



Carnegie Mellon University

Doctoral Dissertation

**Human-AI Interaction in the Era of
Large Language Models (LLMs)**

Behnam Mohammadi

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Marketing*

Tepper School of Business
Carnegie Mellon University

April 2025

Copyright © Behnam Mohammadi 2025
All Rights Reserved.

Dissertation Committee

Tim Derdenger (Chair)

Carnegie Mellon University
Tepper School of Business

Alan Montgomery

Carnegie Mellon University
Tepper School of Business

Nikhil Malik

University of Southern California
Marshall School of Business

Kannan Srinivasan

Carnegie Mellon University
Tepper School of Business

Xiao Liu

New York University
Stern School of Business

Zoey Jiang (outside reader)

Carnegie Mellon University
Tepper School of Business

This page was accidentally left blank.

Acknowledgements

My Ph.D. journey was far from conventional. I often found myself drawn to research topics with scarce or nonexistent literature. These untrodden roads carried inherent uncertainties as success was never guaranteed and the way forward not always clear. Fortunately, some of these ventures proved fruitful, yielding insights worthy of documentation. This, however, would have been impossible without the guidance and support of numerous individuals who accompanied me throughout my atypical doctoral experience. I am indebted to them for helping me navigate the risks of exploring uncharted territories and ultimately arriving at this point.

First and foremost, my sincere appreciation goes to my committee chair, Tim Derdenger, who played a critical role in my development as a researcher by guiding both of my summer papers, one of which was eventually published in Marketing Science. Beyond our academic work, Tim significantly supported me during my job market experience. He dedicated substantial time and effort to crafting an exceptional recommendation letter and offered insightful advice that helped secure a positive outcome. Without his steadfast support, my transition to faculty life would not have been possible. Tim, thank you for your mentorship and for helping me establish my future in academia.

I am equally grateful to Alan Montgomery, whose influence began from the very first moment he interviewed me for the Ph.D. program and continued through many interactions ever since. Alan's impression extends far beyond the two courses on Bayesian methods I had the privilege of taking with him. As head of the Ph.D. program, his balanced and mature counsel have guided me through several crossroads. Alan's support during my job search process has also left a lasting impact on my career trajectory. Thank you, Alan, for opening my eyes to the broader landscape of academia and for your profound impact on both my professional and personal growth.

Special thanks go to Nikhil, my co-author on my first paper, whose collaboration has been an honor and a true learning experience. His meticulous approach to crafting research

worthy of our field's most prestigious journals has instilled in me standards of excellence that will inform my scholarly pursuits throughout my career. His intelligence, attention to intuitive understanding of subjects, and generosity in sharing his experience made our collaboration especially rewarding. Nikhil's support on the job market was also greatly beneficial, and I genuinely look forward to crossing paths again professionally.

I deeply appreciate my friends, whose presence has transformed what could have been an isolating journey into one filled with memorable moments and shared experiences: Martin Michelini and Zahra Ebrahimi, Samuel Levy and Nur Yıldırım, Thomas Devine and Emily Temmer, Jaepil Lee and Joohyun Lee, Jane Pyo, Jenny Oh, Jisoo Huyn, Pim Assavabhokhin, Kevin Mott and Sae-Seul Park, Julie Wang, Angela Xiao, Flora Feng, Jaymo Kim, Kerry Zhang, Yuan Yuan, Qiaochu Wang and Lavender Yang, David Li, Arnav Sood, Neha, Anthony Karahalios, and my job market buddy Poet Larsen. Though geographical separation is inevitable, I hope we continue to share life's significant milestones regardless of distance.

Thanks are also due to Laila Lee and Lawrence Rapp, whose blend of professionalism coupled with warm, approachable demeanor, made their exceptional administrative support invaluable. Your prompt help has been crucial in successfully navigating the many intricacies of the Ph.D. program.

My deepest gratitude travels 7,350 miles to my family in Iran. To my mother Parvin, my father Behrouz, and my sister Behnaz, who have believed in me since childhood and nurtured my intellectual curiosity while accepting my quirks! Despite six years of physical separation and occasional homesickness, your support has never faltered. Mother, during my darkest hours, your strength helped me rise again with renewed determination, and for that, I am eternally grateful.

To Naser Sayyadi, my closest friend of twenty-five years, our friendship transcends cities, countries, and continents. I am grateful we learnt English together, and I cherished our countless gaming sessions of Age of Mythology, DotA, and Counter Strike. Those games became a canvas through which we expressed our worldviews and approaches to challenges. Though I missed witnessing precious moments like the birth of your child, Tarannom, and her first steps, I remain hopeful that our paths will converge again.

Finally, to Elisabeth Estes, my partner and closest companion, whom I met during my first year in the U.S. amidst an unprecedented pandemic: meeting you on that beautiful summer day and growing together has truly been the highlight of my Ph.D. experience. Your love, kindness, and patience have sustained me through both triumphs and setbacks, and I cannot imagine this journey without you. I am incredibly lucky that our paths crossed when they did, and even luckier to have you by my side as we write the next chapter of our lives together in Dallas, Texas.

Preface

When I entered the Ph.D. program, conversing with an AI assistant was barely possible. Now, as I complete this thesis, AI systems not only engage in sophisticated dialogue but excel at tasks once thought resilient to automation—programming, creative writing, image generation, and complex reasoning—often surpassing average human performance. The seismic impact of ChatGPT continues to reverberate throughout society, demanding that we adapt rather than resist this new paradigm. The modern world of AI brings with it unprecedented, time-sensitive questions demanding prompt answers (pun intended!). My dissertation addresses four such questions: Chapter 1 reveals the intricate nuances of AI regulation, demonstrating how one-size-fits-all policies like mandating full explainability may sometimes harm rather than help consumers. Chapter 2 develops an explainability tool for large language models (LLMs) that exposes fundamental differences between human cognition and LLMs, challenging the simplistic view that models which “talk like humans” can effectively simulate consumer behavior. Chapter 3 examines AI creativity. After all, what good is intelligence without the ability to create something new? But creativity does not come without cost, as this study identifies a crucial tradeoff between safety and creative output that impacts which models best serve different purposes. Finally, Chapter 4 addresses the emerging challenge of coordinating networks of AI agents working in tandem, introducing the Pel programming language as a solution for orchestrating such systems.

Together, these studies provide theoretical insights and practical frameworks for navigating the rapidly evolving AI landscape by introducing ideas, concepts, and methodologies that transcend specific AI architectures or models. Alan once advised me to pursue research that would remain relevant for years to come. I am hopeful this work answers that challenge.

Behnam Mohammadi

April 2025

Abstract

The first chapter, joint work with Nikhil Malik, Tim Derdenger, and Kannan Srinivasan, challenges conventional wisdom regarding eXplainable AI (XAI) regulations such as GDPR. Through a game-theoretic model examining XAI methods and levels in a duopoly market with heterogeneous customer preferences, we demonstrate that partial explanations can emerge as an equilibrium in unregulated settings. Importantly, we identify conditions where mandating full explanations through regulation may actually harm consumer surplus rather than enhance it. This finding holds across various policy levers (strict, self-regulating, and lower bound), regardless of firms’ choice of XAI methods and policy objectives including welfare maximization, consumer surplus, and average XAI depth. Our comparative analysis reveals that while strict XAI policies ensure uniform explanation depth, they potentially limit firms’ capacity for differentiation and innovation. Conversely, unregulated XAI, while offering maximum flexibility, may fail to guarantee minimum explanation depth for all consumers. The introduction of flexible approaches—self-regulating XAI and lower-bounded XAI—results in higher consumer welfare than either unregulated or full XAI policies. This research urges policymakers to consider a more nuanced approach when crafting XAI regulations, as a one-size-fits-all policy across all markets, particularly one mandating full explanation, may not yield the desired outcomes. For firms operating in these markets, the optimal strategy may not be to provide full explanations, as partial explanations can emerge as equilibrium strategies that better serve their competitive positioning while still addressing consumer needs.

The second chapter addresses the growing use of LLMs as simulated consumers in marketing research. I develop a novel approach based on Shapley values from cooperative game theory to interpret LLM behavior and quantify the relative contribution of prompt components to model outputs. Through applications in discrete choice experiments and cognitive bias investigations, I uncover what I term the “token noise” effect—a phenomenon where LLM decisions are disproportionately influenced by tokens providing minimal informative content (such as empty lines in a questionnaire!). This finding provides a

theoretical foundation for understanding how LLMs process information and make decisions, revealing fundamental differences from human cognition that must be accounted for in marketing research. For marketers employing LLMs for consumer simulation, this raises significant concerns about the validity of using LLMs as proxies for human subjects and necessitates rigorous validation procedures when using LLMs for preference elicitation or behavior prediction. The proposed Shapley value method offers practitioners a model-agnostic approach for optimizing prompts and mitigating apparent cognitive biases in LLM responses.

The third chapter investigates the unintended consequences of AI alignment techniques on the creative capabilities of language models. Through a series of experiments with the Llama model family (created by Meta/Facebook), I demonstrate that alignment methods like Reinforcement Learning from Human Feedback (RLHF), while reducing bias and harmful outputs, significantly diminish syntactic and semantic diversity. My findings reveal that aligned models exhibit lower entropy in token predictions, form distinct clusters in embedding space, and gravitate toward “attractor states”, indicating limited output diversity. This contributes to our theoretical understanding of AI creativity by conceptualizing the relationship between alignment and creativity as a fundamental trade-off rather than a technical limitation. Marketing teams must strategically balance the benefits of AI safety alignment with creative performance when selecting language models for content generation tasks. Different models may be optimal for different marketing functions—aligned models for customer-facing interactions where consistency and brand safety are paramount, and base models for ideation tasks that benefit from novelty and creativity, such as ad copywriting and customer persona development.

The fourth chapter steps beyond individual models to examine networks of AI agents that work together to accomplish complex goals, such as automating various functions in a business (e.g., customer support, SEO, refunds, etc.). This introduces a new challenge: not just how we build these agents, but how we coordinate them. To address this, I introduce Pel, a programming language I developed from scratch specifically for orchestrating AI agents. Pel offers an elegant, principled framework for multi-agent AI systems, addressing limitations in current methods of controlling LLMs through a syntactically simple yet semantically rich

platform for expressing complex actions, control flow, and inter-agent communication. Its design emphasizes minimal grammar suitable for constrained LLM generation, powerful composition mechanisms, and built-in support for natural language conditions. This advances programming language theory through the development of a domain-specific language (DSL) optimized for AI agent control, proposing a new paradigm for human-AI interaction that incorporates the unique capabilities and limitations of language models. From a managerial perspective, Pel provides marketing technology teams with a specialized tool for building sophisticated marketing automation systems powered by LLMs, to be used in customer engagement and support, content personalization, and multi-channel campaign management.

“Think Different.”

Table of Contents

Chapter 1: Regulating eXplainable AI (XAI)	1
May Harm Consumers	
Chapter 2: Explaining Large Language Models Decisions	31
Using Shapley Values	
Chapter 3: Creativity Has Left the Chat:	61
The Price of Debiasing Language Models	
Chapter 4: Pel, A Programming Language for	98
Orchestrating AI Agents	

This page was unintentionally left blank.

Regulating eXplainable AI (XAI) May Harm Consumers

Behnam Mohammadi*, Nikhil Malik[†], Tim Derdenger[‡], Kannan Srinivasan[§]

June 2024

Abstract

The most recent AI algorithms lack interpretability. eXplainable AI (XAI) aims to address this by explaining AI decisions to customers. Although it is commonly believed that the requirement of fully transparent XAI enhances consumer surplus, our paper challenges this view. We present a game-theoretic model where a policymaker maximizes consumer surplus in a duopoly market with heterogeneous customer preferences. Our model integrates AI accuracy, explanation depth, and method. We find that partial explanations can be an equilibrium in an unregulated setting. Furthermore, we identify scenarios where customers' and firms' desires for full explanation are misaligned. In these cases, regulating full explanations may not be socially optimal and could worsen the outcomes for firms and consumers. Flexible XAI policies outperform both full transparency and unregulated extremes.

Keywords: Machine Learning, Explainable AI, Economics of AI, Regulation, Fairness

*Tepper School of Business, Carnegie Mellon University Email: behnamm@andrew.cmu.edu

[†]Marshall School of Business, University of Southern California Email: maliknik@marshall.usc.edu

[‡]Tepper School of Business, Carnegie Mellon University Email: derdenge@andrew.cmu.edu

[§]Tepper School of Business, Carnegie Mellon University Email: kannans@andrew.cmu.edu

1 Introduction

In recent years, the adoption of Artificial Intelligence (AI) models for decision-making has increased. Gartner identifies AI engineering as one of the top 12 strategic technology trends of 2022¹ and the International Data Corporation (IDC) forecasts global spending on AI systems to exceed \$204 billion by 2025.² AI has been used in various forms in marketing over the years. Segmentation (Valls et al., 2018, Dekimpe, 2020, Pitt et al., 2020) and targeting (Neumann et al., 2019, Drew et al., 2001, Ascarza, 2018) are two prominent applications that have benefitted from AI. AI could also help brands develop successful positioning statements and compelling slogans (Huang and Rust, 2021).

