M2.1评测swe/多语言方向brief-1223

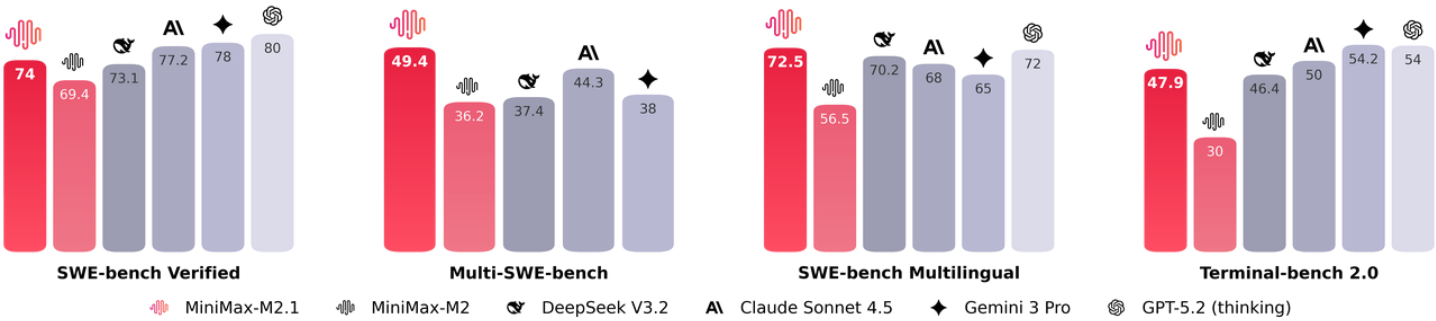
关于M2.1的介绍

M2.1此次的核心更新点在于**【多语言优化——更多工作岗位更多编程语言的可用性】，需要有体现这次更新亮点的case和阐述**

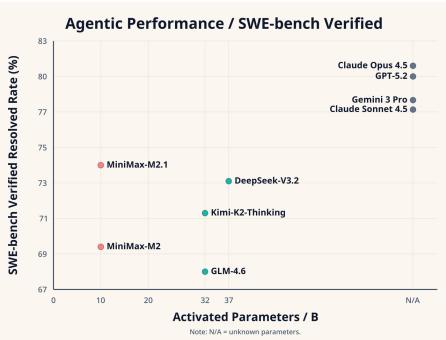
*__痛点及洞察__：大部分的模型在前端和脚本上有比较深度的优化，但是在后端的语言Jave和Golang上会下降，在更底层的C++等语言上最低。由于客户端的环境比前端更复杂，我们发现大部分模型写客户端的能力也显著低于写前端。为了让AI能更广泛的用好，我们需要系统的提升多种工作岗位多种编程语言的可用性

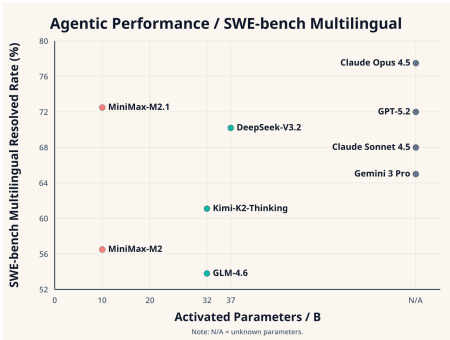
更新1：全新文本模型M2.1，核心亮点如下（12.23上午11点前会做最后修正）

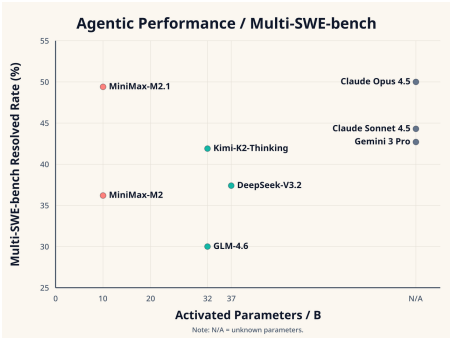
- **多编程语言超越SOTA** - 过去很多模型主要关注 Python，但真实世界的项目大家会用不同语言类解决问题。本次 M2.1 在 Rust/Java/Golang/C++/Kotlin/OC/TS/JS 上都有显著提升。在另外我们特别注重了web3 protocol的覆盖，对于web3类型项目表现优越。Multi-SWE-Bench at 49.4%, exceeding Anthropic Sonnet 4.5 and Gemini 3 Pro。代码审阅 （code comprehension） - 包括代码性能优化等



在软件工程相关场景的核心榜单上，MiniMax-M2.1相比于M2有了显著的提升，尤其是在多语言场景上，超过 Claude Sonnet 4.5并接近Claude Opus 4.5。在相同激活参数下，MiniMax-M2.1 在显著优于其他模型，达到更优的效率-效果兑换。





