Navigating the Deployment Dilemma and Innovation Paradox: Open-Source versus Closed-Source Models

Yanxuan Wu
wyxsjtuer@gmail.com
Artificial Intelligence Research Institute,
Shenzhen MSU-BIT University
Shenzhen, China

Haihan Duan[‡]
duanhaihan@smbu.edu.cn
Artificial Intelligence Research Institute,
Shenzhen MSU-BIT University
Shenzhen, China

Xitong Li[*†]
xli@smbu.edu.cn
Artificial Intelligence Research Institute,
Shenzhen MSU-BIT University
Shenzhen, China

Xiping Hu[*‡§]
huxp@smbu.edu.cn
Artificial Intelligence Research Institute,
Shenzhen MSU-BIT University
Shenzhen, China

Abstract

Recent advances in Artificial Intelligence (AI) have introduced a popular paradigm in Machine Learning (ML) model development: pre-training and domain adaptation. As both closed-source developers and open-source community lead in pre-training foundation models, domain deployers face the dilemma about whether to use closed-source models via API access or to host open-source models on proprietary hardware. Using closed-source models incurs recurring costs, while hosting open-source models requires substantial hardware investments and may lead to potentially lagging advancements. This paper presents a game-theoretical model to examine the economic incentives behind the deployment choice and the impact of open-source engagement strategies on technological innovation. We find that deployers consistently opt for closed-source APIs when the open-source community engages reactively by maintaining a fixed performance ratio relative to closed-source advancements. However, open-source models can become preferable when a proactive open-source community produces high-performance models independently. Furthermore, we identify conditions under which the engagement and competitiveness of the open-source community can either foster or inhibit technological progress. These insights offer valuable implications for market regulation and the future of technology innovation.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**.

Keywords

Deployment; adaptation; open-source; closed-source

ACM Reference Format:
Yanxuan Wu, Haihan Duan, Xitong Li, and Xiping Hu. 2025. Navigating the Deployment Dilemma and Innovation Paradox: Open-Source versus Closed-Source Models. In *Proceedings of the ACM Web Conference 2025 (WWW '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3696410.3714783

1 Introduction

The capability of ML, especially large language models (LLMs), has seen a remarkable increase due to the scaling of training data, computational resources, and model parameters [25, 42]. Most recently, the paradigm of pretraining and domain adaptation has become increasingly important in the development of LLMs [27]. The landscape of foundational models is characterized by two prominent alternatives: open-source and closed-source. Domain experts are typically responsible for deciding which technology to adopt. Consequently, the development of end technology generally follows the process of pre-training, deployment, and adaptation.

The foundational model market is becoming increasingly competitive, primarily due to the emergence of open-source models. Take LLMs as an example: Stanford has reported that of the 149 foundation models released in 2023, 98 were open-source models, such as LLaMA [41], while 23 were closed-source and accessible via public APIs, such as GPT-4 [32, 33]. Also, there has been a significant increase in the proportion of models released with open access [32]. Most recently, the release of the DeepSeek-V3 model marked a milestone for open-source LLMs, as it is claimed to be not only equivalent to GPT-4o but also achieve efficient inference and cost-effective training [13, 16]. Clearly, the engagement of the open-source community has created a competitive landscape for the development of foundational models [1, 9].

The relationship between a competitive market and innovation is complex, with competition capable of both hindering and fostering innovation [17, 40]. Notably, competition between open-source and closed-source models presents unique dynamics distinct from typical firm-to-firm competition as open-source communities often

[*]Co-corresponding author.
[†]Also with, Guangdong Engineering Center for Social Computing and Mental Health
[‡]Also with, Guangdong-Hong Kong-Macao Joint Laboratory for Emotion Intelligence and Pervasive Computing
[§]Also with School of Medical Technology, , Beijing Institute of Technology.

WWW '25, Sydney, NSW, Australia

ACM ISBN 979-8-4007-1274-6/25/04
https://doi.org/10.1145/3696410.3714783

operate with diverse motivations beyond profit, such as community-driven improvement, accessibility, and transparency [4, 20, 31, 37]. Understanding how open-source versus closed-source competition influence the society-level technological progress is essential, as it can provide insights into the forces that drive or inhibit technological innovation, with implications for future regulation.

Additionally, due to the engagement of the open-source community, deployers face a deployment dilemma, navigating complex economic trade-offs in choosing which technology to adopt. On one hand, self-hosting open-source technology entails high costs associated with requisite hardware resources, such as GPUs[14]. On the other hand, using third-party APIs incurs recurring costs [14]. Furthermore, the performance of foundational technology directly impacts that of end technology, which further affects the revenue generated in the end market [27]. Thus, the choice between self-hosting open-source technology and utilizing third-party APIs involves a complex trade-off from an economic perspective. Understanding this process is crucial to studying the economic and technological consequences of open-source technology.

In this paper, we present a comprehensive game-theoretic model to explore the interactions among a closed-source developer, an open-source community, and a deployer, as well as how these interactions affect the competitive and innovative outcomes of foundation model development. We analyze three distinct scenarios: a baseline scenario with **no open-source engagement**, a scenario with **proactive open-source engagement**, where the community independently innovates and determines model performance, and a scenario with **reactive open-source engagement**, in which the open-source community aligns its performance to maintain relative parity with closed-source advancements.

Our analysis reveals that deployment choices consistently fall into one of three primary outcomes: **API-dominant**, where open-source engagement has no significant impact on market status or decision making of other market players compared to scenarios without an open-source alternative; **API-strategic**, where open-source engagement prompts strategic behaviors from closed source developers, yet developers remain incentivized to adopt closed-source technology; and **self-hosting**, where open-source technology completely supersedes the closed source option. The outcomes are highly dependent on the strategy employed by the open-source community. Our findings indicate that open-source engagement can significantly reshape the innovation landscape for foundation models. Specifically, we identify conditions under which open-source competition paradoxically hampers innovation by discouraging closed-source developers from advancing foundational technologies, as well as scenarios where it fosters a "race-to-the-top," encouraging closed-source developers to innovate more aggressively.

The main contributions of this paper are threefold. First, we provide a theoretical framework based on **multi-stage game** and **subgame perfect equilibrium** to analyze the deployment dilemma facing deployers and define three types of deployment outcomes. Second, we investigate two distinct open-source engagement strategies—proactive and reactive—and characterize the conditions under which each strategy encourages or inhibits innovation in foundational technologies. Finally, we discuss broader implications for innovation, offering insights into how the open-source and closed-source competition can support sustainable innovation.

2 Related Work

There is extensive research on technology innovation and competition. Our work specifically examines, from an economic perspective, the impact of open-source community on innovation within the paradigm of pre-training and fine-tuning.

Open-Source Community. The open-source community has driven significant technological advances by providing freely accessible open-source software (OSS) [5, 12]. For instance, 98 of the 149 foundational LLMs released in 2023 were open-source models, such as LLaMA [32, 41]. Moreover, recent research and technical reports have shown that open-source LLMs are rapidly catching up to closed-source commercial LLMs, with the performance gap often supplemented or even closed [1, 9, 13]. Most recently, DeepSeek-V3 has achieved performance comparable to leading closed-source models such as GPT-4o[13, 16]. However, the incentives behind open-source initiatives have been proven to be far beyond profit [4, 20, 31, 37]. Despite the significant role of OSS, there is still a scarcity of research that quantitatively assesses its value [21]. Our work contributes to the literature by investigating how the open-source community's engagement strategies in innovation influence market dynamics and technological outcomes.

Technology Deployment. For domain-specific deployers, deciding between self-hosting open-source technology and adopting closed-source technology has been challenging. Adopting an API may raise concerns such as data ownership, privacy, and stability [9, 10]. However, self-hosting can be prohibitively expensive due to high hardware requirements. For instance, "regular 16-bit adaptation of a LLaMA 65B parameter model requires more than 780 GB of GPU memory" [14]. Moreover, model performance plays a crucial role in the adoption decision. For example, open-source options may incur additional adaptation costs due to their foundational capabilities lagging behind, while cloud APIs also limit the developer's ability to adapt models with custom data [23]. Given the complexity of the adoption decision, our paper is the first to model this deployment dilemma from an economic perspective.

Economic Models of Technology Innovation. Many studies have addressed technological innovation, production, and cooperation between firms. Empirical research has shown that the relationship between competition and innovation can be either negative or positive, depending on multiple factors such as market structure and innovation strategies [17, 40]. However, the incentives for innovation are typically tied to the profit, which does not apply to the open-source community. Bhaskaran and Krishnan [3] present a model of joint work and decision-making between collaborating firms for new product development. However, this model focuses on cooperation rather than competition between firms. Besides, our paper distinguishes by considering the innovation process within the pre-training and adaptation paradigm.

Machine Learning and Game Theory. Our paper broadly contributes to the field of work that employs game theory to analyze the economics of technology, particularly machine learning (ML) models [26, 30, 34]. Specifically, it adds to the study of technology competition and innovation with open-source engagement. Laufer et al.[27] propose a model for "fine-tuning games"; however, their work focuses on joint development and bargaining between two firms instead of open-source engagement and competition. Later,

Xu et al.[43] propose a model to explore how foundation model openness affects the "fine-tuning game" with competing deployers. However, their model neglects the foundation model developer's direct contribution to innovation.

3 Model

In this section, we introduce the game-theoretic framework, with three variations. The baseline model includes only a closed-source foundation model developer, a domain-specific deployer, and end users. In the other two models, an open-source community is introduced, each employing a distinct strategy in technology innovation. Both the closed-source developer and the open-source community focus only on foundation technology, while the deployer must choose one technology for deployment and adaptation.

3.1 Model Setting

Closed-source Technology Developer. The developer F develops a closed-source foundation technology to a performance level $\alpha_0 \in \mathbb{R}_0^+$ and prices the unit usage of the API as γc ($\gamma > 1$), where $c \in \mathbb{R}^+$ is the unit cost of operating the technology, and γ is a multiplier that determines the price.

Open-Source Community. The open-source community O provides a technology at performance level $\alpha \in \mathbb{R}_0^+$ for free adoption. We define O as either ***reactive*** or ***proactive***. Specifically, reactive O means that O maintains the relative performance of the open-source technology in relation to the closed-source one, according to a parameter m, such that $\alpha = m\alpha_0$. In contrast, proactive O means that O independently decides the performance level of the open-source model α, which is no influenced by α_0.