A significant challenge in AI adoption is the interpretability of its decisions or predictions. While early AI models were transparent, modern methods like Deep Neural Networks (DNNs) are opaque, making their decisions hard to understand even for their creators.^{3,4} These latest AI algorithms are often seen as complex “black box” models (Castelvecchi, 2016). Although interpretability might not be crucial for certain applications like email categorization or voice assistants, in many cases, humans are hesitant to trust algorithms that lack interpretability and transparency, especially due to incidents of biased outcomes (Zhu et al., 2018). For instance, Google displayed fewer high-paying job ads to women (Datta et al., 2015), Amazon’s same-day delivery skipped black neighborhoods⁵, and digital camera software struggled to recognize non-white faces.^{6,7}

To address AI interpretability, researchers have shifted their focus to eXplainable AI (XAI), a class of methods that aim to produce “glass box” models that are explainable to humans while maintaining a high level of prediction accuracy⁸ (Abdollahi and NaSRaoui, 2016; Csiszár et al., 2020; Doshi-Velez and Kim, 2017; Holzinger et al., 2017b; Lipton, 2017; Murdoch et al., 2019).⁹ XAI enables human users, including nonexperts, to understand and trust AI systems. Its market size was estimated at \$4.4 billion in 2021, projected to reach \$21.0 billion by 2030 with a CAGR

1<https://www.gartner.com/en/information-technology/insights/top-technology-trends>

2https://www.idc.com/getdoc.jsp?containerId=IDC_P33198

3<https://www.newyorker.com/magazine/2017/04/03/ai-versus-md>

4For example, OpenAI’s GPT-3 model has about 175 billion parameters (Brown et al., 2020).

5<https://www.bloomberg.com/graphics/2016-amazon-same-day/>

6<https://www.inverse.com/article/39483-apple-refund-china>

7<https://futurism.com/ai-bias-black-box>

8<https://www.darpa.mil/program/explainable-artificial-intelligence>

9<https://www.darpa.mil/program/explainable-artificial-intelligence>

of 18.4%.¹⁰ While consumer activists advocate for the regulation of AI and the promotion of transparent XAI,¹¹ little is known about its economic implications for firms and consumers (Adadi and Berrada, 2018). Our paper explores the economics of XAI and answers whether preferences for XAI are aligned between customers and firms, and how policymakers should regulate XAI in response.

We model a duopoly market where firms offer a product based on AI algorithms, e.g., auto insurance (where AI determines rates). We let firms set product quality and price while consumers choose to purchase the product from one of the two firms. AI product quality can be related to the accuracy of the AI model or not—such as the coverage of auto insurance. Besides price and quality, the consumers receive AI explanations (XAI). Firms compete to capture market share by choosing their explanation strategy, quality, and price levels. All customers prefer more quality and more explanations, but customers may differ in how they weigh these preferences relative to price. Customers also differ in their preferences for explanation format or method. Thus, firms’ AI explanation strategy includes the explanation level (amount of information) and explanation method (format of information). The policymaker aims to maximize total welfare by designing XAI policies that regulate XAI choices available to firms.

Despite the zero cost of providing XAI, there are equilibria where firms withhold complete explanations. We demonstrate that customers and firms may have misaligned preferences for full explanations. In some environments, firms prefer less than full explanations, while consumers want full explanations. This occurs even when providing XAI is costless, as dominant firms may sacrifice XAI to limit competition and secure profits. To understand this outcome, consider a dominant firm (high quality and high demand) and a smaller firm (low quality and low demand). Intuitively, the dominant firm may benefit from sacrificing its XAI depth (even though it is costless) to ensure that the smaller firm does not compete on quality and price. This helps both firms relieve competition and make positive profits.

This less-than-full XAI outcome occurs even when the underlying data or model (“secret sauce”) is not revealed by XAI. Researchers may mistakenly infer that AI explanations divulge trade secrets, observing a correlation between the depth of the explanation by a dominant firm and the AI quality

¹⁰By Next Move Strategy Consulting, nextmsc.com

¹¹<https://tinyurl.com/theguardian-AI-watchdog>

of a smaller competitor. Instead, such an empirical observation is fully explained by the interplay between endogenous XAI and quality choices in a competitive environment. Thus, correlations between AI explanations and competitor AI accuracy do not necessarily imply the disclosure of trade secrets, challenging the notion that firms are exempt from providing explanations.

In addressing firms’ incentives to conceal AI explanations, the prevailing regulatory approach often mandates full explanations to end users (Pradhan et al., 2022). Our model does replicate this conventional wisdom, but this outcome is not universal. Surprisingly, we find that full explanations may actually disadvantage consumers under some parameter settings. In these instances, consumers may prefer less than full XAI from one or both firms, leading to higher-quality offerings.

Since neither the firms nor consumers may prefer the naïve full explanation regulation, we consider a *strict* XAI policy where the policymaker chooses the depth of XAI ($\xi_{strict} \in [0, 1]$ with 1 being full explanations) and the firms must offer XAI at this depth ($\xi_i = \xi_{strict}$). Although this allows for less than full explanations, firms are still symmetric and therefore are unable to relieve competition. We are motivated to identify XAI policies that allow firms to differentiate on XAI; we contribute by introducing more flexible policy levers. The first is a *self-regulating* XAI policy in which the industry association (policymaker) sets an XAI depth ($\xi_{opt} \in [0, 1]$), and firms can voluntarily choose to adopt the XAI depth or not at all ($\xi_i \in \{0, \xi_{opt}\}$). The second is a *lower bound* XAI policy, where the policymaker sets a lower bound ($\xi_{LB} \in [0, 1]$) and the firms are free to choose XAI above this bound ($\xi_i \in [\xi_{LB}, 1]$).

Finally, we compare across policies. An unregulated market can outperform the strict full XAI policy. Thus, calls for instituting costly strict regulations may be redundant. The flexible XAI policies (self-regulation or lower bound, which may be inherently cheaper since it’s voluntary for firms rather than mandatory) allow firms to differentiate their XAI depth, which may aid social welfare and even consumer surplus. These findings remain robust in various scenarios: (i) differing XAI methods used by firms, (ii) consumer sensitivity or heterogeneity in XAI compared to product quality, and (iii) alternative objectives for policymakers, such as average XAI depth or fairness.

To our knowledge, our study is the first to systematically integrate AI product representation (price and quality), XAI attributes (level and method), and various consumer preferences. Our model is applicable across various AI algorithms and XAI methods, encompassing different regulatory approaches and policy objectives (social welfare, consumer surplus, XAI fairness). Achieving

tractable closed-form solutions for such a comprehensive model is challenging but accomplished by segmenting strategy spaces into analytically solvable regions. Our approach consciously models the interconnected economic forces while abstracting non-economic factors. This uniqueness offers a framework for addressing further research questions in this domain.

Our paper intersects with three emerging research areas. First, it addresses concerns regarding the negative societal implications of unexplained AI (Troncoso et al. 2023; Malik, 2020) and consumers’ perceptions of AI decision changes (Bertini and Koenigsberg, 2021). Second, it explores the tradeoffs between AI accuracy and privacy concerns associated with explainable AI (Adadi and Berrada, 2018), including the potential for adversarial attacks (Goodfellow et al., 2015) and strategic manipulation (Wang et al., 2022). Third, it examines AI algorithms used by competing firms and the pressure they face to adopt accountable AI practices. Despite this, little is understood about the economic impacts of adopting explainable AI for firms. Our paper provides guidance for policymakers and managers in navigating this landscape.

The paper is structured as follows: Section 2 details the theoretical model for firm choices for XAI depth, quality, price, and the resulting profits and consumer utilities. We motivate some of the model elements based on prior literature. Section 3 presents the core equilibrium results for firms’ choices of XAI depth in an unregulated setting. Section 4.1 highlights where consumer preferences for XAI depth are aligned or misaligned with unregulated firm choices. Motivated by these firm and consumer preferences for the depth of XAI, Section 4.2 lays out XAI policies (strict, self-regulated, and lower bound) to maximize consumer surplus. We compare the various policies to show that flexible policies dominate full XAI. Finally, Section 5 shows the robustness of the findings to model extensions (XAI methods, XAI fairness, etc.).

2 The Model

2.1 Background

The rise of XAI research coincides with the evolving regulatory landscape for AI worldwide. The EU’s GDPR, introduced in 2016, grants individuals the “right to an explanation” for algorithmic decisions.¹² This provision has garnered support from organizations like The European Consumer

¹²See Articles 13 through 15: <https://tinyurl.com/EU-regulation-2016-679>

Organization¹³ and companies like Facebook/Meta. Similar mandates for XAI are proposed in the Digital Republic Act (France) and FCRA (US). However, opposition to these proposals exists due to concerns about inherent explainability limits, the trade-off between XAI and AI accuracy, and debates over whether AI decisions should face higher scrutiny compared to human intuition. More detailed arguments for and against XAI are provided in Appendix A1.2.

In addition to the above argument, firms cite other reasons for not adopting XAI, including concerns about the long-term implications of intellectual property theft, the risk of imitation by competitors, susceptibility to adversarial attacks, or the lack of availability of applicable XAI methods for their AI models (Cinà et al., 2022). Although some articles explore the relationship between XAI and adversarial attacks (Galli et al., 2021; Kuppa and Le-Khac, 2020), there is currently no documented evidence that XAI has directly facilitated or enabled such attack vectors. We do not consider external forces against XAI, such as adversarial threats or engineering limitations of XAI tools, which have been discussed previously. This omission highlights that endogenous competitive forces alone can impose constraints on XAI. However, we acknowledge the complexities involved in adopting AI and XAI strategies and account for this friction, by analyzing the firm’s decision-making process when the XAI strategy is considered before the choice of quality and price.

In specifying the economic utility of XAI, we leverage attributes of XAI introduced by Roman et al. (2021) to ground our modeling. The first attribute is the depth (or ‘complexity’) of the explanation (Roman et al., 2021). XAI methods can be tuned to reveal different amounts of information in the explanations e.g., depth of splits in a decision tree. For example, Facebook provides some local explanations for why a certain ad was shown (Figure 1). Facebook’s AI ad targeting engine likely uses dozens of rules, but in this specific example, they explain only three rules that resulted in the customer receiving a particular ad. We model a firm’s continuous choice of XAI depth and not just a binary choice of whether to offer XAI or not. Roman et al., (2021) found that most users prefer more rather than less complexity in explanations. Following this, in our model consumer utility from depth is always increasing.

The second key aspect of an explanation is its method, or format, as proposed by Roman et al. (2021). Explanation methods can vary across multiple dimensions. They may be based on rules (Martens et al., 2008; Martens et al., 2014), feature importance (Ribeiro et al., 2016; Lundberg et

¹³<https://www.wired.com/story/europes-new-privacy-law-will-change-the-web-and-more/>

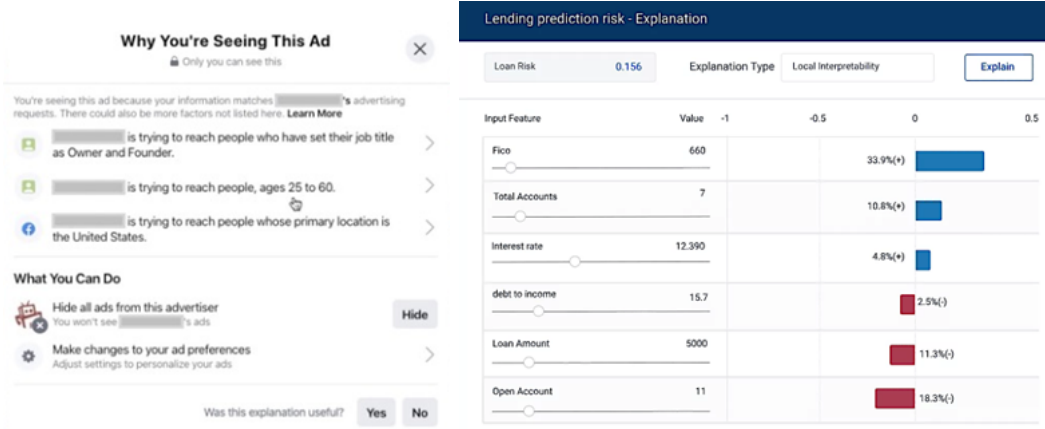


Figure 1: (Left) Facebook provides some explanation about why it shows an ad to you. (Right) When applying for a loan, XAI analyzes each feature and finds its positive/negative effect on the outcome (loan risk). Source: AIMultiple¹⁴

al., 2017), or counterfactuals (Martens et al., 2014). Explanations can take textual or visual forms and can be either global or local. Even within local explanations, various methods such as LIME, LOCO, and SHAP can be employed (Lei et al., 2017; Lundberg and Lee, 2017; Ribeiro et al., 2016). Roman et al. (2021) suggest that consumers may exhibit preferences for one method over another. However, we do not delve into behavioral rationales for such preferences. Instead, our focus is on consumer preference determined by the information set revealed by different methods. Naturally, these information sets may differ only when the methods do not provide full information depth. Although Roman et al. (2021) identify specificity as a third attribute of explanation, we do not incorporate this attribute into our model as it is rooted in behavioral construal level theory (Trope et al., 2010) and is highly context-dependent.