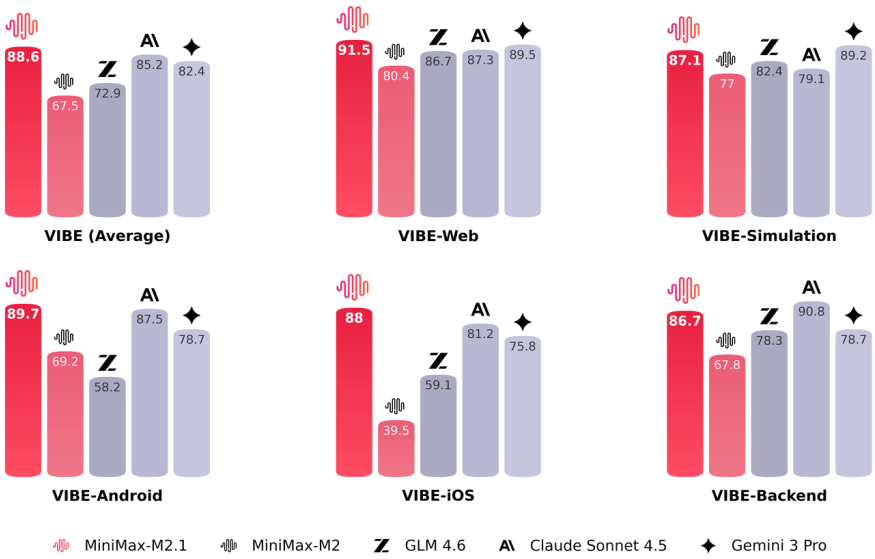


- **AppDev 显著优化 -** 我们培养了模型更好的Web美学品味，并实现了更逼真的科学场景模拟。针对业界普遍的移动端短板，极大增强了原生 Android/iOS 开发能力。Not only vibe webdev, but also vibe appdev.
- **更简洁高效的回复** - 相比 MiniMax M2，MiniMax-M2.1 的模型回复以及思维链更加简洁，体感编程以及响应速度有极大提升。
- **符合指令约束提升，办公场景变为可能** - 作为开源模型中第一个实现先进interleaved thinking的模型系列，M2.1 systematic problem-solving 能力再次升级。模型不仅关注代码执行是否正确，同时关注模型对“复合指令约束”的整合执行能力（具体见 OctoCodingBench）。这部分在一些办公场景可以初步使用（见下面 showcase 和 Toolathlon）
- **脚手架泛化能力加强** - M2.1加强了脚手架泛化能力，在各类编程工具/Agent 上都有出色的表现，如 Claude Code, Droid (Factory AI), Cline, Kilo Code, Roo Code, etc。M2.1出色支持各类Context Management，如Skill.md, Claude.md/agent.md/cursorrule, SlashCommand, etc。
- **最轻量的SOTA模型** - 虽然M2.1在短短2个月内相对于M2做到了显著性能和实用性的提升，但M2.1依旧继承了M2的轻量 模型10B activated；开源模型里面成本最低的SOTA performance模型

更新2：开源评测集VIBE

全新基准 VIBE (Visual & Interactive Benchmark for Execution)，涵盖了 Web、仿真 (Simulation)、Android、iOS 及后端 (Backend) 五大核心子集。不同于传统基准，通过创新的 Agent-as-a-Verifier (AaaV) 范式，VIBE 能够自动评估生成的 Application 在真实运行环境中的交互逻辑与视觉美感

- MiniMax-M2.1 在 VIBE 综合榜单中表现卓越，以平均 88.6 分的成绩展现了接近Claude Opus 4.5的全栈构建能力，并在几乎所有子集上都显著优于Claude Sonnet 4.5



***官方demo集**：M2.1 Showcase（中文版对外）

- 可参考demo集中的评测方向，重点测试能够体现多编程语言的case（尤其是后端）

- 其中大部分会放在官宣blog中使用，媒体可酌情采用，但不要全部放官方case，需要有至少70%以上的原创case

关于撰稿的Tips

以下内容向博主解读**“M2.1模型价值怎么被看见”，包含**

1. 上面的摘要到底在说什么
2. 这个更新点真正“新”在哪里
3. 做测评时，怎么做出适合的价值更大的case

我们希望看到什么？

内容：真实/大胆/有观点有见解的测评（好的与不好的点），言之有物+言之有理

情绪：戳中大家的兴趣点和好奇心，以及体验m2.1的冲动——看完文章观众想要跃跃欲试

标题：一个很好的很有力很精准的标题

【机构媒体/资讯号】请注意——需要适当下调测评篇幅，额外增加以下内容

1. 解读m2.1: 之前发布了m2口碑不错，m2.1的发布是不是技术路径选择的延续（参考上面的多语言的定调）
2. 招股书引发的技术投入&成绩：m2.1跟ipo招股书的关联，坚持做技术投入、模型研发，在薄弱板块也在弯道超车，多模态领域都取得了一定的成绩
3. 文章内容：有一部分内容是资讯、模型做评测demo（自己发挥）、当下的环境下给出媒体的观点（自己发挥）

