Web3 x AI 系列 | 为什么人工智能代理需要经济基础设施

人工智能是目前最热门的话题，持续吸引着资本、人才和关注，但Web3仍然被上一轮的失败所束缚。人们对加密货币感到厌倦是有一定道理的，尤其是在多年来基础设施建设几乎从未得到实际应用之后。但这种比较过于局限于当前的讨论，忽略了人工智能一旦超越聊天界面就会出现的问题。

聊天机器人回答产品内部的提示，而智能代理则可以请求计算资源、调用API、购买数据集、将任务分配给其他模型、检查结果、交付成果，并在用户离开后继续运行。

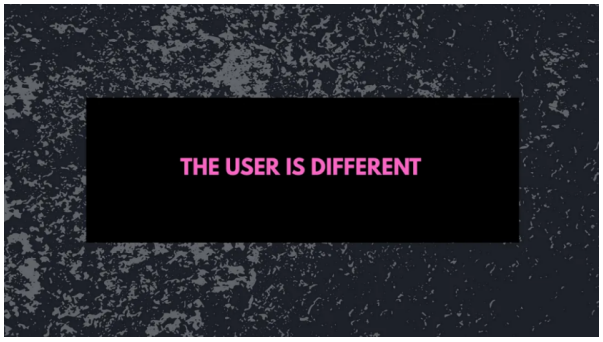
平台可以暂时将这些活动隐藏在一个账户背后，内部计量使用量，并在月底将所有费用计入用户的信用卡。当智能代理在受控环境中运行，平台承担所有后续工作时，这种方式就足够了。

但一旦智能代理跨服务运行，这种机制就会失效，因为经济活动会带来模型质量本身无法解答的问题。哪个账户在执行操作？谁授予了它权限？它可以消费什么？任务执行过程中发生了什么？输出结果被使用后谁获得报酬？其他系统可以依赖哪个记录？

人工智能赋予软件更强的行动能力，但只有当行动能够跨系统进行授权、支付、记录、核查和结算时，才能真正发挥商业价值。Web3 的优势在于它能够比平台账户、订阅或私有数据库更好地处理这些环节。但这并不意味着区块链应该应用于所有人工智能工作流程，而是意味着自主软件开始暴露围绕人类用户、平台账户和计费系统构建的基础设施的局限性——这些系统假定仍然有人能够批准下一步操作。

大多数代理不需要自己的新代币，许多代理仍将留在封闭的产品中，许多人工智能工作流程使用普通数据库、普通账户和普通计费系统会更加高效。真正的挑战在于代理需要支出资金、接收资金、购买资源、雇佣其他代理，以及向受控环境之外的交易对手证明工作成果。

到那时，代理人需要经济基础设施，软件可以直接运行，规则可以被其他系统读取，而不是信任隐藏在一家公司的后端。



大多数互联网基础设施仍然假定用户就在附近，需要用户注册邮箱地址、添加支付方式、接受条款、点击界面、等待确认，并在产品需要时创建新账户。这种模式虽然混乱，但人们已经习惯了。他们可以阅读警告信息、重置密码、批准付款、联系客服，并理解一个半成品的结账流程。

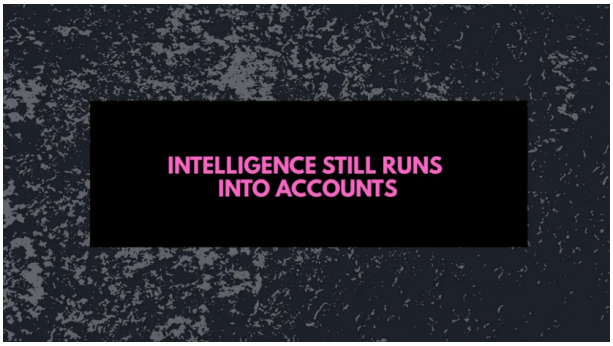
代理商则对整个系统的不同部分提出了更高的要求，因为他们不需要为了界面而界面，也不关心按钮是否友好。他们需要的是可以阅读的规则、可以执行的权限、可以查询的余额、可以调用的服务，以及无需用户坐在屏幕前也能正常运行的支付系统。代理商并不要求互联网操作简单，他们需要的是互联网以软件可以执行的方式呈现规则、余额、限额和承诺。