It is important to note that we assume the quality of AI products and the depth of XAI are independent of each other. In certain applications, such as autonomous vehicles and medical image diagnostics, quality may rely on the accuracy of the AI model. Traditionally, there has been an assumption of a trade-off between AI accuracy and interpretability (Adadi and Berrada, 2018), which would challenge the independence of AI product quality and XAI depth. However, recent research suggests otherwise. Rudin (2019) argues that such a trade-off is a "blind belief" and a "myth" lacking empirical support. Rudin demonstrates that structured data and meaningful features often result in negligible accuracy differences between complex classifiers (like deep neural networks, boosted decision trees, random forests) and simpler, more interpretable ones (like logistic

regression, decision lists) after preprocessing. Therefore, our model intentionally excludes any direct trade-off between XAI and quality.

2.2 Model

We consider a duopoly where two firms, indexed $i = 1, 2$, market products with three attributes: XAI depth $\xi_i \in [0, 1]$, quality $q_i \geq 0$, and price $p_i \geq 0$. Product attributes are observable by customers. Without loss of generality, we assume that firm 1 is the high-quality firm and firm 2 is the low-quality one (i.e. $q_1 \geq q_2$). A quadratic cost for quality is assumed βq_i^2 . We model customer preferences using a characteristics-based approach, defined directly over the attribute dimensions of the available products. Customers are heterogeneous in terms of their quality valuation (i.e. willingness to pay). This heterogeneity is captured by a parameter θ , which is uniformly distributed throughout the population and normalized to $[0, 1]$. Customer j 's flow utility from quality of firm (i) is $\theta_j q_i$. Everyone has the same preference for price, so those who purchase from firm i experience the same flow (dis)utility $-p_i$. The third component of firms' products is the explanation that they offer as the reasoning behind their AI's decision. The flow utility of the customer j from the explanations of the firm i is denoted by u_{ij}^e . Below we formulate u_{ij}^e as a function of firm's XAI method e_i , XAI depth ξ_i and customer's preference for explanation x_j . In aggregate, the utility of the customer j from the purchase of the firm i 's product is:

$$u_{ij} = V + \theta_j q_i - p_i + u_{ij}^e(e_i, \xi_i, x_j) \quad (1)$$

where V is the customer's reservation value, i.e., the intrinsic value that he has for the products in this market.¹⁵

Consider a motivating example of the utility of XAI $u_{ij}^e(e_i, \xi_i, x_j)$. In search of auto insurance, Consumer A receives a (AI-based) quote of \$120 along with an explanation - "*your insurance quote differs from the city average of \$100 because your neighborhood has 18% greater incidents of theft*". This consumer would be largely satisfied, since the neighborhood roughly explains the above-average insurance quote of 20%. Consumer B receives the same quote of \$120 along with

¹⁵This makes use of the common assumption in the discrete-choice framework literature that income effects from price changes are negligible (Cunha et al., 2020), that is, income and prices are additive and separable. Therefore, V_j can be omitted from Eq. 2.1 since it does not vary between products.

an explanation - “*your insurance quote differs from the city average because your neighborhood has 4% lower incidents of theft*”. This consumer would be largely unsatisfied with this explanation. If a different factor was explained - “*households with 2+ uninsured individuals increase collision chances by 23%*”, it would have satisfied consumer B (who let’s say had 3 adult family members without insurance).

To ground this motivating example, consider a simple AI model that predicts an insurance rate $\hat{y}_j = \beta_1 z_{1,j} + \beta_2 z_{2,j}$ using a linear model of customer characteristics \mathbf{z}_j . At less than the full depth of the XAI $\xi_i < 1$, only β_1 is revealed in the explanation. Customers who prefer to know β_1 will be satisfied, while others who prefer to know β_2 will be unsatisfied with this explanation. A different XAI method may explain β_2 and therefore be preferred by the second group of customers. At full XAI depth $\xi_i = 1$ both parameters of the model $\{\beta_1, \beta_2\}$ are revealed, therefore all customers are satisfied by either method. ”It is important to note that the method of explanation need not be linked to the actual algorithm that led to the decision. For example, a decision could be reached by a random forest model and explained using a counterfactual” (Roman et al., 2021) or an importance ranking.

Firms can choose different XAI depth ξ_i and method e_i even if they have the same underlying AI model (and therefore the same accuracy or quality). In the model extension section 5, we discuss why XAI methods (e.g. importance ranking vs. counterfactual) may reveal different information sets (β_1 vs. β_2), and why firms may do this to horizontally differentiate, i.e., $e_1 = 0$, $e_2 = 1$. For our baseline model, we will limit the XAI method to be identical between firms $e_1 = e_2 = 0$. We assume that customers are distributed over the characteristic space so that their distances from the XAI method are uniformly distributed $x_j \sim U[0, 1]$. Our baseline model of the utility of XAI is thus given by:

$$u_{ij}^e(e_i, \xi_i, x_j) = -t(1 - \xi_i) |x_j - e_i| \xrightarrow{e_1=e_2=0} -t(1 - \xi_i) x_j$$

In Appendix A.1.4 we provide a more elaborate decision tree-based AI model and how XAI depth and method impact customer utility from explanations. Here, consumers have a fixed location in a characteristic space. The depth of XAI ξ_i determines “how many” points in characteristic space are explained, the XAI method e_i determines “which” set of points in characteristic space are

explained, and the customer heterogeneity x_j captures the distance between the customer's fixed location and the closest explained point.

Firms' Game: Firm i 's profit is $\pi_i = p_i d_i - \beta q_i^2$ where d_i denotes its market share. We consider zero technical cost of implementing XAI due to the availability of cutting-edge, model-agnostic XAI algorithms (e.g., SHAP, LIME) that can be downloaded as free, open-source packages. Firms aim to maximize their profits in a three-stage game. In stage one, they choose personalization levels ξ_1, ξ_2 . In stage two, they select quality levels q_1, q_2 . Finally, in stage three, they set prices p_1, p_2 . We will enforce non-negative payoffs for the firms. Consequently, the equilibria identified for this three-stage game are also viable in an extended game where firms decide on market entry in stage zero preceding XAI, quality, and price choices. The intuition behind the three-stage game structure is the fact that prices are more flexible than quality in the short term (Cunha et al., 2020). Likewise, product quality (e.g., through the AI model's accuracy) is more flexible than that of the XAI strategy. Thus, decisions in the early stages can be viewed as the firm's long-term strategy, while subsequent stages involve progressively shorter-term decisions. At each stage, firms simultaneously make their own decisions. This three-stage game is of perfect information because both firms have access to not only complete information (rules of the game, available strategies, and the payoffs associated with each outcome) but also all past moves made by both firms. This means that at any point in the game, each firm knows the entire history of the game up to that point.

2.3 Market Structures

This problem is solved using backward induction starting with prices (p_1, p_2) in Stage 3, followed by qualities (q_1, q_2) in Stage 2, and XAI depth (ξ_1, ξ_2) in stage 1. The analysis becomes intricate when we consider that the mathematical expressions for demand functions (d_1, d_2) (and thus for-profit functions π_1, π_2) vary depending on the values of $\xi_i, q_i, p_i, i = 1, 2$. In different ranges of these variables, the indifference lines could intersect θ -axis (Figure 2 left and middle) or x -axis (Figure 2 right) to split the demand between firms.

Consider that the firms are differentiated both in quality and in XAI depth. Intuitively, when the cost of explanation misfit t is small, consumers are more heterogeneous in preference for quality. Consumer demand is likely to be split based on quality preferences, that is, when the indifference

lines intersect the θ -axis. We call this a *quality-dominated* market. When the cost of explanation misfit t is large, consumers are very heterogeneous in preference for XAI. Consequently, consumer demand is likely to split based on explanations, that is, indifference lines intersecting x -axis. We call this an *explanation-dominated* market. Note that t is exogenous, while the market structure *explanation-dominated* or *quality-dominated* is endogenous. Notwithstanding our example above, it is possible that even under small t the firms make endogenous choices about quality and XAI depth (for example equal qualities) that result in an *explanation-dominated* market. In practice, we believe that consumers care first and foremost about the quality of the AI product before its explanation. Therefore, we will focus on the former small t regions in Section 3 and discuss the latter large t regions as an extension in Section 5.

To identify pure strategy equilibria,¹⁶ we assume that both market structures exist and solve for the equilibrium values of ξ_i, q_i, p_i in each market using the appropriate mathematical expressions for demand. Then we examine whether any firm i would be better off if they switched from their optimal values in one market to the other. If no firm unilaterally deviates, then we have found a Nash equilibrium. Note that the special treatment of these two market structures does not alter the game; this only describes the procedure we use to identify pure strategy equilibria.

3 Results

In this section, we discuss equilibrium XAI choices for unregulated firms. Tirole (1988) provided some guidance for the equilibrium results when firms choose quality (q_1, q_2) and prices (p_1, p_2) . Firms can maintain a differentiated quality equilibrium ($q_1 - q_2 > 0$) where the revenue for both firms is increasing in quality differentiation. The higher quality firm prefers to increase quality, and the lower quality firm prefers to decrease quality. Revenue for both firms (even the firm with low quality) is increasing in *maximal differentiation* on quality because it weakens price competition in the last stage. This principle of maximal differentiation is not unbounded. Practical considerations such as available quality levels, cost of quality, and loss of market coverage may limit (or even prohibit) quality differentiation. We will return to these boundaries, but first, we analyze how the addition of the XAI depth choice (ξ_1, ξ_2) in stage 1 changes this equilibrium result.

¹⁶We will limit our examination to pure strategy, therefore any findings about firm or policymaker choices do not consider mixed strategy equilibria.

3.1 Unregulated XAI

To study the unregulated case and for our results to be analytically tractable, we limit the choice of quality to $q_i \in \{q_l, q_h\}$ (with the notation $\Delta q = q_h - q_l$). Without loss of generality, firm 1 is fixed as a high-quality firm. Instead of exhaustively enumerating equilibria under all exogenous parameter ranges, we focus on two-parameter ranges of interest. First, a range where both firms choose the full depth of the XAI ($\xi_1^* = 1, \xi_2^* = 1$) in equilibrium and thus are undifferentiated in XAI, i.e., $\Delta \xi = \xi_2^* - \xi_1^* = 0$. Second, is a range where one firm (high-quality firm) chooses less than the full XAI depth ($\xi_1^* < 1, \xi_2^* = 1$) in equilibrium; thus, the firms are differentiated on XAI i.e., $\Delta \xi = \xi_2^* - \xi_1^* \neq 0$. These two equilibria hold for the mutually exclusive exogenous parameter β, q_l, q_h ranges expressed in Proposition 3.1. Figure 2 (left and middle) illustrates the indifference lines ($u_{1j} = u_{2j}$) under the two equilibria. As the exogenous cost of quality β increases, firms switch from the former to the latter equilibrium. We are interested in unraveling the intuition for when and why unregulated firms switch to less than full XAI even when XAI is costless to offer.

Proposition 3.1. *Unregulated firms with XAI choice $\xi_i \in [0, 1]$, quality choice $q_i \in \{q_l, q_h\}$, and price choice $p_i \geq 0$ set in equilibrium (proof in Appendix A2.1)¹⁷:*

a. **Undifferentiated full XAI** ($\xi_1^* = 1, \xi_2^* = 1$) if $\beta \leq \frac{\Delta q}{9q_l^2}$:

$$(q_h, q_l) = \left(\frac{2\Delta q}{3}, \frac{\Delta q}{3} \right)$$

$$\pi_1^* = \left(\frac{2}{3} \right)^2 \Delta q - \beta q_h^2 \quad \pi_2^* = \left(\frac{1}{3} \right)^2 \Delta q - \beta q_l^2$$

b. **Differentiated partial XAI** ($\xi_1^* = 1 - \Delta \xi_{ug}^*, \xi_2^* = 1$) if $\frac{\Delta q}{9q_l^2} < \beta < \frac{1}{9q_l}$:

$$(q_h, q_l) = \left(\frac{2\Delta q}{3} - \frac{t\Delta \xi_{ug}^*}{6}, \frac{\Delta q}{3} + \frac{t\Delta \xi_{ug}^*}{6} \right)$$

¹⁷For these analytical result, we had to limit firms to choosing between two quality levels $\{q_l, q_h\}$. We employ a numerical simulation to relax this simplification. Numerical simulations utilize Pytorch tensors (widely used for training deep learning models on GPU) for speed improvement. Consistent with Proposition 3.1, there are mutually exclusive exogenous parameter (β, t) ranges where firms set full XAI and less than full XAI. The numerical results are consistent with an upcoming discussion using the simple analytical model on how firms choose XAI and quality levels to weaken competition and maximize profits.