Domain-specific Deployer. The deployer S first decides whether to self-host the open-source technology or use the closed-source API, captured by a variable I: the open-source option if $I = 0$ and the API option if $I = 1$. The deployer then adapts the technology to a level $\alpha_1 \in \mathbb{R}_0^+$, where $\alpha_1 \geq \alpha_0$, and sets the unit price of the end technology as $p \in \mathbb{R}_0^+$. Notably, S must operate its own hardware infrastructure if hosting the open-source technology and purchase the API if using the closed-source technology.

End Users. The end users U consume the technology, resulting in a market demand D. The market demand is determined by both technology's price p and technology's performance α_1 [19, 38, 44]. We assume there is a function $D : \mathbb{R}_0^+ \times \mathbb{R}_0^+ \to \mathbb{R}^+$ such that $D(p, \alpha_1)$ is the demand in the end market with technology at level α_1 and unit price p. $D(p, \alpha_1)$ is monotonically increasing with α_1 and monotonically decreasing with p. We assume that end users cannot consume the foundation technology directly and the demand function is publicly known.

Revenue. Revenue is calculated as demand multiplied by unit price [36]. For each unit of the end technology consumed, one unit of API will be used simultaneously if the closed-source technology is deployed as the foundation technology. The developer F gains revenue $R_F = \gamma c D(p, \alpha_1) I$ by providing inference API to S. The deployer S gains revenue $R_S = pD(p, \alpha_1)$ from the end market.

Cost. Both F and S have two parts of cost: technology production (development or adaptation) cost and operation cost. F has a development cost $C_F(\alpha_0) : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ to produce a general technology at level α_0 and an operation cost of c per unit. S faces a adapting cost $C_S^{api}(\alpha_1; \alpha_0) : \mathbb{R}^+ \to \mathbb{R}^+$ to adapt the closed-source technology from level α_0 to α_1 or a cost function $C_S^{self}(\alpha_1; \alpha) : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ to adapt the open-source technology from level α to α_1. Besides, S faces an operation cost of c per unit if self-hosts open-source technology or γc per unit if uses API. We assume these cost functions are publicly known.

Utility. The utility of developer F, denoted as U_F, and of deployer S, denoted as U_S, are defined as revenue minus cost: $R_i - C_i$, where $i = S, F$. We introduce the following notations for utility: $U_S^{api}, U_S^{self}, U_F^{api}, U_F^{self}$ to represent the utilities of the deployer S, the developer F under the API and self-hosting scenarios. The formulas are as follows:

$$U_S^{api} = (p - \gamma c) D(p, \alpha_1) - C_S^{api}(\alpha_1; \alpha_0) \tag{1}$$

$$U_S^{self} = (p - c) D(p, \alpha_1) - C_S^{self}(\alpha_1; \alpha) \tag{2}$$

$$U_F^{api} = (\gamma c - c) D(p, \alpha_1) - C_F(\alpha_0) \tag{3}$$

$$U_F^{self} = 0 \tag{4}$$

Technology Innovation Outcome (Societal Level). The societal-level technology innovation outcomes are defined as:

$$\alpha_0^{\text{soc}} = I\alpha_0 + (1 - I)\alpha; \alpha_1^{\text{soc}} = \alpha_1. \tag{5}$$

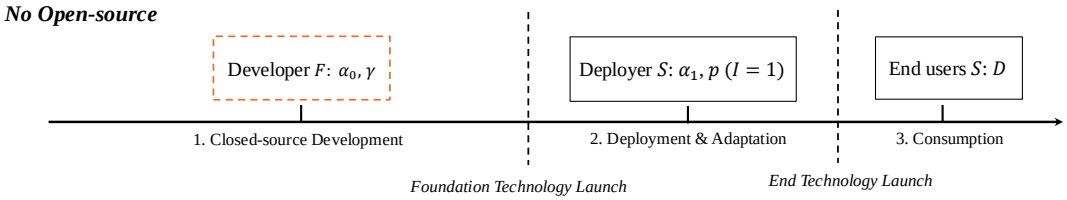
3.2 Game Process

The game process varies according to the open-source community's engagement strategy, resulting in three distinct models as summarized below and illustrated in Figure 1.

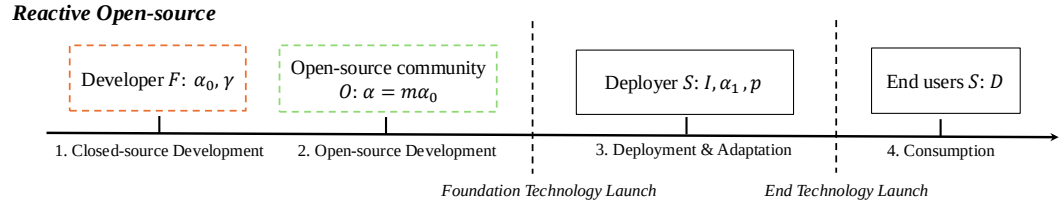
Baseline Game - No Open-Source Engagement. In this scenario, the open-source community O chooses not to engage in the market. Thus, the developer F first brings the foundation technology to a performance level α_0 and sets the unit price for API usage as γc by deciding the multiplier γ. Then, the deployer S adapts the technology to level α_1 and sets the unit price for end technology as p. The end users consume the technology with demand $D(p, \alpha_1)$. Revenue is generated for both F through API usage fees and S through end-user sales.

Game 1 - Proactive Open-Source Engagement. In this scenario, the open-source community O adopts a proactive engagement strategy. First, O independently develops its technology to level α. Second, the closed-source developer F develops its foundation technology to level α_0 and sets the API unit price as γc by choosing the multiplier γ. The deployer S then chooses between self-hosting the open-source technology or accessing the closed-source technology via API. After deployment, S adapts the selected technology to level α_1 and sets the unit price p, resulting in demand $D(p, \alpha_1)$ from end users. The consumption of the end technology generates revenue for the deployer, and for the developer as well, but only if the closed-source technology is chosen.

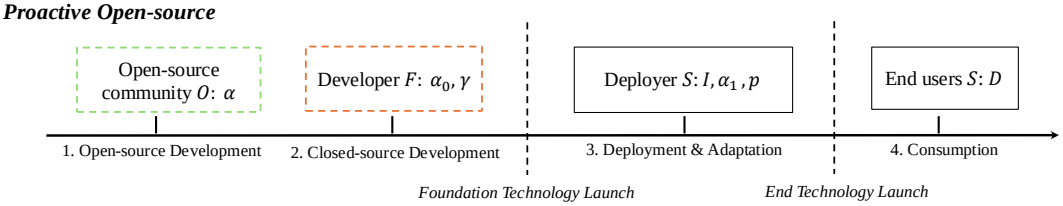
Game 2 - Reactive Open-Source Engagement. In this scenario, the open-source community O follows a reactive engagement strategy. It initially announces this approach by specifying a performance ratio m to indicate how closely it will track the closed-source technology developed by F. Once F has brought its technology to level α_0 and sets the API price as γc, O offers an open-source alternative at level $m\alpha_0$. The subsequent steps, including deployment, adaptation, and consumption, are identical to those in Game 1.



(a) Baseline Game - No Open-source Engagement



(b) Game 1 - With Reactive Open-source Engagement



(c) Game 2 - With Proactive Open-source Engagement

Figure 1: An illustration of the processes for the three games. Game 1 and Game 2 differ from the baseline model in the foundation technology development stage, as they involve open-source community. In Step 2 of the baseline game, $I = 1$ always holds, whereas in Game 1 and Game 2, I can be either 0 or 1, reflecting the deployment decision. The distinction between Game 1 and Game 2 arises from the strategy adopted by the open-source community.

3.3 Solution of the Model

In this section, we provide the general equilibrium of each model derived through backward induction, following the sequential decision-making of the deployer S and the developer F. The solution involves two key steps.

Step 1: Assuming a fixed α_0 and γ (or also α), S maximizes its utility by choosing the optimal technology to deploy, the optimal domain technology level α_1, and the optimal price p. Formally, S solves the following optimization problem:

$$I^*, \alpha_1^*, p^* = \arg\max_{I,\alpha_1,p} IU_S^{api} + (1-I)U_S^{self}. \tag{6}$$

The deployer S will choose to self-host the technology if its utility from self-hosting, denoted as $U_S^{self}(\alpha_1^*, p^*)$, is greater than its utility from using the API service, denoted as $U_S^{api}(\alpha_1^*, p^*)$.

Step 2 in Baseline Game: Anticipating S's response to its decisions regarding the foundation technology level α_0 and the inference price parameter γ, F sets optimal α_0 and γ to maximize its own utility. This leads to the following optimization problem for F:

$$\alpha_0^*, \gamma^* = \arg\max_{\alpha_0,\gamma} U_F^{api}(\alpha_1^*, p^*). \tag{7}$$

Step 2 in Game 1 and Game 2: Similar to the step 2 in the baseline game, F would optimize its utitlity by deciding:

$$\alpha_0^*, \gamma^* = \arg\max_{\alpha_0,\gamma} U_F^{api}(\alpha_1^*, p^*). \tag{8}$$

As F receives revenue only if S decides to use the API, Step 2 is only meaningful when I = 1, which means the restriction:

$$U_S^{self}(\alpha_1^*, p^*) \geq U_S^{api}(\alpha_1^*, p^*). \tag{9}$$

Also, F would develop the technology only when it expects a positive utility, as $U_F^{api}(\alpha_0^*, \gamma^*, \alpha_1^*, p^*) \geq 0$. Else, F would exit the market.

The solution depends on market conditions and the engagement strategy of O. Thus, we offer a set of definitions to help characterize the possible regimes of solutions according to the developer's strategic behavior and the deployer's deployment decisions.

Definition 3.1 (API-DOMINANT SOLUTION). The ***API-dominant solution*** is the solution when developer F's optimal decision, $\alpha_0^*, \gamma^* = \arg\max_{\alpha_0,\gamma} U_F^{api}$, naturally leads to $U_S^{self} \geq U_S^{api}$ and $U_F^{api} \geq 0$. In this situation, S chooses to use API as it naturally dominates the self-hosting option.

Definition 3.2 (API-STRATEGIC SOLUTION). The ***API-strategic solution*** is the solution when developer F's optimal decision, $\alpha_0^*, \gamma^* =$

$\arg\max_{\alpha_0,\gamma} U_{F,api}$ under the constraint $U_S^{self} \geq U_S^{api}$ naturally satisfies $U_F^{api} \geq 0$, while only $\alpha_0^*, \gamma^* = \arg\max_{\alpha_0,\gamma} U_{F,api}$ leads to $U_S^{self} < U_S^{api}$. In this situation, F strategically incentivizes S to choose the API by ensuring that S achieves greater profit from the API option compared to the self-hosting option.