我们不希望看到什么？

泛泛的测评、强行跳跃总结优势、看完文章观众会默认这是一篇草草体验的商单任务

ps我们强烈推荐大家进群，里面有我们最懂m2.1的工程师和400+一同正在内测m2.1的开发者们

如何理解“多语言优化”，并且找对测评方向、并深入阐释它的意义

一、上面的摘要到底在说什么

> 【多语言优化——更多工作岗位更多编程语言的可用性】

本次发布的M2.1，不再只擅长“前端 demo 和脚本”，而是**开始真的能帮不同岗位的人写他们每天要用的代码**。

背景痛点，本质只有一个现在大多数模型其实“偏科”很严重。

写前端 JS / TS：流畅、好看、像样；

写 Python 脚本：能跑、能用；

但一到：Java 后端/Golang 服务/C++ / 客户端 / SDK

就开始：结构不对/工程习惯不对/接口、内存、并发写错

所以不是语言语法问题，是“岗位上下文不对”。

所以我们需要的不是多会几门语法，而是 **开始理解：不同岗位写代码，到底在解决什么问题**。

这是一个 **SWE 级别的优化，而不是“支持更多语言”这种表面升级**。

二、这次更新的“真正亮点”该怎么讲?

❌ **不要这样讲（太空太冷冰冰没感情没逻辑）**

- 支持更多语言
- 多语言能力增强
- 后端语言效果提升

✅ **推荐思考逻辑**

从「岗位 → 任务 → 语言」反推模型能力，从最可见可理解可接触的各种工种解析他们的常见工作任务，再down到最基础最本质的语言层

本次M2.1重点优化swe/多语言的价值⬇️：

1. 从“写得像” → “写得对”

其他/过往的模型：

- Java 能写，但不像 Java 工程
- Go 能跑，但不像 Go 服务

M2.1（*下面just举例子，实际以测评体验为准*）：

- 包结构、接口设计、并发模型更贴近真实工程
- 更少“前端思维强行套后端”

2. 从“单文件” → “工程感”

尤其在：Java/Go/ C++ / 客户端

M2.1能（*just举例子，实际以测评体验为准*）：

- 拆模块
- 识别职责边界
- 避免把 demo 写法当生产代码

3. 从“前端友好” → “客户端 / 后端友好”

客户端环境比前端复杂，历史包袱重、系统差异大。

m2.1能（*下面just举例子，实际以测评体验为准*）：

- 更谨慎地使用 API
- 避免不现实的依赖假设
- 写出更接近真实客户端工程的代码

三、做测评时，如何表达？怎么做出适合的价值更大的case？

1. 测评方向/case推荐

找“我以前用不了，现在能用了”的瞬间，比如：

A. 岗位型测评任务

示例 1：后端 / Go

> 用这个模型，写一个 **Golang 的限流中间件**，要求：

- 支持并发
- 使用 channel / context
- 有清晰的接口定义

👉 对比点：

- 是否还是 JS 式思维
- 是否理解 Go 的并发模型

示例 2：Java 后端

> 用模型实现一个 **Spring Boot 的用户鉴权模块**：

- Controller / Service / Repository 分层
- 合理使用注解
- 不写前端式工具函数

👉 对比点：

- 工程结构
- 注解使用是否“像 Java 人写的”

示例 3：客户端 / C++

> 让模型写一个 **跨平台 C++ 文件读写工具类**：

- 考虑异常
- 不滥用 STL
- 接口清晰

👉 对比点：

- 是否避免“算法竞赛式 C++”
- 是否有工程克制感

B. 跨语言对同一任务

这个非常安利大家去写（虽然比较考验博主自己的多语言能力）

> 同一个需求：

- 用 JS 写
- 用 Go 写
- 用 C++ 写

看模型：

- 有没有换思维方式
- 还是一套模板打天下

2. 好的吸睛的有活人感的标题方向/风格（真的很重要）

我们强烈建议大家撰写吸睛、有重点、有话题、一眼就懂m2.1的亮点的文章标题

- ❌「XX 模型多语言能力评测」（但如果通篇非常solid技术也可以这么写，主要还是看博主自己过往的风格）
- ✅「这个模型终于不像前端写 Go 了」
- ✅「我第一次敢让 AI 写 Java 后端代码」
- ✅「AI 写 C++，这次居然有工程感了」

最后，关于swe和多语言，我们希望大家展示MiniMax-M2.1是否能真正理解不同岗位的工程习惯，而不仅是语言语法。欢迎大家用自己最熟的后端 / 客户端语言，去“为难”M2.1。