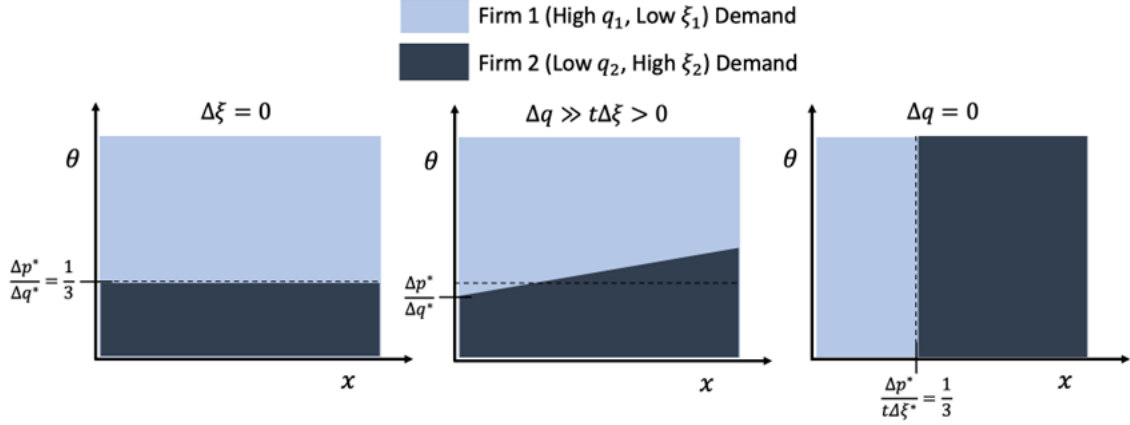


Figure 2: Demand for Firm 1 and Firm 2 from consumers uniformly distributed over explanation preference $x_j \in [0, 1]$ and quality preference $\theta_j \in [0, 1]$. At equilibrium (a) with $\Delta q > 0, \Delta \xi = 0$ the indifference line is horizontal. At equilibrium (b) with $\Delta q > 0, \Delta \xi > 0$ the low-quality firm gains additional demand because of XAI advantage $\xi_2 > \xi_1$. For large values of t (not captured in Proposition 3.1) the indifference line will start to intersect x -axis, eventually becoming horizontal for explanation-dominated equilibrium ($\Delta \xi > 0, \Delta q = 0$)

$$\pi_1^* = \left(\frac{2}{3} - \frac{t \Delta \xi_{ug}^*}{6 \Delta q} \right)^2 \Delta q - \beta q_h^2 \quad \pi_2^* = \left(\frac{1}{3} + \frac{t \Delta \xi_{ug}^*}{6 \Delta q} \right)^2 \Delta q - \beta q_l^2$$

where Δq and $\Delta \xi_{ug}^*$ is the notation of $\Delta q = q_h - q_l$ and $\Delta \xi_{ug}^* = \min \left(1, \frac{6}{t} \left(q_l \sqrt{\frac{\beta}{\Delta q}} - \frac{1}{2} \right) \right)$, respectively, in a market where $2q_l > q_h > q_l$ and $t < (\Delta q/4) * (q_h/q_l)^2$.

Note that firms choose differentiated quality in stage 2 ($\Delta q > 0, q_1^* = q_h, q_2^* = q_l$) under both equilibria. Firms want to differentiate on quality to weaken competition and maximize profits. We see that under both equilibrium outcomes, firm revenues are increasing in the quality differential Δq . Thus, the addition of Stage 1 XAI depth choices does not appear to change each firm's incentive for maximal differentiation in quality. If more than two quality levels were available, firms would prefer to differentiate further in quality (in Stage 2) under Stage 1 XAI choices for either equilibrium.

Next, we compare firm profits π_i^* under both equilibria. Firm 1 has a strictly lower profit in equilibrium (b). This is a direct result of Firm 1's lower XAI depth in equilibrium (b). Therefore, it appears that Firm 1 would prefer not to deviate from full XAI in equilibrium (a) to partial XAI in equilibrium (b). This is true for a small cost of quality β . More generally, when firms can differentiate costlessly in two dimensions, they prefer to only differentiate on the “dominant”

dimension. Among the two dimensions here (quality and XAI), quality is the “dominant” dimension at low t (see the condition on t in Proposition 3.1) because small transportation cost means that consumer heterogeneity in XAI does not substantially influence their utility. Consequently, firms can more effectively weaken competition by differentiating on quality. On the secondary XAI dimension, neither firm has an incentive to differentiate, so both raise their XAI depth (costlessly) to 1.

Given the above intuition, the existence of equilibrium (b) appears surprising. The incentives for Firm 1 (the high-quality firm) to sacrifice its XAI in Stage 1 will become apparent when we provide a close examination of the Stage 2 equilibrium. For a large cost of quality β , the Stage 2 differentiated quality equilibrium ($q_1 > q_2$) cannot be sustained under full XAI ($\xi_1 = 1, \xi_2 = 1$). Instead, the two firms end up in a competitive equilibrium ($q_1 = q_2$) and receive zero profits. Consider, for example, the profit for Firm 2 (lower quality firm) in equilibrium (a). As the cost of quality β increases up to $\beta = \Delta q/9q_l^2$ Firm 2’s profit goes to zero ($\pi_2^* = 0$). If β increases beyond this level, Firm 2 prefers to deviate from differentiated quality ($q_2 = q_l < q_h = q_1, \pi_2 < 0$) to the competitive quality level ($q_2 = q_h = q_1, \pi_2 = 0$). At the boundary $\beta = \Delta q/9q_l^2$, the profit of Firm 1 remains positive, i.e., $\pi_1 > 0 = \pi_2$. To avoid a fully competitive equilibrium, Firm 1 has an incentive to sacrifice its XAI depth in Stage 1 and transfer some of its profit to Firm 2. To see this visually, notice that in Figure 2 (left) the low-quality Firm 2 has a lower demand and price, thus a lower revenue. In Figure 2 (middle) the low-quality Firm 2 gets a slightly higher demand because Firm 1 lowers its XAI depth. The XAI advantage ($\xi_2 > \xi_1$) gives the low-quality firm additional demand and revenue. Thus, discouraging Firm 2 from competing on quality. Therefore, we find that *reverse* XAI differentiation in Stage 1 ($\xi_2 > \xi_1$) helps sustain a differentiated quality equilibrium in Stage 2 ($q_1 > q_2$).

Generalization: Proposition 3.1 only captures equilibria when available quality levels are discrete (q_l, q_h) and explanation misfit cost t are low, we discuss briefly below that similar equilibria with differentiated XAI exist more generally outside of these exogenous parameter ranges.

First, consider the discrete quality levels (q_l, q_h). When $2q_l < q_h$, low-quality Firm 2 (instead of the high-quality Firm 1) has an incentive to sacrifice its XAI depth in stage 1. This happens because under full XAI ($\xi_1 = 1, \xi_2 = 1$), it is the higher quality firm 1 whose profits become negative first as β increases. Here, Firm 1 deviates away from differentiated quality to the competitive quality

choice. The low-quality Firm 2 still has positive profits at this boundary β (across which Firm 1 profits switch from positive to negative) and therefore has an incentive to sacrifice its XAI depth in Stage 1. This interestingly results in an equilibrium where Firm 1 has a higher XAI and higher quality.

To better understand when incentives for the differentiation of XAI kick in, remember that the maximal quality differentiation in Stage 2 is bounded. First, it is bounded by *cost limits*. For example, as the high-quality firm continues to increase its quality q_1 , its revenue increases, but its costs βq_1^2 can start to dominate. Second, maximal differentiation can be bounded by *demand limits*. For example, as the low-quality firm reduces its quality q_2 , it expects not to lose demand to the high-quality firm (weaker competition). But it can lose demand if some consumers decide not to purchase at all ($u_{ij} < 0 \forall i$). When *cost limits* and *demand limits* are increasingly tight at available quality levels, a differentiated quality equilibrium may not be viable for any. In such cases, firms will have incentives for XAI differentiation in Stage 1 to maintain differentiation quality equilibrium in Stage 2.

Second, consider relaxing the upper bound on t in Proposition 3.1. When t is large, firms can more effectively weaken competition by differentiating on XAI (then quality). We had concluded earlier that when firms can differentiate costlessly on two dimensions they prefer to only differentiate on the “dominant” dimension. Intuitively, firms would prefer to differentiate maximally on XAI instead of quality when t is large. We will generalize this more formally, large t regions, when we discuss extensions in Section 5.

3.2 AI Explanation and Accuracy Correlation

As a by-product of the equilibria (a) and (b) captured in Proposition 3.1, we can examine important questions about the relationship between AI explanations and AI quality. Note that an AI product’s accuracy can be a key determinant of its quality. Industry practitioners often raise the concern that AI explanations reveal the “secret sauce” of the underlying AI model. A competitor can therefore use AI explanations to infer good AI model design and thereby improve its own AI accuracy. Empiricists may look for instances where one firm increases its explanation levels and check to see if competing firms improve their AI accuracy simultaneously. This claim can be used to argue against XAI regulations. However, in this section, we will discuss why this conclusion may be

erroneous.

Under equilibria (a) and (b) in Proposition 3.1, revenue for both firms increases with Δq . However, in the differentiated equilibrium of XAI (b), the revenue of both firms is increasing slower with (Δq) . Consequently, if more than two quality levels were available, we would expect a smaller quality differentiation $(q_1^* - q_2^*)$. More generally, when firms differentiate on a secondary dimension (XAI here), they endogenously differentiate less on the “dominant” dimension (quality here). Furthermore, under the differentiated XAI equilibrium (b), (relative to (a)), Firm 1 can increase q_1 (to increase Δq) but faces a tighter *cost limit* $(q_1^*(\Delta\xi > 0) < q_1^*(\Delta\xi = 0))$ on increasing q_1 resulting in a smaller q_1^* . Firm 2 can decrease q_2 (to increase Δq) and face a weaker *demand limit* $(q_2^*(\Delta\xi > 0) < q_2^*(\Delta\xi = 0))$, resulting in a smaller q_2^* . Overall, we would expect lower q_1^*, q_2^* and average quality $(q_{avg} = (q_1^* d_1^* + q_2^* d_2^*) / (d_1^* + d_2^*))$ at equilibrium (q_1^*, q_2^*) .

Now consider a market where the cost of quality β increases such that the equilibrium shifts from full XAI (a) to differentiated partial XAI (b). Our model predicts that the larger firm (high quality, high demand, high revenue) will reduce its XAI depth, and the smaller firm will reduce its quality levels. Empirically, there appears to be a **positive correlation between AI explanations by the large firm and AI quality of the smaller competitor**. One may jump to the causal conclusion that AI explanations reveal the “secret sauce” that smaller competitors use to improve their AI models. However, this conclusion can be erroneous. The correlation is fully explained by the impact of differentiated XAI on incentives to maximally differentiate on quality and is not driven by any learning about the underlying AI model from AI explanations.

Looking further into the quality and XAI tension, we discuss the impact on average (market) AI explanation and quality. Consider both firms offering less than the full XAI, i.e. $\xi_1 = \xi_2 < 1$ (still $\Delta\xi = |\xi_1 - \xi_2| = 0$), the equilibrium expression for prices remains the same as above. Firm 2 is more constrained in decreasing q_2 (to increase Δq) from a *demand limit*. We would expect lower differentiation and higher average quality (q_{avg}^*) at equilibrium (q_1^*, q_2^*) . Empirically there appears to be a **negative correlation between average AI explanations and average investment in AI quality**. Again, one may jump to the causal conclusion that AI explanations reveal the “secret sauce” that inhibits investment in AI models. However, this conclusion is again potentially erroneous.

4 XAI Regulations

In Section 3, we presented the results of our analysis for the unregulated case. We observed equilibria in which firms provide full or partial explanations in different parameter spaces. Section 4.1 examines consumer surplus to answer whether consumers always prefer full XAI. We also identify when firms and consumers have conflicting preferences for full vs. partial XAI depths to guide when and how policymakers should regulate firm XAI depths. Section 4.2 proposes various regulatory levers that the policymaker can deploy.