私钥、签名、gas费、智能合约调用、多重签名逻辑和可编程账户对普通用户来说极其繁琐，主要是因为它们暴露了太多底层机制。而代理程序则更接近这些机制运行，它们可以签署交易、监控余额、检查合约状态、重试失败的调用，并执行条件逻辑，而无需将每一步都转化为用户可感知的用户体验。

对于只想使用应用程序的用户来说，钱包功能过于复杂；但对于需要在限额内支出的软件而言，策略控制的账户可以成为一个有效的操作边界。

对于只想一键完成操作的用户来说，gas费令人烦恼；但赞助gas费或账户抽象可以将交易成本转化为后台策略。

智能合约对消费者来说界面笨拙，但对于只关心规则是否执行的代理程序来说，它可以作为一个服务端点。



人工智能领域最容易犯的错误是将智能本身视为问题的全部。一旦模型能够编写代码、分析文档、生成设计或规划工作流程，人们就会觉得最难的部分已经解决。然而，模型一旦离开沙盒环境，那些常见的经济问题又会重新出现。

即使智能体找到了合适的数据集，仍然需要访问条款和支付方式。如果它选择计算服务提供商，就需要制定并执行预算。如果它将工作交给专业的评估人员，就需要资金管理、支付款项并保留支票记录。如果输出结果之后被重复使用，系统还需要归属机制和收益分配机制，否则工作成果就会与其产生的输入、支付和支票脱节。

平台可以通过以下方式在自身内部解决部分问题：为智能体提供内部账户、在数据库中计量使用情况、按月向客户收费，并通过支持服务解决纠纷。当代理仅限于单个产品时，这种方法行之有效；但当代理跨越信任边界时，其效力就会减弱，因为每个外部服务都有其自身的账户系统、计费方式、权限模型以及私密的事件记录。

API 密钥是为软件访问而设计的，但它们并非旨在将软件转化为可问责的经济主体。支付系统可以向个人或公司收费，但无法赋予小型代理对单次任务付款的完全自主权。SaaS 计费模式适用于订阅服务，但对于那些可能只想为评估、数据提取、推理调用或从其他代理处获得的特定工作支付几美分的代理来说，这种模式并不适用。

钱包改变了问题的性质，因为它为代理提供了一个可寻址的账户，该账户可以承载价值并与跨服务的合约进行交互。

Uptick 的 AA Wallet 就是一个很好的例子，它将账户推向了更接近策略层的位置，账户价值与软件的支出、可调用的服务以及何时需要其他审批层介入挂钩。该钱包并没有让模型更智能，也没有消除产品设计的必要性，但它为代理商提供了一种在单一公司数据库之外承担责任的方式。

即使人工在聊天窗口中阅读，人工智能的输出结果也很难验证，而一旦代理开始在不同系统间传递工作，验证的压力就会更大。结果可能看起来合情合理，但下一个代理需要的不仅仅是答案，它还需要知道使用了哪些数据、调用了哪些工具、哪个评估人员检查了输出结果、是否已支付款项，以及工作流程的任何部分是否失败、更改或事后获得批准。

中心化平台可以提供自身产品的日志，但这些日志通常止步于平台边缘。一旦工作在代理、工具、数据提供商、计算提供商和人工审核人员之间流转，记录就会被分割成私有的部分。每个参与者都可以证明自己的部分，但除非某个平台拥有相关的账户、支付、权限和历史记录，否则整个工作流程将难以重构。

Web3 的作用比为代理的所有操作提供完整的审计跟踪要窄。某些执行过程应该保持私密，某些数据应该链下存储，某些检查仍然依赖于外部系统。更强大的应用场景是共享选定操作、付款、权限和证明的记录，不同的系统可以读取这些记录，而无需依赖单一公司来保存完整的历史记录。

一旦输出结果包含付费数据、租用计算资源、外部评估和人工审核等因素，所有权的认定就变得更加复杂。如果没有可移植的记录，每个下游用户都只能依赖平台内部对事件的描述。有了更强大的溯源层，归属、许可、收入分配和争议解决将更容易评估，因为作品本身承载了更多历史记录。

一款优秀的 Web3 x AI 产品在移除区块链后性能应该有所下降。如果区块链层主要增加成本、拖慢工作流程，或者创建了类似代币的融资渠道，那么该产品只是把 Web3 当作噱头。区块链必须能够处理一些系统难以替代的功能，例如其他服务可以识别的账户、代理商无法绕过的支出规则、双方都能验证的支付条件，或者工作流程离开公司后端后仍然有效的记录。