Definition 3.3 (SELF-HOSTING SOLUTION). The ***self-hosting solution*** is the solution when no sets of $\{\alpha_0, \gamma\}$ exists that simultaneously satisfies $U_S^{self} \geq U_S^{api}$ and $U_F^{api} \geq 0$. In this situation, S opts to self-host the open-source technology rather than utilize the API provided by F.

Notice that any solution will fall into one of three categories: an API-dominant solution, an API-strategic solution, or a self-hosting solution. Analyzing the general form is challenging due to multiple sequential decision steps, each requiring the consideration of multiple factors. At each stage, either the developer or deployer must determine optimal values for multiple variables such as performance levels, pricing, and deployment choices, which interact in complex ways across stages. This interdependence makes deriving a general solution intricate, necessitating specifications to gain clearer insights. Accordingly, in Section 4, we present theorems about the solutions under specified demand and cost functions. Subsequently, in Section 5, we present theorems about technological outcomes within a broader class of utility functions.

4 Analysis of Separable Multiplicative Demand and Quadratic Cost

In order to produce closed-form solutions and understand how the players in the model interact with each other, we take the form of separable multiplicative demand and quadratic cost, which are commonly used in business research.

The demand function is expressed in separable multiplicative form: $D(p, \alpha_1) = d_1(p) * d_2(\alpha_1)$, where $d_1(p)$ measures the effect of price and $d_2(\alpha_1)$ measures the effect of quality [2, 11]. For the price-dependent part, the linear model has been widely used in the economic and business literature, including theoretical models [6, 15, 35, 36, 39] and empirical estimations [7, 22]. For the quality effect, we take the form $d_2(\alpha_1) = \alpha_1$ [29]. In our setting, quality refers to the level of technological development. Thus, we get the demand function as:

$$D(p, \alpha_1) = (a - bp)\alpha_1 \tag{10}$$

Here, $a > 0$ and $b > 0$ are constant parameters representing the market size and price sensitivity respectively. The demand would decrease with price and increase with product quality. Also, there would be no sales if the quality (technology level) is zero.

The quadratic form for modeling cost is widely adopted in the literature on economics and management science [3, 8, 18, 24, 28, 44]. Following Laufer et al. [27], we assume that the cost increases quadratically with advancements in technology:

$$\phi(\alpha_0) = K_F \alpha_0^2 \tag{11}$$

$$\phi(\alpha_1; \alpha_0) = K_S^{api}(\alpha_1 - \alpha_0)^2 \tag{12}$$

$$\phi(\alpha_1; \alpha) = K_S^{self}(\alpha_1 - \alpha)^2 \tag{13}$$

Here, K_F, K_S^{self}, and K_S^{api} are positive constants, reflecting that marginal costs should increase with technology advancement [27]. Note that hardware costs are always included in the pre-training stage. However, in the adaptation stage, deployer S does not need to set up hardware if using APIs. Besides, non-hardware costs, including expenses such as labor, exist in both stages. Thus, K_F and K_S^{self} include both non-hardware and hardware costs while K_S^{api} includes only non-hardware costs. The cost factors are decomposed as:

$$K_F = K_{PRE} + K_G \tag{14}$$

$$K_S^{api} = K_{FT} \tag{15}$$

$$K_S^{self} = K_{FT} + K_G \tag{16}$$

- K_{PRE} represents the non-hardware cost component in the pre-training cost K_F
- K_{FT} represents the non-hardware cost component in the adapting cost K_S^{self} and K_S^{api}
- K_G represents the hardware cost component in the pre-training cost K_F and the adapting cost K_S^{self}

Thus, the utilities of developer F, deployer S, and end users U are as follows:

$$U_S^{api} = (p - \gamma c)(a - bp)\alpha_1 - K_S^{api}(\alpha_1 - \alpha_0)^2 \tag{17}$$

$$U_F^{api} = (\gamma c - c)(a - bp)\alpha_1 - K_F \alpha_0^2 \tag{18}$$

$$U_S^{self} = (p - c)(a - bp)\alpha_1 - K_S^{self}(\alpha_1 - \alpha)^2 \tag{19}$$

$$U_F^{self} = 0 \tag{20}$$

4.1 Equilibrium without O

Theorem 4.1 (API-Dominant Solution). *Without O, the equilibrium always falls into the API-dominant solution, yielding the following strategies:*

$$\gamma^* = \frac{5\theta + 3 - 2\beta(3 + \theta) - \sqrt{\delta}}{8(1 - \beta)}, \tag{21}$$

$$\alpha_0^* = \frac{b}{4K_F}\left(c\gamma^* - c\right)\left(\frac{a}{b} - c\gamma^*\right), \tag{22}$$

$$p^* = \frac{a}{b} + c\gamma^*, \tag{23}$$

$$\alpha_1^* = \alpha_0^* + \frac{b\left(\frac{a}{b} - c\gamma^*\right)^2}{8K_{FT}}, \tag{24}$$

where:

$$\delta = (5\theta + 3 - 2\beta(3 + \theta))^2 - 16(1 - \beta)(\theta^2 + 3\theta - 2\beta(1 + \theta)),$$

$$\theta = \frac{a}{bc}, \quad \beta = \frac{K_{FT}}{K_{PRE} + K_G}.$$

A proof of the above result is provided in Appendix A.1. Notice that the deployer S's decision on adaptation effort equals $\alpha_1^* - \alpha_0^* = \frac{b\left(\frac{a}{b} - c\gamma^*\right)^2}{8K_{FT}}$, which is independent of developer F's decision on foundation technology performance α_0^*. This finding is consistent with the finding from a previous research by Laufer et al. [27]. Moreover, both the developer F's decision on foundation technology performance α_0^* and the deployer S's decision on domain-specific technology performance α_1^* are independent of the open-source technology performance, which is reasonable as

the open-source technology is naturally dominated by the closed-source technology and cannot influence the market. The results are plotted in Appendix Figure B.1.

4.2 Subgame Perfect Equilibrium with a Reactive O under a Fixed m

When the open-source community adopts a reactive strategy, the subgame perfect equilibrium under a given m may lead to different solutions based on various market factors, captured by cost-related parameters $\{K_G, K_{FT}, K_{PRE}, c\}$ and demand-related parameters $\{a, b\}$. Among these factors, we focus on K_G, which represents the hardware cost. First, we specify the forms of solutions. Then, we demonstrate how K_G and m characterize the equilibrium solution.

Theorem 4.2. *With a reactive O, the API-dominant solution results in strategies that are identical in form to those presented in Theorem 4.1.*

This conclusion is straightforward because, under an API-dominant solution, the engagement of the open-source community does not affect the dynamics of the original game, leaving the strategic outcome unchanged.

Theorem 4.3 (API-Strategic Solution with Reactive O). *With reactive O, the API-strategic solution yields strategies:*

$$I^* = 1, p^* = \frac{1}{2}\left(\frac{a}{b} + c\gamma^*\right), \alpha_1^* = \alpha_0^* + \frac{b\left(\frac{a}{b} - c\gamma^*\right)^2}{8K_{FT}},$$

and α_0^* and γ^* is the solution of:

$$\begin{cases} \left(16(\theta-\gamma^*)^2 - m(\theta-1)^2\right)\alpha_0^* = \left(\frac{(\theta-1)^4}{K_{FT}+K_G} - \frac{(\theta-\gamma^*)^4}{K_{FT}}\right)bc^2, \\ \frac{2\left(bc^2(\gamma^*-1)(\theta-\gamma^*)-4(K_{PRE}+K_G)\alpha_0^*\right)}{\alpha_0^*(\theta+1-2\gamma^*)+\frac{bc^2}{8K_{FT}}(\theta-\gamma^*)^2(3+\theta-4\gamma^*)} = \frac{-bc^2\left((\theta-\gamma^*)^2-m(\theta-1)^2\right)}{\frac{bc^2}{8K_{FT}}(\theta-\gamma^*)^3+(\theta-\gamma)\alpha_0^*}, \end{cases}$$

,where $\theta = \frac{a}{bc}$.

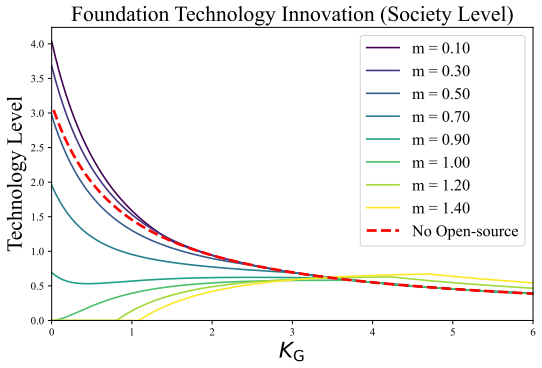
A proof of Theorem 4.3 is provided in Appendix A.2. Note that the existence of a feasible solution in Theorem 4.3 is guaranteed by Theorem 4.4, while its uniqueness is ensured by maximizing U_F.

Theorem 4.4 (Guaranteed API Outcome). *With reactive O, the equilibrium always falls into either an API-dominant or an API-strategic solution, meaning there always exists a set of $\{\alpha_0^*, \gamma^*, \alpha_1^*, p^*\}$ that satisfies $U_S^{self} \geq U_S^{api}$ and $U_F \geq 0$ simultaneously.*

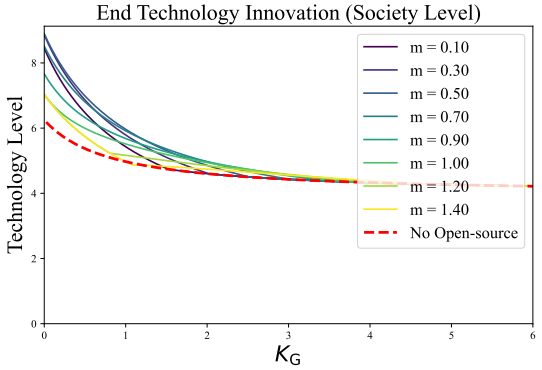
A proof of Theorem 4.4 is provided in Appendix A.3. Notably, when O adopts a reactive strategy, F can influence technology innovation in a way that strategically deters O and encourages S to adopt the closed-source technology. Counterintuitively, even when m is high—indicating that the open-source technology significantly outperforms the closed-source technology—the deployer S is still incentivized to utilize the closed-source technology via API.

Next, we illustrate the impact of reactive open-source engagement on technological outcomes using numerical results. In Figure 2, we show the results using parameters ($a = 8, b = 1, c = 0.5, K_{FT} = K_{PRE} = 1$), and let m range from 0.1 to 1.4. The robustness checks are provided in Appendix C.