4.1 Consumer Surplus and Firm Profits

Under Tirole (1988), when firms choose quality (q_1, q_2) and prices (p_1, p_2) , consumers dislike quality differentiation, since it weakens firm competition and increases prices. Our model adds an additional dimension (XAI) for firms to differentiate on, over the work of Tirole 1988. However, it is important to address whether Tirole (1988) holds with this additional dimension. Proposition 4.1 provides consumer surplus U under equilibria (a) and (b) discussed in the last section. Under the undifferentiated full XAI equilibrium (a), consumer surplus U is trivially decreasing in quality differentiation (increasing Δq and decreasing q_2). For the differentiated partial XAI equilibrium (b), consumer surplus U decreases in quality differentiation (increasing Δq and decreasing q_2) and in XAI differentiation $\Delta \xi^*$.

Proposition 4.1. *The consumer surplus U under each equilibrium is (proof in Appendix A2.2):*

$$a. \text{ Undifferentiated full XAI equilibrium } (\xi_1^* = \xi_2^* = 1); (q_h, q_l) = \left(\frac{2\Delta q}{3}, \frac{\Delta q}{3} \right)$$

$$U = \frac{q_l}{2} - \frac{\Delta q}{9}$$

$$b. \text{ Differentiated XAI equilibrium } (1 - \Delta \xi_{ug}^*, 1); (q_h, q_l) = \left(\frac{2\Delta q}{3} - \frac{t\Delta \xi^*}{6}, \frac{\Delta q}{3} + \frac{t\Delta \xi^*}{6} \right)$$

$$U = \frac{q_l}{2} - \frac{\Delta q}{9} - \frac{t\Delta \xi}{36}\alpha; \alpha > 0$$

Next, we compare the surplus of consumers across the equilibria of (a) and (b). Under equilibrium (b), relative to equilibrium (a), the consumer (surplus U) is hurt by XAI differentiation. If

consumers, through a policymaker, could influence the depth of XAI, consumers would prefer the full undifferentiated XAI.

Although consumer preference for full XAI is consistent with conventional wisdom, there are caveats to this conclusion. Consider enforcing full XAI when the cost of quality β is high ($\beta > \Delta q/9q_l^2$ in Proposition 3.1). Here, the profits of Firm 2 (lower quality) would be negative under differentiated quality ($q_1 = q_h, q_2 = q_l$). Thus, firms establish fully competitive undifferentiated qualities, resulting in competitive equilibrium profits ($\pi_1^* = \pi_2^* = 0$) that cause firms to be indifferent between market entry and not entering. Consequently, a monopoly is also an equilibrium and the policymaker may not regulate to a full explanation. Rather, a differentiated partial XAI should be examined when the cost of quality β is high. In doing so, an equilibrium is sustained with positive profits for both firms.

Consider another scenario that allows for differentiated XAI. In Proposition 3.1 equilibrium (a), firms (undifferentiated on XAI) can maximally differentiate on quality. Unregulated firms have no incentive to differentiate on XAI when they can differentiate successfully on the “dominant” quality dimension. Now consider when firms are forced to exogenously differentiate on XAI rather than in the unregulated case. The equilibrium (b.) of Proposition 3.1 states that Firm 2 (the firm with the XAI advantage) does not deviate from low quality in stage 2. However, when a regulator forces no XAI differentiation or a very large amount of XAI differentiation and Firm 1 chooses to sacrifice XAI, Firm 2, the firm now with the XAI advantage, will be better off to deviate from low quality to high quality in stage 2. Firm 2 is also able to capture 100% of the demand by undercutting Firm 1’s price in stage 3. Given this, Firm 1 in stage 1 will never elect to sacrifice XAI by a large degree. Thus, a sufficiently large exogenously forced XAI differentiation can result in undifferentiated quality. Now compare these two settings from a consumer’s point of view – undifferentiated XAI allowing maximal differentiation on quality (Proposition 3.1(a) and Proposition 4.1(a)) vs. forced differentiated XAI resulting in undifferentiated quality. In the former, consumers face differentiated quality and in the latter, they face differentiated XAI. Consumers dislike both quality and XAI differentiation. But when they dislike quality differentiation more, consumers may prefer differentiated partial XAI ($\xi_2 - \xi_1 > 0$). In essence, consumers prefer the firm to sacrifice XAI for higher quality.¹⁸

¹⁸Consumers have similar preferences for less than full XAI when there is a single monopoly firm.

Proposition 4.2. *There exist exogenous parameter ranges where (proof in Appendix A2.3):*

- a. Both firms and consumers prefer full XAI ($\xi_1 = \xi_2 = 1$)*
- b. Firms prefer differentiated XAI ($\xi_1 < 1, \xi_2 = 1$) while consumers prefer full XAI ($\xi_1 = \xi_2 = 1$)*
- c. Both firms and consumers prefer differentiated XAI ($\xi_1 < 1, \xi_2 = 1$)*

In setting (a), no regulation is required given that XAI preferences for both consumers and firms are aligned for a full explanation which occurs for $\beta \leq \Delta q/9q_i^2$. But Proposition 4.2 also identifies scenarios where regulation of XAI may be required given firm profits and consumer surplus. In setting (c), while firms and consumers prefer less than full XAI, they may prefer different levels. However, regulation should not mandate full explanations. In setting (b) firms and consumers are clearly misaligned, leading a policymaker to consider regulating firms at a full XAI depth.

4.2 Regulatory Levers—The Policymaker’s Problem(s)

The policymaker’s goal is to maximize either total consumer surplus U which is calculated by integrating Eq. 1 over demand for firm 1 (d_1) and for firm 2 (d_2) or Total Welfare (W), which includes consumer surplus and firm profits. Below we discuss results related to consumer surplus and relegate our discussion of total welfare to the Appendix. We are motivated to look for policies that allow some room for firms to set XAI less than full (to cater to setting (b) and (c) in Proposition 4.2 for example).

We consider three policy setups: (1) **Strict** XAI, (2) **Self-Regulating** XAI and (3) **Lower Bounded** XAI.¹⁹ In strict XAI, the policymaker sets the XAI depth ξ_{strict} and enforces this policy ($\xi_1 = \xi_2 = \xi_{strict}$). In self-regulating XAI, industry firms are motivated by the threat of possible government regulation that they take it upon themselves to impose self-regulatory actions (DeMarzo et al., 2005). Here, the policymaker is replaced by an industry association which sets the depth of XAI ξ_{SR} as a guideline, but firms are free to choose whether they will offer XAI at this depth or not at all ($\xi_i \in \{0, \xi_{SR}\}$).²⁰ For self-regulation policy, the industry association (policymaker) must

¹⁹In Appendix A3 we provide industry examples (outside of XAI context) where such regulatory levers have been used.

²⁰One such example of self-regulation is in the food industry where the industry has provided guidance on lowering children’s food marketing to its firms

predict the equilibrium strategies of the firms and set the self-regulating ξ accordingly. Similar to the self-regulating policy, in a lower bound policy, the policy maker sets a minimum level of XAI i.e., $\xi_i \in [\xi_{LB}, 1]$ and firms are free to choose their XAI depth within these limits. Here, the government regulator, through the choice of ξ_{LB} , can shape the market structure and hence the equilibrium strategies of the firms.

In the rest of this section, we will discuss the optimal choice for the policymaker in each policy environment. Then we will compare the consumer surplus across the policies.

Strict Policy ($\xi_i = \xi_{strict}$): The first policy we discuss is a strict policy that enforces a uniform level of XAI ξ_{strict} across all firms such that $\xi_i = \xi_{strict}$. Current conventional wisdom may lead policymakers to set a strict full explanation standard, that is, $\xi_{strict} = 1$. However, we find that $\xi_{strict} = 1$ is not guaranteed to maximize consumer surplus (or social welfare). The loss of firm profits at $\xi_{strict} = 1$ is based on Proposition 4.2 (b), where only consumers, but not firms, prefer full XAI. But why would consumers not prefer a regulatory policy of ($\xi_{strict} = 1$)? The reason is counterintuitive, since consumers lose explanations, while firms remain undifferentiated in both cases. The rationale for this becomes clearer when we examine the tension between average XAI depth and quality.

Note that when the XAI depths of the firm are exogenously set to undifferentiated levels ($\xi_1 = \xi_2 = \xi_{strict}$) then firms want to differentiate maximally on quality. However, as discussed above, quality differentiation is bound by *cost limits* and *demand limits*. As ξ_{strict} is reduced from 1, *demand limits* becomes tighter, i.e. the low-quality firm starts losing market coverage and, therefore, its profit. High-quality Firm 1 can transfer some of its demand (by lowering quality) to low-quality Firm 2. Eventually, both firms reach equal competitive quality and zero profits ($\pi_1 = \pi_2 = 0$). While the consumers lose surplus from lack of explanations, they gain surplus due to high competitive qualities and low prices. In general, consumers can be better off at $\xi_{strict} < 1$.

Self-Regulating Policy ($\xi_i \in \{0, \xi_{SR}\}$): The second policy that we study is a self-regulating policy where firms can either choose to offer XAI at ξ_{SR} depth or not offer XAI at all, that is, $\xi_i \in \{0, \xi_{SR}\}$. This allows the firms to differentiate on the depth of the XAI if only one firm opts to offer XAI at ξ_{SR} , which is set by the industry association (policymaker). The findings from the application of such a policy also extend²¹ to a similar policy where firms choose between

²¹The consumer surplus is scaled up under $\xi_i \in \{1 - \xi_{SR}, 1\}$ compared to $\xi_i \in \{0, \xi_{SR}\}$, without changing any key

$\xi_i \in \{1 - \xi_{SR}, 1\}$ (similar to the unregulated equilibrium in Proposition 3.1 (b)).

Proposition 4.2 (a) provides one trivial scenario where a cheaper self-regulating policy works as well as the costly strict policy since both firms and consumers prefer full XAI—but the need to regulate here is limited. In Proposition 4.2 (b), only consumers prefer full XAI. Proposition 3.1(b) provides one such equilibrium in which unregulated firms would choose differentiated XAI ($\xi_2^* - \xi_1^* = \Delta\xi_{ug}^* > 0$). The rationale for $\Delta\xi_{ug}^* > 0$ was that Firm 1 (high-quality) reduces its XAI depth by $\Delta\xi_{ug}^*$ to give Firm 2 (low quality) additional demand to discourage it from deviating to a competitive quality level. The ideal choice for Firm 1 would be to provide Firm 2 just enough XAI advantage to make non-zero profits at low quality. A self-regulating standard of $\xi_{SR} < \Delta\xi_{ug}^*$ does not allow Firm 1 to provide sufficient XAI advantage to Firm 2. Similarly, a self-regulating standard of $\xi_{SR} = 1$ gives Firm 1 two options – not sacrifice XAI at all or sacrifice XAI to 0. Given these options, Firm 1 may choose to opt in. This is because Firm 1, which prefers to reduce its XAI depth by a moderate $\Delta\xi_{ug}^*$, does not want to reduce its XAI to 0 by opting out. Thus, the cheaper self-regulating policy again works as well as the costly strict policy. Finally, in Proposition 4.2 (c) scenario, both firms and consumers prefer partially differentiated XAI (but not necessarily at the same level). Here, an $\xi_{SR} < 1$ improves both consumer surplus and firm profits (therefore also social welfare) relative to $\xi_{SR} = 1$. Similar to strict policymaking, $\xi_{SR} = 1$ is not guaranteed to maximize consumer surplus (or social welfare) under a self-regulating policy.

Lower Bounded Policy ($\xi_i \in [\xi_{LB}, 1]$): Finally, we consider a more flexible lower bound regulation ξ_{LB} . In this policy, the policymaker sets a minimum level of XAI i.e., $\xi_i \in [\xi_{LB}, 1]$, which promotes competition while maintaining a minimum level of XAI for all consumers. The lower bounded policy may be less realistic than a strict or self-regulating policy. It is unclear if there is a mechanism for firms to communicate arbitrary XAI depths outside of setting regulatory standards. Under a lower bound policy $\xi_i \in [0.5, 1]$, it is unclear how firms communicate different XAI depth of say 0.75 or 0.9. Yet again, $\xi_{LB} = 1$ is not guaranteed to maximize consumer surplus (or social welfare) under the lower bound policy. The rationale comes from Proposition 4.2 (c). When both firms and consumers prefer partially differentiated XAI a lower bound standard of ξ_{LB} allows Firm 1 to sacrifice its XAI depth up to a maximum of $(1 - \xi_{LB})$. Further in Proposition 4.2 (b) where consumer prefers full XAI but firms do not, if $(1 - \xi_{LB}) < \Delta\xi_{ug}^*$ the XAI advantage available to

intuition.

Firm 2 may not be sufficient to discourage it from deviating to competitive quality. Consequently, even a lower bound choice $\xi_1 \in [\xi_{LB}, 1]$ results in $(\xi_1^* = 1, \xi_2^* = 1)$. Thus, a cheaper lower bound policy works as well as a costly strict policy.