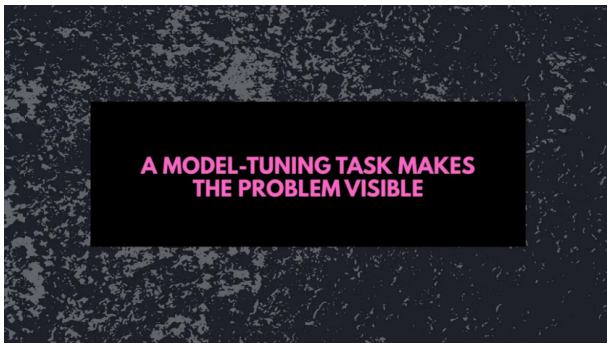
读取钱包数据的聊天机器人或许很方便，但这并不能说明代理商的基础设施状况。与 AI 产品绑定的代币或许能够创造市场，但这并不能证明该代币能够协调任何事务。在社交媒体上发帖的机器人或许有助于分发，但发帖和回复并不会改变工作的支付、审核或记录方式。

真正的考验始于工作流程内部。

当代理商购买服务时，钱包会为其分配一个其他系统可以访问的账户。当从该账户支出时，智能账户可以限制金额、已批准的服务以及需要额外审批层的操作。当任务依赖于另一方时，支付条件可以冻结资金，直到工作完成。当输出流向下一个系统时，记录会提供足够的历史信息，以便下一个系统在信任之前判断发生了什么。

当用户只想要一个感觉正常的应用程序时，开放式结算、可编程账户、可移植历史记录和可组合合约就很难被接受。代理商并不要求普通的消费者流程，他们需要的是机器可读的承诺和执行路径，而区块链只有在提供这些路径的同时，又不强制所有参与者使用同一平台时，才能发挥作用。

内部客户支持人员不需要链上结算，财务团队内部的报告人员最好还是留在公司现有的技术栈中。当工作流程跨越多个组织、资金按操作流动、任务从一个代理商传递到另一个代理商，或者需要在原始平台不再参与后仍然保留记录时，Web3 才有资格被实施。



一个简单的模型调优任务就足以暴露协调问题。

用户请求代理针对特定用例提升模型性能，代理需要识别性能差距，请求或购买数据，租用计算资源，运行训练或微调，将结果发送给评估人员，针对特殊情况引入人工审核，交付更新后的模型，并将款项支付给所有参与其中的人员。

这些操作并非天方夜谭，但难点在于协调，因为在传统的平台架构中，每个步骤都依赖于不同的账户。计算资源提供商使用一套计费系统，数据提供商使用另一套，评估人员使用另一套，而人工审核人员的报酬则完全来自其他渠道。代理可以调用 API，但经济往来却分散在私有的仪表盘、发票、使用日志和支持邮件中。

代理原生架构将从智能账户和既定策略开始。代理可以设定消费限额，使用已获批准的提供商，超过阈值时需要申请批准，并通过双方都能读取的支付通道进行支付。支付方式需要转变，因为当代理商的工作分解成涉及多个服务的小操作时，他们并不适合订阅、发票和信用卡表单等支付方式。他们可能需要按请求、按推理、按评估、按数据提取或按下游使用付费。

如果服务流程围绕 Uptick 微支付服务构建，那么重复的小额请求就不必变成单独的结账环节。一次授权即可涵盖在批准限额内的多笔小额支付，验证可以在服务使用过程中进行，结算可以稍后批量处理，而不是强制每个操作都进入单独的链上交易。工作交付后，支付记录、输入、检查和发布条件可以随输出一起迁移，而不是被困在每个提供商的私有系统中。

将所有细节都放在公共账本上是错误的，因为大多数代理商的工作流程都需要私密执行、选择性披露、数据加密、链下存储以及防止敏感信息被公开的策略。其有用之处在于共享的经济框架，参与者可以就谁执行了操作、支付了什么、资金释放的条件以及哪个账户拥有支出权限达成一致。

账户抽象使该框架更加安全，因为代理的操作权限不必等同于账户所有者的全部权限。智能账户可以设置限额、批量操作、支付 gas 费、对异常行为要求额外批准、轮换密钥以及在故障后恢复访问权限。对于用户而言，这些功能使钱包的用户体验更加友好；而对于代理而言，这些功能定义了其实际可以执行的操作策略。