As shown in Figure 2a, foundation technology innovation is generally hindered when m is high. This is because, at higher m values,



(a) Foundation Technology



(b) End Technology

Figure 2: Technological Outcomes - No vs. Reactive Open-source Engagement ($a = 8, b = 1, c = 0.5, K_{FT} = K_{PRE} = 1$)

the closed-source developer may choose to strategically reduce technology performance to deter open-source alternatives. When m decreases to a low level, the closed-source developer can gain a greater technology advantage by enhancing performance, which becomes an economical strategy to attract the deployer toward the closed-source API.

Interestingly, we observe in Figure 2b that end technology experiences higher levels of innovation. This outcome arises because the closed-source developer not only adjusts technology performance but also lowers the API price, allowing the deployer to achieve higher unit profit from end technology. This incentivizes the deployer to further adapt the technology to a higher level, thereby driving technology innovation in the specific domain.

4.3 Subgame Perfect Equilibrium with a Proactive O under a Fixed α

Similar to the previous section, we first specify the forms of each solution under a fixed α and then analyze the equilibrium solution.

Theorem 4.5. *With a proactive O, the API-dominant solution results in strategies that are identical in form to those presented in Theorem 4.1.*

Theorem 4.6 (API-Strategic Solution with Proactive O). *With proactive O, the API-strategic solution yields strategies:*

$$I^* = 1, p^* = \frac{1}{2}\left(\frac{a}{b} + c\gamma^*\right), \alpha_1^* = \alpha_0^* + \frac{b\left(\frac{a}{b} - c\gamma^*\right)^2}{8K_{FT}},$$

and α_0^* and γ^* are the solution of:

$$\begin{cases} 16\left((\theta-\gamma^*)^2\alpha_0^* - (\theta-1)^2\alpha\right) = \left(\frac{(\theta-1)^4}{(K_{FT}+K_G)} - \frac{(\theta-\gamma^*)^4}{K_{FT}}\right) bc^2, \\ \frac{bc^2(\gamma^*-1)(\theta-\gamma^*) - 4(K_{PRE}+K_G)\alpha_0^*}{\alpha_0^*(\theta+1-2\gamma^*) + \frac{bc^2}{8K_{FT}}(\theta-\gamma^*)^2(3+\theta-4\gamma^*)} = \frac{-4bc^2(\theta-\gamma^*)}{8\alpha_0^* + \frac{bc^2}{K_{FT}}(\theta-\gamma^*)^2}, \end{cases}$$

, where $\theta = \frac{a}{bc}$.

A proof of Theorem 4.6 is provided in Appendix A.4. Note that the solution from Theorem 4.6 must always satisfy $\alpha_0^* \geq 0$, $\gamma^* \geq 1$, and $U_F(p^*, \alpha_1^*, \alpha_0^*, \gamma^*) \geq 0$. If these conditions are not met, the equilibrium defaults to the self-hosting solution described below.

Theorem 4.7 (Self-Hosting Solution with Proactive O). *With proactive O, self-hosting solution yields strategies:*

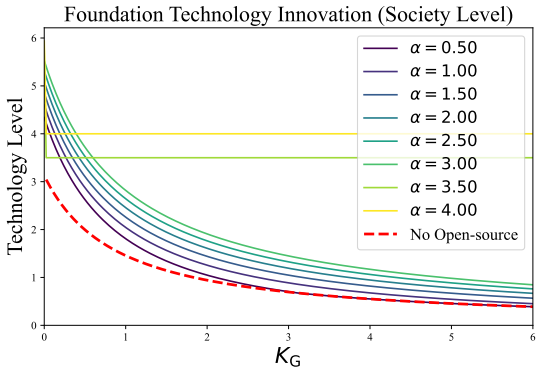
$$\gamma^* = 1, \alpha_0^* = 0, I^* = 0, p^* = \frac{1}{2}\left(\frac{a}{b} + c\right), \alpha_1^* = \alpha + \frac{b\left(\frac{a}{b} - c\right)^2}{8(K_{FT} + K_G)},$$

A proof of Theorem 4.7 is provided in Appendix A.5. Note that the developer F's decision on the foundation technology performance α_0^* is always zero, indicating that F exits the game. Consequently, the deployer S adopts a self-hosting approach. Interestingly, the unit price of the end technology, p, remains constant. This is due to the marginal cost of operations being fixed at c and the end users' price sensitivity remaining stable at b. Additionally, as the hardware cost K_G decreases, S is incentivized to enhance the technology to a higher performance level as the incremental advancement $(\alpha_1 - \alpha)$ grows. Also, the utilities of the deployer, U_S, and the end users, U_U, increase as the hardware cost K_G decreases.

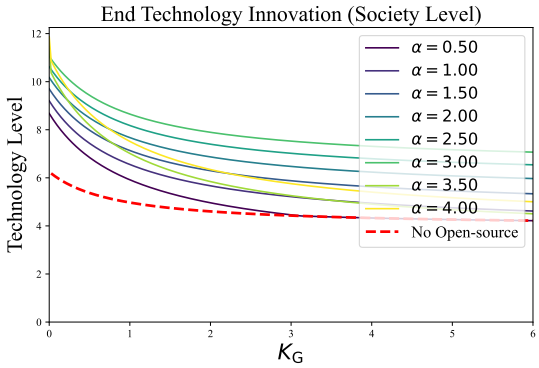
Theorem 4.8 (Existence of Self-Hosting Outcome). *With proactive O, given cost-related parameters $\{K_{FT}, K_{PRE}, c\}$ and demand-related parameters $\{a, b\}$, there exists a threshold $\alpha^H \in \mathbb{R}^+$ such that $\forall K_G$, the game results in a self-hosting solution if $\alpha \in (\alpha^H, +\infty)$.*

A proof of Theorem 4.8 is provided in Appendix A.6. The insight is that when O adopts a proactive strategy and develops the open-source technology to a sufficiently high performance level, the developer F may initially be able to incentivize the deployer by either enhancing the closed-source technology or lowering the API price. During this process, profit gradually transfers from the developer to the deployer. However, as the performance of the open-source technology continues to increase, a point is reached where the developer can no longer offer enough incentives to attract the deployer while still ensuring its own profitability. Consequently, if the open-source technology achieves a high enough performance level, the closed-source developer foresees an unprofitable market and opts not to enter, ultimately resulting in a self-hosting outcome.

Next, we illustrate the impact of proactive open-source engagement on technology outcomes using numerical results. In Figure 3, we show the results using parameters $(a = 8, b = 1, c = 0.5, K_{FT} = K_{PRE} = 1)$, and let m range from 0.5 to 4. The robustness checks are provided in Appendix C.



(a) Foundation Technology



(b) End Technology

Figure 3: Technology Outcomes - No vs. Proactive Open-source Engagement ($a = 8$, $b = 1$, $c = 0.5$, $K_{FT} = K_{PRE} = 1$)

As shown in Figure 3, proactive open-source engagement may lead to an increase in both foundation and end technology performance levels. The intuition is that when the open-source community independently advances, rather than adjusting to closed-source performance, the closed-source developer cannot deter open-source technology by strategically reducing model performance. Instead, the developer enhances the closed-source technology and lowers the API price to attract deployers to use the closed-source API, ultimately benefiting both foundation and end technology.

However, two horizontal lines appear in Figure 3a at $\alpha = 3.5$ and $\alpha = 4$, respectively. This indicates that, at these levels, open-source technology becomes advanced enough to drive the closed-source developer to exit the market. As we can see in Figure 3a, these levels of open-source innovation may actually hurt innovation in both foundation and end technology.The intuition is that the API option naturally forms a sharing scheme for hardware infrastructure between foundation model developers and deployers, which can be beneficial. However, self-hosting open-source models incurs inefficiencies in hardware infrastructure usage. Thus, in extreme conditions where deployers are on the verge of choosing between an open-source or closed-source model, the technology level under the closed-source option can be higher than that under the open-source option.

5 Impact of Open-source Engagement on Foundation Technology Innovation

In the previous section, we showed the impact of open-source engagement using numerical results under specified demand and cost functions. In this section, we examine the technological outcomes within a broader class of utility functions. First, we define the class of utility functions to which our conclusions apply. Then, we formally state the conditions under which open-source engagement may either encourage or hinder foundation technology innovation.

5.1 Concave and Unimodal Utility

First, we introduce two assumptions regarding the utility functions.

Definition 5.1 (Strictly Unimodal Function). A function $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is called strictly unimodal over x and y if there exists a value $m \in D \subset \mathbb{R}$ such that f is strictly increasing for $x \leq m$ and strictly decreasing for $x \geq m$, and there exists a value $n \in D \subset \mathbb{R}$ such that f is strictly increasing for $y \leq n$ and strictly decreasing for $y \geq n$.

Assumption 1: The developer's utility $U_F(\alpha_0, \gamma)$ is strictly concave; that is, $\frac{\partial^2 U_F}{\partial \alpha_0^2} < 0$ and $\frac{\partial^2 U_F}{\partial \gamma^2} < 0$.

Assumption 2: The developer's utility $U_F(\alpha_0, \gamma)$ is strictly unimodal. This implies there exists a maximum utility at some values of α_0 and γ over their respective ranges.

Note: that the analysis in Section 4 satisfies these assumptions, ensuring that our conclusions hold within that framework.

5.2 Foundation Technology Innovation

Here, we formally state the theorems identifying the conditions under which open-source community engagement enhances or hinders foundation technology innovation.

Theorem 5.2. *Assume that the developer's strategy with no open-source engagement is $\{\alpha_0^*, \gamma^*\}$, yielding utility $U_S^{api}(\alpha_0^*, \gamma^*)$. After the engagement of a reactive open-source community, the developer's strategy shifts to $\{\alpha_0'^*, \gamma'^*\}$, yielding utility $U_S^{self}(\alpha_0'^*, \gamma'^*)$.*

The foundation technology level decreases ($\alpha_0'^* < \alpha_0^*$) if:

$$\frac{\partial U_S^{api}}{\partial \alpha_0} < \frac{\partial U_S^{self}}{\partial \alpha_0} \quad \text{and} \quad \frac{\partial^2 U_F}{\partial \gamma \partial \alpha_0} > 0.$$

Conversely, the foundation technology level increases ($\alpha_0'^* > \alpha_0^*$) if:

$$\frac{\partial U_S^{api}}{\partial \alpha_0} > \frac{\partial U_S^{self}}{\partial \alpha_0} \quad \text{and} \quad \frac{\partial^2 U_F}{\partial \gamma\, \partial \alpha_0} < 0.$$

Theorem 5.3. *Assume that the developer's strategy with no open-source engagement is $\{\alpha_0^*, \gamma^*\}$, yielding utility $U_S^{self}(\alpha_0^*, \gamma^*)$. After proactive engagement by the open-source community, the developer's strategy shifts to $\{\alpha_0'^*, \gamma'^*\}$, yielding utility $U_S^{self}(\alpha_0'^*, \gamma'^*)$.*

The foundation technology level increases ($\alpha_0'^* > \alpha_0^*$) if:

$$\frac{\partial^2 U_F}{\partial \gamma \partial \alpha_0} < 0.$$

The proofs for Theorem 5.2 and Theorem 5.3 are provided in Appendix A.7 and Appendix A.8, respectively. Note that these conditions are sufficient but not necessary for the stated outcomes.