Corollary 4.3. *Under a variety of XAI policy levers (strict, self-regulating, and lower bound) the optimal choice of regulated level (ξ_{strict}^* , ξ_{SR}^* and ξ_{LB}^* respectively) to maximize consumer surplus (and social welfare) is not always 1. Comparing across policy levers, the self-regulating XAI (and similarly flexible Lower Bounded XAI) policy can result in greater consumer surplus (and social welfare) than the more extreme policies of - Strict XAI and Unregulated XAI (Proof in Appendix A2.4).*

5 Extension

In this section, we test the robustness of our findings while extending the model to an environment where firms use **different XAI methods**. In the model presented in Section 2 firms use the same XAI method, this corresponds to the firms located at the center of the Hotelling line (i.e., $e_1 = e_2 = 0.5$). We analytically model firms' deployment of different XAI methods by setting $e_1 = 0, e_2 = 1$. Note that the choice of the XAI method (e_i) is a long-term strategy that often requires multiple levels of approval within the organization and with the regulators. The way XAI is presented also shapes how customers interact with the firm's product. Therefore, changing e_i requires a significant investment in product repositioning. Consequently, we treat firms' locations on the Hotelling line as exogenous decisions made before the start of the game. Intuitively, an environment with different XAI methods allows a new dimension on which firms have a fixed difference and, therefore, changes their incentives to differentiate on XAI depth and quality.

In section 2.2 we presented two XAI methods as revealing different explanations for the same underlying AI model $\hat{y}_j = \beta_1 z_{1,j} + \beta_2 z_{2,j}$. For concreteness, consider two XAI methods. The first method iterates over the features and ranks them based on their explanation or predictive power in *isolation*. For example, the model $\beta_1 z_{1,j}$ may have an R square higher than $\beta_2 z_{2,j}$. At less than full explanation, this XAI method would reveal only β_1 in its explanation. The second method provides counterfactuals intended to convey the features that may make it easier for consumers to improve and change their prediction \hat{y}_j . At a less than full explanation, this XAI method would

only reveal β_2 in its counterfactuals. In Appendix A.1.5, we provide a practical example of feature importance and counterfactual methods applied to the same AI model. Fundamentally, the set of features explained by XAI methods can differ because the XAI methods have different objectives.

An environment with different XAI methods also requires an examination of **XAI Fairness**. Absolute XAI fairness refers to the situation in which all customers on the Hotelling line receive the same XAI utility. This notion of XAI fairness is similar to the *equal outcome* definition of AI fairness (Hardt et al., 2016) and can only happen when firms offer full explanations ($\xi_1 = \xi_2 = 1$).²² If XAI in equilibrium turns out to be less than full XAI i.e., $\xi < 1$, then different customer predictions on the Hotelling line x_j receive unequal explanations. Consider two customer groups – protected (historically marginalized) and unprotected. The predictions for the two groups may be best explained by different features, e.g., employment or education for one group, but credit history and purpose of loan for another group. On the Hotelling line, members of the protected group will be closely clustered together (say on the right extreme) but distant from the unprotected group (say in the middle). If the two firms use the same XAI method (situated in the middle) and offer less than full explanations, the protected group systematically receives less explanation. When firms use the same XAI method and therefore coincide on the Hotelling line, the goal of XAI fairness and average XAI depth become equivalent. When firms use different XAI methods, the objective of XAI fairness is not aligned with the average depth of XAI.

Besides extending to different XAI methods, we allow for a continuous choice of quality instead of the two discrete levels discussed in the above model. We further discuss results (i) for regions with a large exogenous parameter t , and (ii) other policymaker objectives besides consumer surplus. In Appendix A4, we provide formal equilibrium solutions. These model extensions will create too many permutations if every choice of regulatory policy (strict or self-regulating), XAI method (same or different), market structure (E or Q), and policymaker objective (welfare or fairness) are discussed. Therefore, we only discuss here scenarios that reveal new insights for policymaking or managerial implications.

²²We must have $u_{1j}^e = u_{2j}^e \ \forall x_j \in [0, 1]$. Thus, $\xi_1 = \xi_2 = 1$ and $u_{1j}^e = u_{2j}^e = 0$.

5.1 Strict, Lower-Bounded and Unregulated XAI Policy Policies

Strict Policy: Recall that under the setting of identical XAI methods, strict policies eliminate differentiated XAI depth ($\xi_1 = \xi_2 = \xi_{\text{strict}}$), the market cannot be divided based on XAI differentiation $\Delta\xi$. As ξ_{strict} decreases, firm profits strictly decrease, eventually going to $\pi_1 = \pi_2 = 0$, as firms set competitive qualities and prices. Under different XAI methods, the strict XAI policy level (ξ_{strict}) has an additional force. As ξ_{strict} decreases, the horizontal differentiation between the firms on the Hotelling line increases because the horizontal differentiation of each firm to the other is $t(1 - \xi_{\text{strict}})|e_1 - e_2|$. This in turn relieves price competition and improves profits. Consequently, **firm profits are not necessarily maximized at $\xi_{\text{strict}} = 1$ under different XAI methods.**

More generally, we can confirm Proposition 4.3 (i.e., $\xi_{\text{strict}}^* = 1$ may not maximize welfare W) even under a different XAI method, continuous quality choices, and a wider range of t . Furthermore, we can extend this conclusion beyond a total welfare W (or consumer surplus U) objective. XAI Fairness objective is trivially achieved using a strict XAI policy with $\xi_{\text{strict}} = 1$. But the strict full explanations do not maximize welfare. Intuitively, welfare and fairness are best balanced when firms use different XAI methods and offer symmetric XAI $\xi_i = \xi_{\text{fair}}^* \in [\xi_{\text{strict}}^*, 1]$.²³

Unregulated Policy: In the scenario where firms employ the same XAI methods, we identified an exogenous parameter range (β, t) where unregulated firms in equilibrium offer less than full XAI (Proposition 3.1 and Proposition 4.2). Under different XAI methods, we can confirm this result. Consider large t regions with the use of different XAI methods. In an unregulated environment, firms with a choice of continuous quality and XAI choose symmetric XAI levels:

$$\xi_{ug}^* = 1 - \frac{1}{36\beta t}$$

The equilibrium ξ_i^* approaches 1 as t increases. This result of firms mirroring each other's XAI depth when unregulated simplifies the comparison of firm profits under unregulated and strict policies.²⁴ Firms prefer strict policy $\xi_{\text{strict}} = \xi_{ug}^*$ and are worse off for ξ_{strict} above or below ξ_{ug}^* . Under a lower-bounded policy $\xi_{LB} \leq \xi_{ug}^*$, firms choose $\xi_i = \xi_{ug}^*$. With any lower-bound policy $\xi_{LB} > \xi_{ug}^*$, firms will be worse off relative to an unregulated environment.

²³In Appendix A5 we discuss XAI fairness in more detail.

²⁴But while mirroring each other's XAI *levels* leads to a symmetric equilibrium in this market, mirroring each other's XAI *method* results in no equilibrium at all.

5.2 Self-Regulating XAI Policy

Recall that under the same XAI method and small t regions, we discussed an equilibrium in which firm 1 (high quality) opts out to sacrifice its XAI while firm 2 (lower quality) opts in to receive some XAI advantage. We now consider large t regions and different XAI methods.

Large t and Same XAI Method: Firms differentiate on XAI depth i.e., firm 1 opts in while firm 2 does not. Thus, self-regulating XAI introduces a measure of horizontal differentiation even though firms are located at the same point on the Hotelling line. Although firms are undifferentiated, XAI by one firm helps both firms earn higher profits. Interestingly, firm 1 (the high-quality firm) lowers its quality as ξ increases, while firm 2 (the low-quality firm) does the opposite. The intuition is that: when firms are flexible to differentiate on a dominant dimension ($\Delta\xi$) and secondary dimension (Δq), they differentiate maximally only on the dominant dimension. Profits increase with quality for both firms ($\partial\pi_i/\partial q_i > 0$, $i = 1, 2$), leading them to choose almost identical qualities if the cost of quality approaches zero,²⁵ hence a reduction in the quality gap and profits. Comparing small t and large t regions, **quality and XAI are substitutes** in helping firms weaken competition and make positive profits.

Large t and Different XAI Method: We find that no firm has the incentive to offer explanations at low levels ($\xi < 1 - \frac{1}{36\beta t}$), but when ($\xi > 1 - \frac{1}{36\beta t}$) Firm 1 (the high-quality firm) opts in. There are multiple forces at play. As the level of explanation ξ increases, the XAI depth of differentiation increases. This weakens competition by creating greater differentiation in XAI, but it also reduces the Hotelling line distance between the two firms and therefore increases competition. Moreover, there is an indirect effect through quality. Firm 1 (the high-quality firm) opts in at ($\xi_1 = \xi_{SR}$) and offers a higher level of quality compared to a baseline in which no firm offers explanations. Firm 2 (the low-quality firm) responds by lowering its quality. Consequently, the quality gap grows in ξ and contributes to greater profits. As Firm 1 (Firm 2) increases (decreases) its quality level, it charges a higher price than Firm 2. Firm 1 captures more customers while also increasing its price, whereas Firm 2 loses its market share while lowering its price. Unlike the scenario where firms use the same XAI method, **quality and XAI are complementary**. We show that with Firm 1 offering XAI, Firm 2 cannot do any better by adopting XAI, ruling out the

²⁵Which is not possible because of the equilibrium condition ($\beta t\xi > 2/9$).

$(\xi_1^*, \xi_2^*) = (\xi, \xi)$ strategy.

Lemma 5.1. *Under self-regulating XAI policy $\xi_i \in \{0, \xi_{SR}\}$ and large t , regardless of XAI methods, at most one firm opts in resulting in a partial XAI equilibrium.*

Furthermore, we can extend this result beyond the objective of total welfare W (or consumer surplus U). In practice, the policymaker may find it challenging to measure total welfare. For political or PR reasons, the policymaker may seek to maximize an objective that is easier to measure and publicize, such as the total number of firms that opt-in to offer XAI or the average depth of explanations received by the customer. We find that when firms use the same (different) XAI method, the number of firms that offer XAI is constant or decreasing (increasing) in ξ . In summary, under self-regulating XAI, the number of firms that offer XAI or the average XAI depth is not strictly increasing in ξ_{SR} . Consequently, we show that $\xi_{SR} = 1$ does not maximize the number of firms offering XAI or average XAI depth.

Finally, we compare self-regulating and strict policies. We confirm that for small βt , there is no additional total welfare from the strict XAI policy over the self-regulation policy. We also have an interesting counterintuitive result that the policymaker’s choice of an optimal strict XAI depth may be lower than the optimal self-regulating level ($\xi_{strict}^* < \xi_{SR}^*$). Consider, for instance, the case of different XAI methods under high βt . On one hand, under self-regulating XAI, the policymaker asks for full explanations ($\xi_{SR}^* = 1$) but *only one* firm will choose to offer XAI. On the other hand, if XAI is strict, the policymaker asks for partial explanations ($\xi_{strict}^* < 1$) but *both* firms comply. Thus, one setting has a higher explanation depth while the other has a greater number of firms that offer XAI.

6 Conclusion

This research paper offers a deep dive into the complex dynamics of eXplainable Artificial Intelligence (XAI) regulations and their impact on firm competition and social welfare. The study’s key finding is a nuanced departure from the common assumption that regulating an AI product to full explanation is preferred. The finding that a full XAI explanation is not a recommended regulatory strategy holds for a variety of XAI policy levers (strict, self-regulating, and lower bound), regardless of firms’ choice of XAI methods (same or different) and variety of policy objectives (e.g., welfare,

consumer surplus, average XAI depth).

We are also able to compare across XAI policy levers. While strict XAI policies ensure a uniform explanation depth across the market, they potentially limit firms’ capacity for differentiation and innovation in their XAI offerings. Unregulated XAI, while offering maximum flexibility, may fail to guarantee a minimum explanation depth for all consumers. We introduce flexible policies of self-regulating XAI and lower bounded XAI that result in higher consumer welfare than the extremes of unregulated or full XAI policies. In summary, this research urges policymakers to consider a more nuanced approach when crafting XAI regulations. A one-size-fits-all policy (across all markets), particularly one mandating full explanation, may not yield the desired outcomes.

Future research could examine different aspects of XAI, such as the number of explained features or the distinction between inferential and counterfactual explanations. It would be beneficial to investigate the dynamics of XAI in scenarios where consumers may strategically manipulate AI decision-makers, which could influence firms’ willingness to provide full algorithmic transparency. Subsequent studies could also empirically validate our theoretical assumptions (e.g., order for strategic firm choice of XAI method, depth and quality) as well as findings in real-world settings. Finally, while our model excludes factors such as adversarial attacks and model privacy issues, examining their role in the practical implementation of XAI could offer valuable insights into regulatory strategies.

7 Funding and Competing Interests

The authors have no funding or conflicts of interest to report.

