts(.exe)`
  - `EVA_MODELS/{llm,embedding,speech2text,text2speech,text2image}/...`
- 程序首次启动会在同级目录创建 `EVA_TEMP/`，用于保存配置、历史与中间产物。

</details>

测试

- 在构建目录启用测试：`cmake -B build -S . -DEVA_ENABLE_FUNCTIONAL_TESTS=ON`
- 运行单元测试：`cmake --build build --target xtool_tests xtool_harness_tests && ctest -L unit`
- 运行功能测试（需要可执行文件与场景配置）：`ctest -L functional`
- 场景文件位于 `tests/functional/scenarios`，每个 JSON 描述 command/timeout/期望输出，可指向本地或链接模式脚本。
- 生成覆盖率报告：`cmake -B build -S . -DEVA_ENABLE_COVERAGE=ON && cmake --build build && ctest -L unit && cmake --build build --target coverage`（需要提前安装 `gcovr`，报告输出在 `build/reports/coverage/index.html` / `build/reports/coverage/coverage.xml`）。

致谢

- [llama.cpp](https://github.com/ggerganov/llama.cpp)
- [whisper.cpp](https://github.com/ggerganov/whisper.cpp)
- [stable-diffusion.cpp](https://github.com/leejet/stable-diffusion.cpp)
- [TTS.cpp](https://github.com/mmwillet/TTS.cpp)
- [Qt 5.15](https://www.qt.io/)