These theorems highlight the differences and similarities between reactive and proactive open-source engagement in influencing foundation technology innovation as below.

- **Reactive Open-source Engagement**: When the deployer's utility gain from using the API is less sensitive to α_0 than the self-hosting utility, the developer may strategically reduce the open-source competitiveness by lowering the technology performance. Due to the positive interaction between α_0 and γ in closed-source models, lowering γ can help mitigate the rate of utility decline resulting from reduced technology performance. Conversely, if the developer observes that the deployer's utility from the API is highly sensitive to α_0 compared to self-hosting utility, a 'race-to-the-top' scenario arises where the developer innovates more aggressively. With a negative interaction between α_0 and γ, lowering γ helps counteract the rate of utility loss from potential over-innovation. Additionally, a lower API price always incentivizes deployers to adopt the API.
- **Proactive Open-source Engagement**: As technology improves, the developer's utility may experience diminishing returns. Given the negative interaction between α_0 and γ, lowering γ can help reduce the rate of utility decline from potential over-innovation. A lower API price also incentivizes deployers to adopt the API. However, if the open-source models are too advanced, the closed-source model developer may exit the market.

6 Conclusion

This paper proposes a theoretical model that analyzes the interactions among closed-source developers, the open-source community, and deployers under the paradigm of pretraining and adaptation. By examining three scenarios surrounding open-source engagement strategy — no, proactive, and reactive engagement — the model highlights how different open-source strategies can shape deployment outcomes and impact innovation.

Our analysis are particularly valuable for innovators and regulators, as they provide insights into factors impacting technological progress. We encourage regulators to design policies to address key concerns such as ownership of technology and hardware price. For example, policymakers might incentivize open-source contributions through funding mechanisms or tax benefits, ensuring that the community has sufficient resources to innovate independently. Additionally, regulators must prevent unethical innovation through mimicry and copying patents, especially within the open-source community. Such interventions can help balance closed-source and open-source models, ensuring sustainable innovation.

Future research could build on this model by investigating additional factors such as the diverse motivations within open-source communities and the competition in end markets. These considerations are particularly relevant as technologies continue to evolve, especially the AI market. We believe that societal outcomes are essential in shaping the technology market, and formalizing these considerations through theoretical models can provide a more comprehensive view of the ecosystem. By doing so, researchers and policymakers can help guide balanced and sustainable innovation strategies that maximize societal benefits.

References

[1] Samy Ateia and Udo Kruschwitz. 2024. Can Open-Source LLMs Compete with Commercial Models? Exploring the Few-Shot Performance of Current GPT Models in Biomedical Tasks. *arXiv preprint* (July 2024). https://arxiv.org/abs/2407.13511

[2] Barry L Bayus. 1995. Optimal dynamic policies for product and process innovation. *Journal of Operations Management* 12, 3-4 (1995), 173–185.

[3] Sreekumar R Bhaskaran and Vish Krishnan. 2009. Effort, revenue, and cost sharing mechanisms for collaborative new product development. *Management Science* 55, 7 (2009), 1152–1169.

[4] Andrea Bonaccorsi and Cristina Rossi. 2003. Why open source software can succeed. *Research policy* 32, 7 (2003), 1243–1258.

[5] Andrea Bonaccorsi and Cristina Rossi. 2003. Why Open Source software can succeed. *Research Policy* 32, 7 (2003), 1243–1258. doi:10.1016/S0048-7333(03)00051-9

[6] Jeremy I Bulow, John D Geanakoplos, and Paul D Klemperer. 1985. Multimarket oligopoly: Strategic substitutes and complements. *Journal of Political economy* 93, 3 (1985), 488–511.

[7] Oscar R Burt and Durward Brewer. 1971. Estimation of net social benefits from outdoor recreation. *Econometrica: Journal of the Econometric Society* (1971), 813–827.

[8] Gary H Chao, Seyed MR Iravani, and R Canan Savaskan. 2009. Quality improvement incentives and product recall cost sharing contracts. *Management science* 55, 7 (2009), 1122–1138.

[9] Hailin Chen, Fangkai Jiao, Xingxuan Li, Chengwei Qin, and et al. 2024. ChatGPT's One-year Anniversary: Are Open-Source Large Language Models Catching up? *arXiv preprint* (2024). https://arxiv.org/abs/2311.16989

[10] Lingjiao Chen, Matei Zaharia, and James Zou. 2023. How is ChatGPT's behavior changing over time? *arXiv preprint* (2023). https://arxiv.org/abs/2307.09009

[11] Régis Chenavaz. 2012. Dynamic pricing, product and process innovation. *European Journal of Operational Research* 222, 3 (2012), 553–557.

[12] Bretthauer David. 2001. Open Source Software: A History. *Published Work* (2001). https://digitalcommons.lib.uconn.edu/libr_pubs/7

[13] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, and et al. 2024. DeepSeek-V3 Technical Report. (2024). arXiv:2412.19437 [cs.CL] https://arxiv.org/abs/2412.19437

[14] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. *arXiv preprint* (2023). https://arxiv.org/abs/2305.14314

[15] Gadi Fibich, Arieh Gavious, and Oded Lowengart. 2003. Explicit solutions of optimization models and differential games with nonsmooth (asymmetric) reference-price effects. *Operations Research* 51, 5 (2003), 721–734.

[16] Gemma Conroy and Smriti Mallapaty. 2025. How China Created AI Model DeepSeek and Shocked the World. *Nature* (Jan. 2025). doi:10.1038/d41586-025-00259-0

[17] Richard Gilbert. 2006. Looking for Mr. Schumpeter: Where are we in the competition–innovation debate? (2006).

[18] Sudheer Gupta. 2008. Research note—Channel structure with knowledge spillovers. *Marketing Science* 27, 2 (2008), 247–261.

[19] Haresh Gurnani and Murat Erkoc. 2008. Supply contracts in manufacturer-retailer interactions with manufacturer-quality and retailer effort-induced demand. *Naval Research Logistics (NRL)* 55, 3 (2008), 200–217.

[20] Il-Horn Hann, Jeff Roberts, Sandra Slaughter, and Roy Fielding. 2002. Economic incentives for participating in open source software projects. *ICIS 2002 Proceedings* (2002), 33.

[21] Manuel Hoffmann, Frank Nagle, and Yanuo. Zhou. 2024. The Value of Open Source Software. *Harvard Business School Strategy Unit Working Paper*, Article 24-038 (2024). https://ssrn.com/abstract=46931483

[22] Richard A Ippolito and Robert T Masson. 1978. The social cost of government regulation of milk. *The Journal of Law and Economics* 21, 1 (1978), 33–65.

[23] Chandra Irugalbandara, Ashish Mahendra, Roland Daynauth, Tharuka Kasthuri Arachchige, Jayanaka Dantanarayana, Krisztian Flautner, Lingjia Tang, Yiping Kang, and Jason Mars. 2024. Scaling Down to Scale Up: A Cost-Benefit Analysis of Replacing OpenAI's LLM with Open Source SLMs in Production. In *2024 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 280–291. doi:10.1109/ISPASS61541.2024.00034

[24] Yannan Jin, Qiying Hu, Sang Won Kim, and Sean X Zhou. 2019. Supplier development and integration in competitive supply chains. *Production and Operations management* 28, 5 (2019), 1256–1271.

[25] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, and et al. 2020. Scaling Laws for Neural Language Models. *arXiv preprint* (Jan. 2020). https://arxiv.org/abs/2001.08361

[26] Jon Kleinberg and Manish Raghavan. 2020. How do classifiers induce agents to invest effort strategically? *ACM Transactions on Economics and Computation (TEAC)* 8, 4 (2020), 1–23.

[27] Benjamin Laufer, Jon Kleinberg, and Hoda Heidari. 2024. Fine-Tuning Games: Bargaining and Adaptation for General-Purpose Models. *ACM* 11 (May 2024), 66–76. doi:10.1145/3589334.3645366

[28] Cuihong Li. 2013. Sourcing for supplier effort and competition: Design of the supply base and pricing mechanism. *Management Science* 59, 6 (2013), 1389–1406.

[29] Guowei Liu, Jianxiong Zhang, and Wansheng Tang. 2015. Joint dynamic pricing and investment strategy for perishable foods with price-quality dependent demand. *Annals of Operations Research* 226 (2015), 397–416.

[30] Lydia T Liu, Nikhil Garg, and Christian Borgs. 2022. Strategic ranking. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2489–2518.

[31] Marcus Maher. 1999. Open source software: The success of an alternative intellectual property incentive paradigm. *Fordham Intell. Prop. Media & Ent. LJ* 10 (1999), 619.

[32] Nestor Maslej, Loredana Fattorini, Raymond Perrault, Vanessa Parli, Anka Reuel, Erik Brynjolfsson, , and et al. 2024. *The AI Index 2024 Annual Report*. Retrieved October 14, 2024 from https://aiindex.stanford.edu/wp-content/uploads/2024/05/HAI_AI-Index-Report-2024.pdf AI Index Steering Committee, Institute for Human-Centered AI, Stanford University, Stanford, CA.

[33] OpenAI, Josh Achiam, Sandhini Agarwal Steven Adler, Lama Ahmad, and et al. 2024. GPT-4 Technical Report. *arXiv preprint* (2024). https://arxiv.org/abs/2303.08774

[34] Juan Perdomo, Tijana Zrnic, Celestine Mendler-Dünner, and Moritz Hardt. 2020. Performative prediction. In *International Conference on Machine Learning*. PMLR, 7599–7609.

[35] Nicholas C Petruzzi and Maqbool Dada. 1999. Pricing and the newsvendor problem: A review with extensions. *Operations research* 47, 2 (1999), 183–194.

[36] Robert S Pindyck et al. 2018. *Microeconomics*.