8 References

- [1] Abdollahi, B., NaSRaoui, O., 2016. Explainable Restricted Boltzmann Machines for Collaborative Filtering. ArXiv160607129 Cs Stat.
- [2] Adadi, A., Berrada, M., 2018. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). IEEE Access 6, 52138–52160. <https://doi.org/10.1109/ACCESS.2018.2870052>
- [3] Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., Herrera, F., 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges

- toward responsible AI. *Inf. Fusion* 58, 82–115. <https://doi.org/10.1016/j.inffus.2019.12.012>
- [4] Bertini, M., Koenigsberg, O., 2021. The Pitfalls of Pricing Algorithms. *Harv. Bus. Rev.*
 - [5] Bhatt, U., Xiang, A., Sharma, S., Weller, A., Taly, A., Jia, Y., Ghosh, J., Puri, R., Moura, J.M.F., Eckersley, P., 2019. Explainable Machine Learning in Deployment.
 - [6] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D., 2020. Language Models are Few-Shot Learners. <https://doi.org/10.48550/arXiv.2005.14165>
 - [7] Castelvechi, D., 2016. Can we open the black box of AI? *Nature* 538, 20–23. <https://doi.org/10.1038/538020a>
 - [8] Csiszár, O., Csiszár, G., Dombi, J., 2020. Interpretable neural networks based on continuous-valued logic and multicriteria decision operators. *Knowl.-Based Syst.* 199, 105972. <https://doi.org/10.1016/j.knosys.2020.105972>
 - [9] Cunha, M., Osório C., A.M., Ribeiro, R.M., 2020. Endogenous Product Design and Quality When Consumers Have Heterogeneous Limited Attention (SSRN Scholarly Paper No. 2860456). Social Science Research Network, Rochester, NY. <https://doi.org/10.2139/ssRn.2860456>
 - [10] Datta, Amit, Tschantz, M.C., Datta, Anupam, 2015. Automated Experiments on Ad Privacy Settings: A Tale of Opacity, Choice, and Discrimination. <https://doi.org/10.48550/arXiv.1408.6491>
 - [11] DeMarzo, P. M., Fishman, M. J., & Hagerty, K. M. (2005). Self-Regulation and Government Oversight. *The Review of Economic Studies*, 72(3), 687–706.
 - [12] Doshi-Velez, F., Kim, B., 2017. Towards A Rigorous Science of Interpretable Machine Learning. ArXiv170208608 Cs Stat.
 - [13] Fu, R., Jin, G.Z., Liu, M., 2022. Human-Algorithm Interactions: Evidence from Zillow.com. Working Paper Series. <https://doi.org/10.3386/w29880>
 - [14] Galli, A., Marrone, S., Moscato, V., Sansone, C., 2021. Reliability of eXplainable Artificial Intelligence in Adversarial Perturbation Scenarios, in: Del Bimbo, A., Cucchiara, R., Sclaroff, S., Farinella, G.M., Mei, T., Bertini, M., Escalante, H.J., Vezzani, R. (Eds.), *Pattern Recognition. ICPR International Workshops and Challenges, Lecture Notes in Computer Science*. Springer International Publishing, Cham, pp. 243–256. https://doi.org/10.1007/978-3-030-68796-0_18
 - [15] Goodfellow, I.J., Shlens, J., Szegedy, C., 2015. Explaining and Harnessing Adversarial Examples. ArXiv14126572 Cs Stat.
 - [16] Goodman, B., Flaxman, S., 2017. European Union Regulations on Algorithmic Decision-Making and a “Right to Explanation.” *AI Mag.* 38, 50–57. <https://doi.org/10.1609/ai mag.v38i3.2741>

- [17] Hardt, M., Price, E., Srebro, N., 2016. Equality of Opportunity in Supervised Learning. <https://doi.org/10.48550/arXiv.1610.02413>
- [18] Holzinger, A., Plass, M., Holzinger, K., Crisan, G.C., Pintea, C.-M., Palade, V., 2017. A glass-box interactive machine learning approach for solving NP-hard problems with the human-in-the-loop. ArXiv170801104 Cs Stat.
- [19] Kuppa, A., Le-Khac, N.-A., 2020. Black Box Attacks on Explainable Artificial Intelligence(XAI) methods in Cyber Security, in: 2020 International Joint Conference on Neural Networks (IJCNN). Presented at the 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. <https://doi.org/10.1109/IJCNN48605.2020.9206780>
- [20] Law Library of Congress (U.S.), G.L.R.D., , 2019. Regulation of artificial intelligence in selected jurisdictions. <https://purl.fdlp.gov/GPO/gpo123733>
- [21] Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R.J., Wasserman, L., 2017. Distribution-Free Predictive Inference For Regression. ArXiv160404173 Math Stat.
- [22] Lipton, Z.C., 2017. The Mythos of Model Interpretability. ArXiv160603490 Cs Stat.
- [23] Liu, B., Wei, L., 2021. Machine gaze in online behavioral targeting: The effects of algorithmic human likeness on social presence and social influence. Comput. Hum. Behav. 124, 106926. <https://doi.org/10.1016/j.chb.2021.106926>
- [24] Lundberg, S.M., Lee, S.-I., 2017. A Unified Approach to Interpreting Model Predictions, in: Advances in Neural Information Processing Systems. Curran Associates, Inc.
- [25] Malik, N., 2020. Does Machine Learning Amplify Pricing Errors in Housing Market?: Economics of ML Feedback Loops. <https://doi.org/10.2139/ssRn.3694922>
- [26] Martens, D and Huysmans, J and Setiono, R and Vanthienen, J and Baesens, B, Rule Extraction from Support Vector Machines: An Overview of Issues and Application in Credit Scoring, Studies in Computational Intelligence (SCI), 80, pp 33–63 (2008)
- [27] Martens, D, Provost, F, Explaining data-driven document classifications, MIS Quarterly, 38(1), pp 73–100 (2014)
- [28] Miller, T., 2019. Explanation in artificial intelligence: Insights from the social sciences. Artif. Intell. 267, 1–38. <https://doi.org/10.1016/j.artint.2018.07.007>
- [29] Mittelstadt, B., Russell, C., Wachter, S., 2019. Explaining Explanations in AI, in: Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* '19. Association for Computing Machinery, New York, NY, USA, pp. 279–288. <https://doi.org/10.1145/3287560.3287574>
- [30] Murdoch, W.J., Singh, C., Kumbier, K., Abbasi-Asl, R., Yu, B., 2019. Interpretable machine learning: definitions, methods, and applications. Proc. Natl. Acad. Sci. 116, 22071–22080. <https://doi.org/10.1073/pnas.1900654116>
- [31] Netzer, O., Lemaire, A., Herzenstein, M., 2019. When Words Sweat: Identifying Signals for Loan Default in the Text of Loan Applications. J. Mark. Res. 56, 960–980. <https://doi.org/10.1177/0022243719852959>

- [32] Orekondy, T., Schiele, B., Fritz, M., 2019. Knockoff Nets: Stealing Functionality of Black-Box Models. Presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4954–4963.
- [33] Patel, N., Shokri, R., Zick, Y., 2022. Model Explanations with Differential Privacy, in: 2022 ACM Conference on Fairness, Accountability, and Transparency, FAccT '22. Association for Computing Machinery, New York, NY, USA, pp. 1895–1904. <https://doi.org/10.1145/3531146.3533235>
- [34] Pollack, R., 2016. Opening the Black Box: In Search of Algorithmic Transparency. <https://doi.org/10.2139/ssRn.3837723>
- [35] Pradhan, R., Lahiri, A., Galhotra, S., Salimi, B., 2022. Explainable AI: Foundations, Applications, Opportunities for Data Management Research, in: 2022 IEEE 38th International Conference on Data Engineering (ICDE), pp. 3209–3212. <https://doi.org/10.1109/ICDE53745.2022.00300>
- [36] Ramon, Y., Vermeire, T., Toubia, O., Martens, D., & Evgeniou, T. (2021). Understanding consumer preferences for explanations generated by XAI algorithms. arXiv preprint arXiv:2107.02624.
- [37] Ribeiro, M.T., Singh, S., Guestrin, C., 2016. “Why Should I Trust You?”: Explaining the Predictions of Any Classifier. Proceedings of the 22nd ACM SIGKDD, pp. 1135–1144.
- [38] Roth, A.E. (Ed.), 1988. The Shapley Value: Essays in Honor of Lloyd S. Shapley. Cambridge University Press, Cambridge. <https://doi.org/10.1017/CB09780511528446>
- [39] Rudin, C., 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat. Mach. Intell. 1, 206–215. <https://doi.org/10.1038/s42256-019-0048-x>
- [40] Shapley, L.S., 1951. Notes on the N-Person Game — II: The Value of an N-Person Game. RAND Corporation, Santa Monica, CA. <https://doi.org/10.7249/RM0670>
- [41] Tirole, J. (1988). The theory of industrial organization. MIT press.
- [42] Ukanwa, K., Rand, W., Zubcsek, P.P., 2022. Algorithmic Fairness and Service Failures: Why Firms Should Want Algorithmic Accountability. <https://doi.org/10.2139/ssRn.4148214>
- [43] Yang, C., Rangarajan, A., Ranka, S., 2018. Global Model Interpretation via Recursive Partitioning. ArXiv180204253 Cs Stat.
- [44] Zhou, J., Chen, F., Holzinger, A., 2022. Towards Explainability for AI Fairness, in: Holzinger, A., Goebel, R., Fong, R., Moon, T., Müller, K.-R., Samek, W. (Eds.), XxAI - Beyond Explainable AI: International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers, Lecture Notes in Computer Science. Springer International Publishing, Cham, pp. 375–386. https://doi.org/10.1007/978-3-031-04083-2_18

Explaining Large Language Models Decisions Using Shapley Values

Behnam Mohammadi

First Version: March 29, 2024

Latest Update: November 11, 2024

Abstract

The emergence of large language models (LLMs) has opened up exciting possibilities for simulating human behavior and cognitive processes, with potential applications in various domains, including marketing research and consumer behavior analysis. However, the validity of utilizing LLMs as stand-ins for human subjects remains uncertain due to glaring divergences that suggest fundamentally different underlying processes at play and the sensitivity of LLM responses to prompt variations. This paper presents a novel approach based on Shapley values from cooperative game theory to interpret LLM behavior and quantify the relative contribution of each prompt component to the model’s output. Through two applications—a discrete choice experiment and an investigation of cognitive biases—we demonstrate how the Shapley value method can uncover what we term the “token noise” effect, a phenomenon where LLM decisions are disproportionately influenced by tokens providing minimal informative content. This finding raises concerns about the robustness and generalizability of insights obtained from LLMs in the context of human behavior simulation. Our model-agnostic approach extends its utility to proprietary LLMs, providing a valuable tool for practitioners and researchers to strategically optimize prompts and mitigate apparent cognitive biases. Our results underscore the need for a more nuanced understanding of the factors driving LLM responses before relying on them as substitutes for human subjects in survey settings. We emphasize the importance of researchers reporting results conditioned on specific prompt templates and exercising caution when drawing parallels between human behavior and LLMs.

Keywords: Large Language Models (LLMs), Shapley Value, Machine Learning, Interpretability, eXplainable AI (XAI), Cognitive Biases

1. Introduction

The rapid advancement of large language models (LLMs) has opened up exciting possibilities for understanding and simulating human behavior, with potential applications in various domains, including economic surveys, marketing research, and consumer behavior analysis. The appeal of utilizing LLMs for practitioners and researchers lies in their potential to exhibit reasoning patterns akin to humans. If LLMs can indeed demonstrate such capabilities, they could function as highly scalable proxies to traditional resource-intensive survey studies. Armed with the ability to rapidly collect “human-like” data through LLM simulations, marketers could significantly accelerate research cycles by testing different product configurations, pricing strategies, or advertising messages on virtual consumers before launching them in the real world. However, to harness the full potential of LLMs in business applications, we must first understand the factors that drive their behavior and the extent to which they truly reflect human cognitive processes.

While LLMs have been shown to sometimes mirror documented cognitive biases found in human psychology experiments, there are also glaring divergences that suggest fundamentally different underlying processes at play. Moreover, the sensitivity of LLM responses to variations in input prompts has led to concerns about the robustness and generalizability of the insights obtained from these models. A key challenge lies in interpretability—how can we disentangle the factors truly driving an LLM’s behavior? When an LLM provides an output concordant with an established human bias, is it genuinely simulating the associated cognitive process? Or is the LLM simply locking onto superficial token-level signals unrelated to the core semantics?

To illustrate the concerning fragility of LLM responses, consider a simple discrete choice experiment where we ask an LLM to choose between two flight options, A and B. When presented with the basic prompt, the model assigns certain probabilities to choosing each option based on the logits associated with response tokens. However, a striking phenomenon emerges when we introduce an imperceptible modification to the prompt: the sequential addition of newline characters after the question¹. Despite being invisible to human readers and carrying no semantic meaning, each additional newline character significantly—and unpredictably—alters the choice probabilities.

¹ By repeatedly pressing “Enter/Return” at the end of the question.