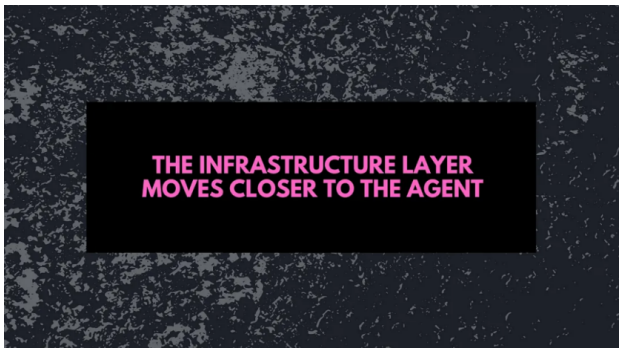
Web3 x AI 的弱化版本在系统尚未具备任何协调能力之前就急于使用代币。一旦代理开始出现，这种本能反应就更容易被合理化，因为自主软件听起来似乎应该拥有自己的资金、市场和治理体系。但更难的问题是，代币在工作流程中究竟发挥着什么作用。

当代币支持共享基础设施的访问、结算、验证、收益分配或治理时，它就具有意义。计算

提供商可能需要基于使用量的报酬，数据提供商可能需要在代理使用其工作成果时获得报酬，评估人员可能需要激励措施来诚实地检查输出结果，工具开发者可能需要一种方式，让代理在不同产品中调用他们的服务时获得收益。

测试方法很简单。

如果代币消失后，工作流程仍然运行得更好、更便宜、更顺畅，那么它可能只是一个噱头。如果移除代币会破坏访问、结算、验证或共享所有权，那么它实际上可能是基础设施的一部分，而不仅仅是附加在它上面的市场概念。



第一波人工智能价值的涌现集中在技术栈中最容易被感知的部分。模型需要电力、数据中心、芯片和工程人才，因此最接近这些资源限制的公司抓住了市场需求。这一层仍然很重要，但随着模型访问变得更加便捷，价值不再局限于此。

随着技术栈的成熟，瓶颈向上转移。更棘手的问题不再仅仅是谁能训练模型或提供GPU，而是代理如何访问服务、支付资源费用、证明工作成果、分配价值、存储记录以及与平台之外的系统进行协调。

正是在这个时候，真正有用的基础设施开始更靠近代理本身。

代理需要一个围绕模型运行的环境，其中包含工具、内存、账户、权限、支付、策略、数据访问、评估和结算等功能。其中一些功能仍将保持集中式，因为集中式基础设施通常更便宜、更易于控制。当代理需要与不共享同一数据库的参与者协调时，开放的支付通道（Open Rails）就变得至关重要。因为如果工作流程需要在多个服务之间流动，账户、支付和记录层就不能完全位于同一个后端。

随着代理功能日益强大，平台控制的价值也随之提升。如果一家公司拥有代理界面、账户系统、支付通道、任务历史记录和分发渠道，它就可以决定代理可以使用哪些服务、哪些贡献者可以获得报酬、哪些记录可见以及哪些外部系统可以参与其中。这或许效率很高，但也重新构建了Web3一直声称要消除的平台依赖性，只不过现在这种依赖性存在于可以代表用户执行操作的软件之下。

Web3不必成为人工智能的大脑，它必须在代理需要钱包、可编程支出规则、可验证的任务记录、托管、收入路由和跨服务结算等功能时发挥作用。这本身就足以完成大量工作，而且也让讨论聚焦于真正的瓶颈，而不是泛泛地谈论互联网的未来。

下一代实用的 Web3 x AI 产品乍看之下或许平平无奇，但它们将使代理更容易管理账户余额、支付 API 调用费用、证明任务已通过审核、在贡献者之间分配收益，或在不同服务之间共享信誉记录。Uptick 的 AA Wallet、AI

Agent Wallet 和兼容 x402 的微支付服务就属于这一操作层。在这一层，代理在执行操作前需要设置账户限额，支付需要随请求转移，并且记录在工作流程离开后端后仍可读取。

只能通过单一平台账户进行操作的代理在该平台内功能强大，但在其他平台则显得笨拙。一旦需要购买服务、证明工作、路由支付或跨系统共享历史记录，后端无法独自处理所有权限、争议和记录。

这正是市场比较所忽略的部分。一旦代理跨服务工作，账户、支付和事件记录就需要随工作转移，否则代理的效用仍然仅限于其所依赖的平台账户。

Uptick Network

hello@uptickproject.com

@Uptickproject

@Uptickproject

Uptick Network

Uptick Network