[37] Cristina Rossi and Andrea Bonaccorsi. 2005. Intrinsic vs. extrinsic incentives in profit–oriented firms supplying Open Source products and services. *First Monday* (2005).

[38] Mehdi Seifbarghy, Khashayar Nouhi, and Amin Mahmoudi. 2015. Contract design in a supply chain considering price and quality dependent demand with customer segmentation. *International Journal of Production Economics* 167 (2015), 108–118.

[39] Nirvikar Singh and Xavier Vives. 1984. Price and quantity competition in a differentiated duopoly. *The Rand journal of economics* (1984), 546–554.

[40] Jianmin Tang. 2006. Competition and innovation behaviour. *Research policy* 35, 1 (2006), 68–82.

[41] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, and et al. 2023. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint* (2023). https://arxiv.org/abs/2302.13971

[42] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, and et al. 2022. Emergent Abilities of Large Language Models. *arXiv preprint* (Jan. 2022). https://arxiv.org/abs/2206.07682

[43] Fasheng Xu, Xiaoyu Wang, Wei Chen, and Karen Xie. 2024. The Economics of AI Foundation Models: Openness, Competition, and Governance. *SSRN* (Aug. 2024). doi:10.2139/ssrn.4999355

[44] Jianhong Yu and Shihua Ma. 2013. Impact of decision sequence of pricing and quality investment in decentralized assembly system. *Journal of Manufacturing Systems* 32, 4 (2013), 664–679.

A Proofs

A.1 Proof of Theorem 4.1

As $I = 1$, the utility functions of are:

$$U_S^{api} = (p - \gamma c)(a - bp)\alpha_1 - K_S^{api}(\alpha_1 - \alpha_0)^2,$$
$$U_F^{api} = (\gamma c - c)(a - bp)\alpha_1 - K_F\alpha_0^2$$

where:

$$K_S^{api} = K_{FT}, \quad K_F^{api} = K_{PRE} + K_G.$$

Step 1: Utility Maximization of S for a Fixed α_0 and γ

$$\frac{\partial U_S^{api}}{\partial p} = (a + bc\gamma - 2bp)\alpha_1 = 0$$
$$\Rightarrow \quad p^* = \frac{\frac{a}{b} + c\gamma}{2}.$$

$$\frac{\partial U_S^{api}}{\partial \alpha_1} = (p - \gamma c)(a - bp) - 2K_S^{api}(\alpha_1 - \alpha_0) = 0.$$

Substituting $p^* = \frac{\frac{a}{b}+c\gamma}{2}$, we get:

$$\alpha_1^* = \alpha_0 + \frac{b\left(\frac{a}{b} - c\gamma\right)^2}{8K_{FT}}.$$

The optimal choices for S are therefore:

$$p^* = \frac{\frac{a}{b} + c\gamma}{2}, \quad \alpha_1^* = \alpha_0 + \frac{b\left(\frac{a}{b} - c\gamma\right)^2}{8K_{FT}}.$$

Step 2: Utility Maximization of F Based on S's Response

Substituting $p^* = \frac{\frac{a}{b}+c\gamma}{2}$ and $\alpha_1^* = \alpha_0 + \frac{b(\frac{a}{b}-c\gamma)^2}{8K_{FT}}$, we have:

$$U_F^{api} = \frac{1}{2}bc^2(\gamma - 1)(\theta - \gamma)\left(\alpha_0 + \frac{bc^2(\theta - \gamma)^2}{8K_{FT}}\right) - K_F\alpha_0^2,$$
$$U_S^{api} = \frac{1}{4}bc^2(\theta - \gamma)^2\alpha_0 + \frac{b^2c^4}{64K_{FT}}(\theta - \gamma)^4,$$

, where $\theta = \frac{a}{bc}$

$$\frac{\partial U_F^{api}}{\partial \alpha_0} = \frac{1}{2}(\gamma - 1)c\,(a - bc\gamma) - 2K_F\alpha_0 = 0$$
$$\Rightarrow \quad \alpha_0^* = \frac{(\gamma - 1)c\,(a - bc\gamma)}{4K_F}.$$

$$0 = \frac{\partial U_F^{api}}{\partial \gamma} = \frac{1}{2}bc^2\left(\alpha_0^*(\theta + 1 - 2\gamma^*) + \frac{bc^2}{8K_{\mathrm{FT}}}(\theta - \gamma^*)^2(3 + \theta - 4\gamma^*)\right),$$
$$\Rightarrow \quad 0 = eq_a \cdot \gamma^2 + eq_b \cdot \gamma + eq_c,$$

where:

$$eq_a = 4(1 - \beta),$$
$$eq_b = 2\beta(3 + \theta) - 3 - 5\theta,$$
$$eq_c = \theta^2 + 3\theta - 2\beta(1 + \theta),$$
$$\beta = \frac{K_{FT}}{K_{PRE} + K_G}.$$

The discriminant δ is given by:

$$\delta = eq_b^2 - 4 \cdot eq_a \cdot eq_c.$$

Solving for the optimal γ using the quadratic formula, we find:

$$\gamma^* = \frac{-eq_b - \sqrt{\delta}}{2 \cdot eq_a} = \frac{5\theta + 3 - 2\beta(3 + \theta) - \sqrt{\delta}}{8(1 - \beta)}.$$

A.2 Proof of Theorem 4.3

From A.5, we have:

$$U_S^{self}(\alpha_1^*, p^*) = \frac{1}{4}bc^2(\theta - 1)^2\alpha + \frac{b^2c^4}{64(K_{FT} + K_G)}(\theta - 1)^4.$$

In this API case, $\alpha = m\alpha_0$, thus:

$$U_S^{self}(\alpha_1^*, p^*) = \frac{1}{4}bc^2(\theta - 1)^2 m\alpha_0 + \frac{b^2c^4}{64(K_{FT} + K_G)}(\theta - 1)^4.$$

From Section A.1, we have:

$$U_S^{api}(\alpha_1^*, p^*) = \frac{1}{4}bc^2(\theta - \gamma)^2\alpha_0 + \frac{b^2c^4}{64K_{FT}}(\theta - \gamma)^4$$

The goal is to solve:

$$\alpha_0^*, \gamma^* = \arg\max_{\alpha_0, \gamma} U_F^{api}(\alpha_1^*, p^*),$$

subject to:

$$U_S^{self}(\alpha_1^*, p^*) \le U_S^{api}(\alpha_1^*, p^*). \tag{25}$$

Define the Lagrangian with multiplier λ:

$$\mathcal{L} = U_F^{api}(\alpha_1^*, p^*) + \lambda\left(U_S^{api}(\alpha_1^*, p^*) - U_S^{self}(\alpha_1^*, p^*)\right).$$

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \alpha_0} = 0, \\ \frac{\partial \mathcal{L}}{\partial \gamma} = 0, \\ \frac{\partial \mathcal{L}}{\partial \lambda} = 0 \end{cases}$$

Thus,

$$\begin{cases} \frac{\partial U_F^{api}}{\partial \alpha_0} + \lambda(\frac{\partial U_S^{api}}{\partial \alpha_0} - \frac{\partial U_S^{self}}{\partial \alpha_0}) = 0, \\ \frac{\partial U_F^{api}}{\partial \gamma} + \lambda(\frac{\partial U_S^{api}}{\partial \gamma} - \frac{\partial U_S^{self}}{\partial \gamma}) = 0, \\ U_S^{self}(\alpha_1^*, p^*) = U_S^{api}(\alpha_1^*, p^*) \end{cases}$$

Thus,

$$\begin{cases} \frac{\partial U_F^{api}}{\partial \alpha_0}(\frac{\partial U_S^{api}}{\partial \gamma} - \frac{\partial U_S^{self}}{\partial \gamma}) = \frac{\partial U_F^{api}}{\partial \gamma}(\frac{\partial U_S^{api}}{\partial \alpha_0} - \frac{\partial U_S^{self}}{\partial \alpha_0}), \\ U_S^{self}(\alpha_1^*, p^*) = U_S^{api}(\alpha_1^*, p^*) \end{cases}$$

The partial derivatives of U_S^{api} and U_S^{self} are as follows:

$$\begin{cases} \frac{\partial U_S^{api}}{\partial \alpha_0} &= \frac{1}{4}bc^2(\theta-\gamma)^2, \\ \frac{\partial U_S^{api}}{\partial \gamma} &= -\frac{b^2c^4}{16K_{\mathrm{FT}}}(\theta-\gamma)^3 - \frac{1}{2}bc^2(\theta-\gamma)\alpha_0, \\ \frac{\partial U_S^{self}}{\partial \alpha_0} &= \frac{1}{4}mbc^2(\theta-1)^2, \\ \frac{\partial U_S^{self}}{\partial \gamma} &= 0, \\ \frac{\partial U_F^{api}}{\partial \alpha_0} &= \frac{1}{2}(\gamma-1)c\,(a-bc\gamma) - 2K_F\alpha_0, \\ \frac{\partial U_F^{api}}{\partial \gamma} &= \frac{1}{2}bc^2\left(\alpha_0^*(\theta+1-2\gamma^*) + \frac{bc^2}{8K_{\mathrm{FT}}}(\theta-\gamma^*)^2(3+\theta-4\gamma^*)\right) \end{cases}$$

The optimal values α_0^* and γ^* satisfy:

$$\begin{cases} \left(16(\theta-\gamma^*)^2 - m(\theta-1)^2\right)\alpha_0^* = \left(\frac{(\theta-1)^4}{K_{\mathrm{FT}}+K_{\mathrm{G}}} - \frac{(\theta-\gamma^*)^4}{K_{\mathrm{FT}}}\right)bc^2, \\ \frac{2\left(bc^2(\gamma^*-1)(\theta-\gamma^*)-4(K_{\mathrm{PRE}}+K_{\mathrm{G}})\alpha_0^*\right)}{\alpha_0^*(\theta+1-2\gamma^*)+\frac{bc^2}{8K_{\mathrm{FT}}}(\theta-\gamma^*)^2(3+\theta-4\gamma^*)} = \frac{-bc^2\left((\theta-\gamma^*)^2-m(\theta-1)^2\right)}{\frac{bc^2}{8K_{\mathrm{FT}}}(\theta-\gamma^*)^3+(\theta-\gamma)\alpha_0^*}, \end{cases}$$

where $\theta = \frac{a}{bc}$.

A.3 Proof of Theorem 4.4

To prove Theorem 4.4, we need to find α_0^*, γ^* satisfying:

$$\begin{cases} U_S^{self}(\alpha_1^*, p^*) \le U_S^{api}(\alpha_1^*, p^*), \\ U_F^{api} \ge 0, \end{cases}$$

From A.1 and A.2, we know it is equivalent to find α_0^*, γ^* satisfying:

$$\begin{cases} \left(16(\theta-\gamma^*)^2 - m(\theta-1)^2\right)\alpha_0^* > \left(\frac{(\theta-1)^4}{K_{\mathrm{FT}}+K_{\mathrm{G}}} - \frac{(\theta-\gamma^*)^4}{K_{\mathrm{FT}}}\right)bc^2, \\ \frac{1}{2}bc^2(\gamma-1)(\theta-\gamma)\left(\alpha_0 + \frac{bc^2(\theta-\gamma)^2}{8K_{FT}}\right) - K_F\alpha_0^2 > 0 \end{cases}$$

, where $\theta = \frac{a}{bc}$.

Also, reasonable solution should satisfy $\alpha_0^* > 0$ and $1 < \gamma^* < \theta$.

$U_F^{api}(\alpha_0)$ is a quadratic function, opening downward, with $U_F^{api}(\alpha_0 = 0) > 0$ and axis of symmetry given by:

$$\frac{bc^2}{4K_F}(\gamma-1)(\theta-\gamma) > 0,$$

Define $\alpha_0^1 = \frac{bc^2}{4K_F}(\gamma-1)(\theta-\gamma)$.

A.3.1 Case 1: $m \ge 1$.

$$(\theta-\gamma)^2 - m(\theta-1)^2 < 0.$$

Thus, we must have:

$$\frac{(\theta-1)^4}{K_{\mathrm{FT}}+K_{\mathrm{G}}} < \frac{(\theta-\gamma^*)^4}{K_{\mathrm{FT}}} \Rightarrow (\theta-\gamma)^4 > \left(\frac{K_{\mathrm{FT}}}{K_{\mathrm{FT}}+K_{\mathrm{G}}}\right)^{1/4}\cdot(\theta-1).$$

We can always find γ^* to satisfy this condition.

Also,

$$\alpha_0 < \frac{\left(\frac{(\theta-1)^4}{K_{\mathrm{FT}}+K_{\mathrm{G}}} - \frac{(\theta-\gamma^*)^4}{K_{\mathrm{FT}}}\right)bc^2}{16(\theta-\gamma^*)^2 - m(\theta-1)^2} = \alpha_0^{\mathrm{cut}}.$$

Thus, an example solution:

$$\begin{cases} \gamma^* = \theta - \frac{1}{2}\left(\theta - 1 + \left(\frac{K_{\mathrm{FT}}}{K_{\mathrm{FT}}+K_{\mathrm{G}}}\right)^{1/4}\cdot(\theta-1)\right), \\ \alpha_0^* = \min\{\alpha_0^1, \alpha_0^{\mathrm{cut}}\}. \end{cases}$$

A.3.2 Case 2: $0 < m < 1$. .

If $0 \le m^2 < \frac{K_{\mathrm{FT}}}{K_{\mathrm{FT}}+K_{\mathrm{G}}} < 1$ (when hardware cost is relatively low), let

$$\gamma^* \in \left(1, \theta - \left(\frac{K_{\mathrm{FT}}}{K_{\mathrm{FT}}+K_{\mathrm{G}}}\right)^{1/4}\cdot(\theta-1)\right),$$

which ensures:

$$\begin{cases} (\theta-\gamma)^2 - m(\theta-1)^2 > 0, \\ \frac{(\theta-1)^4}{K_{\mathrm{FT}}+K_{\mathrm{G}}} - \frac{(\theta-\gamma^*)^4}{K_{\mathrm{FT}}} < 0. \end{cases}$$

An example solution:

$$\begin{cases} \gamma^* = \theta - \frac{1}{2}\left(\theta - 1 + \left(\frac{K_{\mathrm{FT}}}{K_{\mathrm{FT}}+K_{\mathrm{G}}}\right)^{1/4}\cdot(\theta-1)\right), \\ \alpha_0^* = \alpha_0^1. \end{cases}$$

If $\frac{K_{\mathrm{FT}}}{K_{\mathrm{FT}}+K_{\mathrm{G}}} \le m^2 < 1$ (when hardware cost is relatively high), let

$$\gamma^* \in \left(1, \theta - m^{1/2}\cdot(\theta-1)\right),$$

which ensures:

$$\begin{cases} (\theta-\gamma)^2 - m(\theta-1)^2 > 0, \\ \frac{(\theta-1)^4}{K_{\mathrm{FT}}+K_{\mathrm{G}}} - \frac{(\theta-\gamma^*)^4}{K_{\mathrm{FT}}} < 0. \end{cases}$$

An example solution:

$$\begin{cases} \gamma^* = \theta - \frac{1}{2}\left(\theta - 1 + m^{1/2}\cdot(\theta-1)\right), \\ \alpha_0^* = \alpha_0^1. \end{cases}$$

A.4 Proof of Theorem 4.6

From A.5, with proactive open-source community, we have:

$$U_S^{self}(\alpha_1^*, p^*) = \frac{1}{4}bc^2(\theta-1)^2\alpha + \frac{b^2c^4}{64(K_{FT}+K_G)}(\theta-1)^4.$$

Same as A.2, the goal is to solve:

$$\alpha_0^*, \gamma^* = \arg\max_{\alpha_0,\gamma} U_F^{api}(\alpha_1^*, p^*),$$

subject to:

$$U_S^{self}(\alpha_1^*, p^*) \le U_S^{api}(\alpha_1^*, p^*).$$

Similar as A.2, we can solve the problem with KKT. The only difference is that $\frac{\partial U_S^{self}}{\partial \alpha_0} = 0$

A.5 Proof of Theorem 4.7

When $I = 0$ and with a proactive open-source community, the utility functions for the deployer S and the developer F are given by:

$$U_S^{self} = (p-c)(a-bp)\alpha_1 - K_S^{self}(\alpha_1-\alpha)^2,$$
$$U_F^{self} = 0,$$

where $K_S^{self} = K_{FT} + K_G$.

Step 1: Solving for Optimal p^* and α_1^* for S

$$\frac{\partial U_S^{self}}{\partial p} = (a-2bp+bc)\alpha_1 = 0,$$
$$\Rightarrow \quad p^* = \frac{1}{2}\left(\frac{a}{b}+c\right).$$
$$\frac{\partial U_S^{self}}{\partial \alpha_1} = (p-c)(a-bp) - 2K_S^{self}(\alpha_1-\alpha) = 0.$$

Substituting $p^* = \frac{1}{2}\left(\frac{a}{b}+c\right)$,

$$\alpha_1^* = \alpha + \frac{b\left(\frac{a}{b}-c\right)^2}{8(K_{FT}+K_G)}.$$

Thus, the optimal utility for S in a self-hosting setup with proactive O is:

$$U_S^{self}(\alpha_1^*, p^*) = \frac{1}{4}bc^2(\theta-1)^2\alpha + \frac{b^2c^4}{64(K_{FT}+K_G)}(\theta-1)^4$$

, where $\theta = \frac{a}{bc}$.

Step 2: Confirming Developer's Choice (Setting $I = 0$)

Since $U_F^{self} = 0$ when S chooses self-hosting, the developer F gains no utility. This setup implies that the optimal strategy for the developer is to exit the market, yielding:

$$\gamma^* = 1, \quad \alpha_0^* = 0, \quad I^* = 0.$$

A.6 Proof of Theorem 4.8

Contrary A.3, we need to illustrate: when α is high, there is no solution of α_0^*, γ^* satisfying:

$$\begin{cases} 16(\theta-\gamma^*)^2\alpha_0^* - 16(\theta-1)^2\alpha > \left(\frac{(\theta-1)^4}{K_{\mathrm{FT}}+K_{\mathrm{G}}} - \frac{(\theta-\gamma^*)^4}{K_{\mathrm{FT}}}\right)bc^2, \\ U_F^{api} = \frac{1}{2}bc^2(\gamma-1)(\theta-\gamma)\left(\alpha_0 + \frac{bc^2(\theta-\gamma)^2}{8K_{FT}}\right) - K_F\alpha_0^2 > 0 \end{cases}$$

, where $\theta = \frac{a}{bc}$.

It is equivalent to:

$$\begin{cases} 16(\theta-\gamma^*)^2\alpha_0^* > \left(\frac{(\theta-1)^4}{K_{\mathrm{FT}}+K_{\mathrm{G}}} - \frac{(\theta-\gamma^*)^4}{K_{\mathrm{FT}}}\right)bc^2 + 16(\theta-1)^2\alpha, \\ \frac{1}{2}bc^2(\gamma-1)(\theta-\gamma)\left(\alpha_0 + \frac{bc^2(\theta-\gamma)^2}{8K_{FT}}\right) - K_F\alpha_0^2 > 0 \end{cases}$$

Denote $R = \left(\frac{(\theta-1)^4}{K_{\mathrm{FT}}+K_{\mathrm{G}}} - \frac{(\theta-\gamma^*)^4}{K_{\mathrm{FT}}}\right)bc^2 + 16(\theta-1)^2\alpha, L = 16(\theta-\gamma^*)^2$

Thus,

$$\begin{cases} L\alpha_0^* > R, \\ \frac{1}{2}bc^2(\gamma-1)(\theta-\gamma)\left(\alpha_0 + \frac{bc^2(\theta-\gamma)^2}{8K_{FT}}\right) - K_F\alpha_0^2 > 0 \end{cases}$$

Lets $\alpha > \frac{(\theta-1)^4}{K_{\mathrm{FT}}}bc^2$, thus,

$$\begin{cases} \alpha_0^* > \frac{R}{L} > 0., \\ \frac{1}{2}bc^2(\gamma-1)(\theta-\gamma)\left(\alpha_0 + \frac{bc^2(\theta-\gamma)^2}{8K_{FT}}\right) - K_F\alpha_0^2 > 0 \end{cases}$$

As $U_F^{api}(\alpha_0)$ is a quadratic function, opening downward, with $U_F^{api}(\alpha_0 = 0) > 0$ and axis of symmetry given by:

$$\frac{bc^2}{4K_F}(\gamma-1)(\theta-\gamma) > 0,$$

We only need to substitute $\alpha_0^* = \frac{R}{L}$ in $U_F^{api}(\alpha_0 *)$ and show $U_F^{api}(\alpha_0) < 0$ when α is high.

$$U_F^{api}\left(\alpha_0^* = \frac{R}{L}\right) = -K_F\left(\frac{R}{L}\right)^2 + \frac{bc^2}{2}(\gamma-1)(\theta-1)\frac{R}{L} + \frac{b^2c^4}{16K_{\mathrm{FT}}}(\theta-\gamma)^3(\gamma-1)$$
$$= (-K_FR^2 + \frac{bc^2}{2}(\gamma-1)(\theta-1)RL + \frac{b^2c^4}{16K_{\mathrm{FT}}}(\theta-\gamma)^3(\gamma-1)L^2)/L^2.$$

as $\begin{cases} R \geq R_{\min} = R(\gamma^*=1) = \left(\frac{(\theta-1)^4}{K_{\mathrm{FT}}+K_{\mathrm{G}}} - \frac{(\theta-1)^4}{K_{\mathrm{FT}}}\right)bc^2 + 16(\theta-1)^2\alpha, \\ R \leq R - bc^2\frac{(\theta-1)^4}{K_{\mathrm{FT}}+K_{\mathrm{G}}} = -\frac{(\theta-\gamma^*)^4}{K_{\mathrm{FT}}}bc^2 + 16(\theta-1)^2\alpha, \\ L \leq L_{\max} = L(\gamma^*=1) = 16(\theta-1)^2, \\ \theta-\gamma \leq \theta-1. \end{cases}$

Thus,

$$L^2U_F^{api}\left(\alpha_0^* = \frac{R}{L}\right) \leq -K_F\left(\left(\frac{(\theta-1)^4}{K_{\mathrm{FT}}+K_{\mathrm{G}}} - \frac{(\theta-\gamma^*)^4}{K_{\mathrm{FT}}}\right)bc^2 + 16(\theta-1)^2\alpha\right)^2$$
$$+\frac{bc^2}{2}(\gamma-1)(\theta-1)\left(-\frac{(\theta-\gamma^*)^4}{K_{\mathrm{FT}}}bc^2 + 16(\theta-1)^2\alpha\right)16(\theta-1)^2$$
$$+\frac{b^2c^4}{16K_{\mathrm{FT}}}(\theta-1)^3(\gamma-1)16^2(\theta-1)^4.$$

Obviously, $L^2U_F^{api}\left(\alpha_0^* = \frac{R}{L}\right)$ is a quadratic function of α, opening downward. Thus, when α is high enough to let $L^2U_F^{api}\left(\alpha_0^* = \frac{R}{L}\right) < 0$, we cannot find a solution of α_0^*, γ^* to make $U_F^{api} > 0$, which the equilibrium falls into self-hosting. The cut-off α^H can be the right root of $L^2U_F^{api}\left(\alpha_0^* = \frac{R}{L}\right) = 0$

A.7 Proof of Theorem 5.2

From A.5, we know $\alpha_0'^*$ and γ'^* is the solution of

$$\begin{cases} \frac{\partial U_F^{api}}{\partial \alpha_0} + \lambda(\frac{\partial U_S^{api}}{\partial \alpha_0} - \frac{\partial U_S^{self}}{\partial \alpha_0}) = 0, \\ \frac{\partial U_F^{api}}{\partial \gamma} + \lambda \frac{\partial U_S^{api}}{\partial \gamma} = 0, \\ U_S^{self} = U_S^{api} \end{cases}$$

However, α_0^* and γ^* is the solution of:

$$\begin{cases} \frac{\partial U_F^{api}}{\partial \alpha_0} = 0, \\ \frac{\partial U_F^{api}}{\partial \gamma} = 0, \end{cases}$$

Also, we know that $U_S^{self}(\alpha_0^*, \gamma^*) > U_S^{api}(\alpha_0^*, \gamma^*)$. Else it falls into a API-dominant solution.

Part One: if

$$\frac{\partial U_S^{api}}{\partial \alpha_0} < \frac{\partial U_S^{self}}{\partial \alpha_0} and \frac{\partial^2 U_F}{\partial \gamma \partial \alpha_0} > 0. \tag{26}$$

We know

$$\begin{cases} \frac{\partial U_F^{api}}{\partial \alpha_0}(\gamma'^*, \alpha_0'^*) > 0, \\ \frac{\partial U_F^{api}}{\partial \gamma}(\gamma'^*, \alpha_0'^*) > 0 \end{cases}$$

We discuss the value of $\alpha_0'^*$ and γ'^*:

(1) $\alpha_0'^* > \alpha_0^*, \gamma'^* > \gamma^*$: unreasonable. As $U_S^{self}(\alpha_0'^*, \gamma'^*) - U_S^{api}(\alpha_0'^*, \gamma'^*) > U_S^{self}(\alpha_0'^*, \gamma^*) - U_S^{api}(\alpha_0'^*, \gamma^*) > U_S^{self}(\alpha_0^*, \gamma^*) - U_S^{api}(\alpha_0^*, \gamma^*) > 0$

(2) $\alpha_0'^* > \alpha_0^*, \gamma'^* < \gamma^*$: unreasonable. As $\frac{\partial U_F^{api}}{\partial \alpha_0}(\gamma'^*, \alpha_0'^*) < \frac{\partial U_F^{api}}{\partial \alpha_0}(\gamma^*, \alpha_0'^*) < \frac{\partial U_F^{api}}{\partial \alpha_0}(\gamma^*, \alpha_0^*) = 0$

(3) $\alpha_0'^* < \alpha_0^*, \gamma'^* > \gamma^*$: unreasonable. As $\frac{\partial U_F^{api}}{\partial \gamma}(\gamma'^*, \alpha_0'^*) < \frac{\partial U_F^{api}}{\partial \gamma}(\gamma^*, \alpha_0'^*) < \frac{\partial U_F^{api}}{\partial \gamma}(\gamma^*, \alpha_0^*) = 0$

Thus, $\alpha_0'^* < \alpha_0^*, \gamma'^* < \gamma^*$ is the only feasible solution.

Part Two: if

$$\frac{\partial U_S^{api}}{\partial \alpha_0} > \frac{\partial U_S^{self}}{\partial \alpha_0} and \frac{\partial^2 U_F}{\partial \gamma \partial \alpha_0} < 0. \tag{27}$$

We know

$$\begin{cases} \frac{\partial U_F^{api}}{\partial \alpha_0}(\gamma'^*, \alpha_0'^*) < 0, \\ \frac{\partial U_F^{api}}{\partial \gamma}(\gamma'^*, \alpha_0'^*) > 0 \end{cases}$$

We discuss the value of $\alpha_0'^*$ and γ'^*:

(1) $\alpha_0'^* > \alpha_0^*, \gamma'^* > \gamma^*$: unreasonable. As $\frac{\partial U_F^{api}}{\partial \gamma}(\gamma'^*, \alpha_0'^*) < \frac{\partial U_F^{api}}{\partial \gamma}(\gamma^*, \alpha_0'^*) < \frac{\partial U_F^{api}}{\partial \gamma}(\gamma^*, \alpha_0^*) = 0$

(2) $\alpha_0'^* < \alpha_0^*, \gamma'^* < \gamma^*$: unreasonable. As $\frac{\partial U_F^{api}}{\partial \alpha_0}(\gamma'^*, \alpha_0'^*) > \frac{\partial U_F^{api}}{\partial \alpha_0}(\gamma^*, \alpha_0'^*) > \frac{\partial U_F^{api}}{\partial \alpha_0}(\gamma^*, \alpha_0^*) = 0$

(3) $\alpha_0'^* < \alpha_0^*, \gamma'^* > \gamma^*$: unreasonable. As $U_S^{self}(\alpha_0'^*, \gamma'^*) - U_S^{api}(\alpha_0'^*, \gamma'^*) > U_S^{self}(\alpha_0'^*, \gamma^*) - U_S^{api}(\alpha_0'^*, \gamma^*) > U_S^{self}(\alpha_0^*, \gamma^*) - U_S^{api}(\alpha_0^*, \gamma^*) > 0$

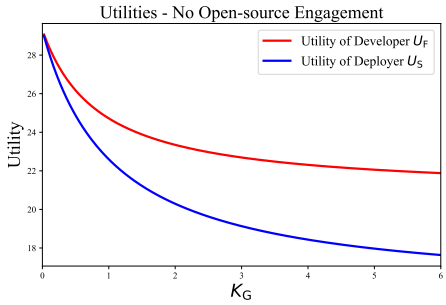
Thus, $\alpha_0'^* > \alpha_0^*, \gamma'^* < \gamma^*$ is the only feasible solution.

A.8 Proof of Theorem 5.3

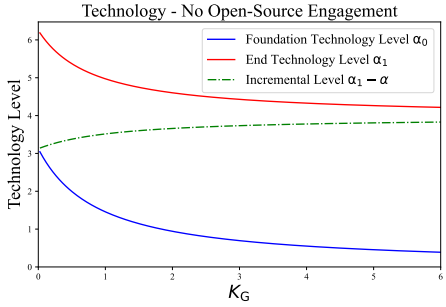
The proof is the same as the Part Two of A.7

B Figures

B.1 Equilibrium Without Open-Source Engagement



(a) Utility



(b) Technology

Figure 4: Equilibrium Outcomes without Open-Source Engagement ($a = 8, b = 1, c = 0.5, K_{FT} = K_{PRE} = 1$)

C Robustness Check of Numerical Results

We tested 256 combinations of parameter values, as detailed in Table 1. The condition $a > b \cdot c$ was enforced to ensure positive utility. The results remain robust across all tested combinations.

Table 1: Parameter Settings for Robustness Check

Parameter	Values
a	8
b	1, 1.5, 2, 2.5
c	$1, 1 + \frac{(a/b-1)}{8}, 1 + \frac{(a/b-1)}{4}, 1 + \frac{(a/b-1)}{2}$
K_{FT}	1, 3, 6, 9
K_{PRE}	1, 3, 6, 9

D Notations

Table 2: Notations

Symbols	Meanings
F	closed-source foundation technology developer
S	Domain-specific deployer
O	Open-source community
U	End user
a	Total potential market demand
b	Price sensitivity of end user
c	Unit operation cost of foundation technology
γ	Price multiplier of API
α_0	Closed-source foundation technology performance
α_0^{soc}	Foundation technology performance at social level
α	Open-source foundational technology performance
α_1	End technology performance
α_1^{soc}	End technology performance at social level
m	Relative performance of open-source to closed-source foundational technology
K_F	Cost factor for developing foundational technology
K_S^{api}	Cost factor for adapting technology in API scenario
K_S^{self}	Cost factor for adapting technology in self-hosting scenario
K_{GPU}	Hardware cost parameter
K_{PRE}	Non-hardware cost parameter for developing foundation technology
K_{FT}	Non-hardware cost factor for adapting technology
p	Price of domain-specific technology
D	Actual demand in the end market
U_S	Utility of deployer S
U_F	Utility of developer F