Figure 1 illustrates this phenomenon through a series of snapshots showing how the choice probabilities change with different numbers of newlines. With no newlines (top-left), the model strongly prefers option A. However, after adding just 10 newlines (bottom-left), the probability of choosing option B rises notably, only to drop again with 11 newlines (bottom-right). Figure 2 reveals the full extent of this instability by tracking these probabilities across 200 newline insertions. The resulting pattern defies simple explanation: the choice probabilities exhibit chaotic fluctuations, with option B's probability occasionally surging above 0.6 before abruptly falling back near zero. This behavior starkly contrasts with human decision-making, where blank space in a questionnaire would not influence responses. More troublingly, the non-linear and seemingly random nature of these probability shifts means we cannot systematically correct for this type of noise in LLM decisions.

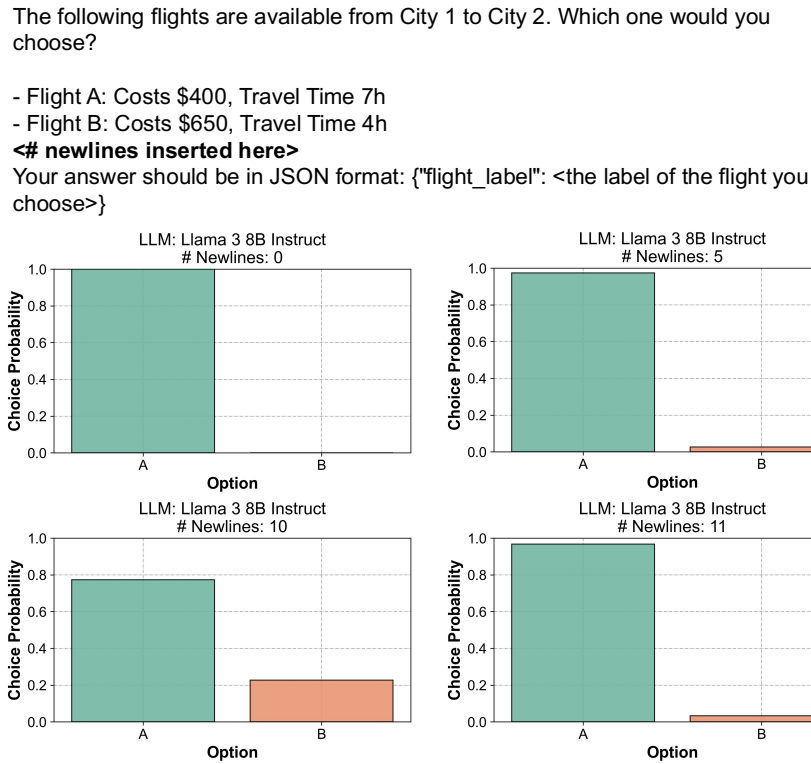


Figure 1. Impact of newline characters on choice probabilities: four snapshots showing how the LLM’s preferences between flight options change with different numbers of newline characters. The model’s strong initial preference for option A (probability ≈ 1.0 with no newlines) becomes unstable as newlines are added, demonstrating the sensitivity of LLM responses to seemingly innocuous formatting changes.

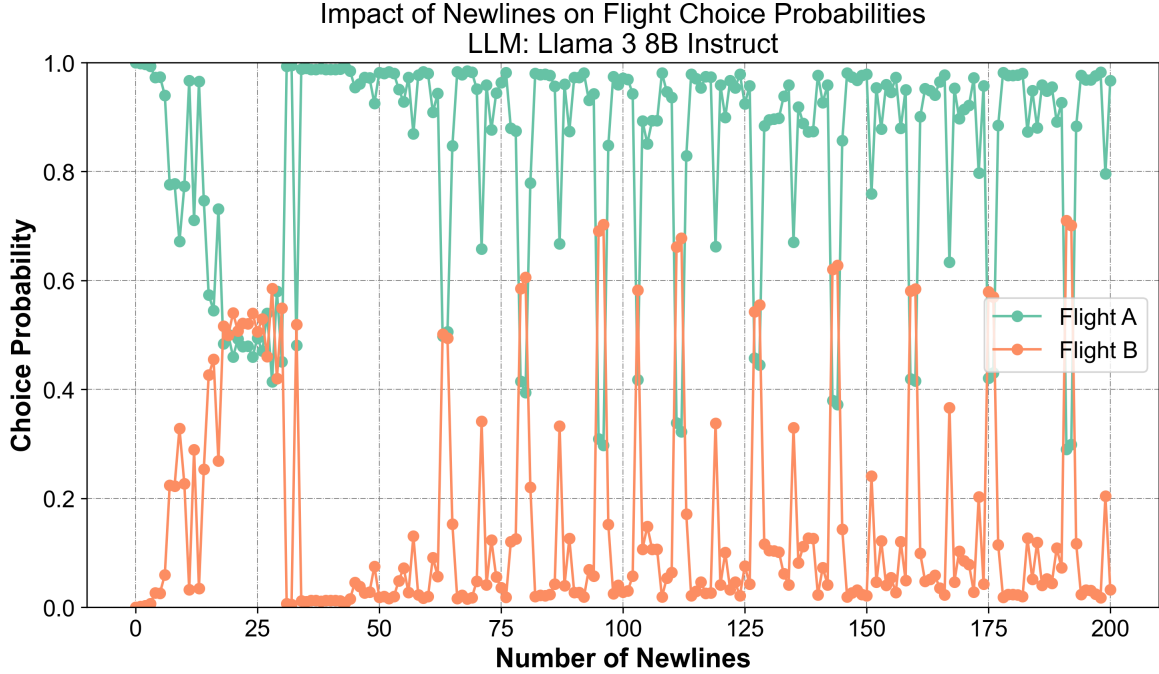


Figure 2. Complete trajectory of choice probabilities for flight options A and B as the number of newline characters increases from 0 to 200. The chaotic fluctuations in probabilities demonstrate the unpredictable impact of newline characters on the LLM’s decision-making process, with no discernible pattern or convergence even after many iterations.

This example underscores a critical reality: every token in a prompt, regardless of its semantic significance, can substantially impact an LLM’s decision-making process. The sensitivity to ostensibly irrelevant tokens raises fundamental questions about the reliability of LLM responses in experimental settings. Furthermore, it highlights the urgent need for interpretability methods that can not only identify how each token contributes to the LLM’s output but also quantify these effects. Our proposed approach, based on Shapley values from cooperative game theory, addresses precisely this challenge.

Our Shapley value method treats the elements of a prompt as “players” in a game and quantifies the relative contribution of each component to the LLM’s output. This approach allows us to identify key tokens that significantly impact the model’s decisions, differentiating between tokens that carry semantic weight and those that do not, which ideally should exert minimal influence on the model’s responses.

The results reveal an even more pervasive phenomenon beyond the newline character effect, which we call “token noise”: the model’s decision is unduly affected by tokens with negligible informational content, such as articles, prepositions, and even single words like “flight”—tokens that provide little to no meaningful content about the choice options

themselves. For instance, in the same flight choice scenario discussed above, while one would intuitively expect the tokens representing dollar amounts and flight durations to be the prime determinants of the LLM’s choice, our findings show otherwise. This token noise effect, combined with the sensitivity to format-level changes like newline characters, casts serious doubt on whether the LLM’s outputs truly reflect an understanding of the semantic content or a capacity for human-like decision-making.

Our proposed Shapley value method diverges significantly from conventional eXplainable AI (XAI) techniques, such as SHAP, commonly used in machine learning. We begin by transforming the prompt into a vector format, representing it as a template with fields that can be substituted with arbitrary values. This template serves as the basis for generating different versions of the prompt. The LLM is then modeled as a function that maps this prompt vector to a probability distribution over possible outputs. Utilizing Shapley values, we compute the contribution of each segment of the prompt to the LLM’s prediction, thereby quantifying the significance of different token groups in shaping the model’s predictions. This stands in stark contrast to SHAP’s approach which considers each token as an isolated feature, lacking the capacity to recognize the semantic interconnections between groups of tokens. Moreover, SHAP’s reliance on multiple data points for comparison with an average input model is impractical for LLM scenarios where typically only a single, specific prompt is analyzed. Our methodology carefully navigates this by using the prompt’s template as a reference, sidestepping the limitations inherent in SHAP’s comparative framework.

Moreover, in this paper the algorithm for estimating Shapley values deviates from traditional approaches like Kernel SHAP. Instead of averaging marginal contributions across random data points, we use a moving average approach that enhances stability and accuracy without needing to store intermediate values, making our estimation technique well-suited for the single prompt vector scenario.

Another key advantage of our Shapley value method is that it is model-agnostic, working on any LLM, even proprietary models that are closed-source and gated behind an API, such as OpenAI’s GPT¹, Google’s Gemini², or AnthropicAI’s Claude³. This

¹ <https://chat.openai.com/>

² <https://gemini.google.com/app>

³ <https://claude.ai/>

universality offers a major benefit over mechanistic interpretability approaches that often require access to the model’s weights.

We demonstrate the effectiveness of our Shapley value method through two applications. In the first, we investigate a discrete choice experiment akin to those used in marketing research studies such as conjoint analyses. Our analysis uncovers the outsized impact of token noise, with the LLM’s choices being heavily influenced by tokens carrying little semantic information about the choice options. Furthermore, we demonstrate how the insights from our Shapley value method can be utilized to make subtle adjustments to the prompt phrasing, resulting in significant changes to the choice probabilities of large language models. This raises concerns about the validity and robustness of using such models as proxies for human subjects.

The second application explores the presence of cognitive biases in LLMs, using the framing effect as an illustrative example. While LLMs exhibit apparent sensitivity to framing, our Shapley value analysis reveals that this behavior is largely an artifact of token noise rather than genuine cognitive processing. Crucially, we showcase how leveraging insights from our method enables practitioners to strategically optimize prompts and mitigate the apparent framing effects, questioning the genuine presence of this bias in LLMs.

Our findings underscore the importance of researchers incorporating Shapley value analysis as a standard practice when studying LLM behavior. This approach contributes to a more nuanced and comprehensive interpretation of research findings, akin to reporting confidence intervals in statistical analyses. Furthermore, we emphasize the necessity of releasing results conditioned on the specific prompt templates used, enabling a thorough evaluation of the impact of prompt formulations.

The remainder of this paper is structured as follows. Section 2 reviews the relevant literature on cognitive biases in LLMs and their potential applications in marketing, highlighting the need for novel interpretability methods. Section 3 introduces our methodology, explaining the Shapley value framework and its implementation. Section 4 presents the results from our two marketing-focused case studies, showcasing the insights gained from the Shapley value analysis. Finally, Section 5 discusses the key conclusions, limitations, and future research directions.

2. Background and Literature Review

The emergence of LLMs has sparked significant interest in their potential to simulate human behavior and cognitive processes. Several researchers have explored the intriguing

possibility of using LLMs as proxies for human subjects in various contexts, including decision-making tasks, surveys, opinion polling, and marketing research. However, a growing body of literature has raised concerns about the presence of cognitive biases in LLMs, casting doubt on their ability to accurately represent human behavior.

Several studies have investigated the presence of cognitive biases and reasoning abilities in LLMs, drawing comparisons to human performance on similar tasks. (Binz and Schulz, 2023) treated GPT-3 as a participant in a psychology experiment, subjecting it to a battery of canonical experiments from the cognitive psychology literature. They found that GPT-3 exhibited behavior similar to humans in tasks related to decision-making, information search, and deliberation. However, the authors also noted that small perturbations to the input prompts could lead to vastly different responses from the model, raising questions about the robustness and consistency of LLM behavior. Similarly, (Shaki et al., 2023) demonstrated that GPT-3 exhibits cognitive effects such as priming, distance, SNARC, and size congruity, but not anchoring. (Jones and Steinhardt, 2022) examined OpenAI’s Codex model and identified biases such as anchoring effects, framing effects, and a tendency to mimic frequent training examples, drawing parallels with human cognitive biases. (Hagendorff and Fabi, 2023) demonstrated that GPT-3 exhibits behavior resembling human-like intuition and the associated cognitive errors, while more advanced models such as ChatGPT and GPT-4 have learned to avoid these errors, performing in a “hyperrational” manner. (Koo et al., 2023) benchmarked LLMs as evaluators and found evidence of cognitive biases, such as the egocentric bias, where models prefer to rank their own outputs highly. (Fuchs et al., 2022) discussed methods that demonstrate assumptions about cognitive limitations and biases in human reasoning, and their potential application to LLMs. (Loya et al., 2023) examined the sensitivity of LLMs’ decision-making abilities to prompt variations and hyperparameters, finding that simple adjustments could lead to human-like exploration-exploitation trade-offs.

(Talboy and Fuller, 2023) expanded on this line of research by examining several documented cognitive biases in LLMs. The authors demonstrated the presence of these biases and discussed the implications of using biased reasoning under the guise of expertise. While their work highlights the potential risks associated with the increasing adoption of LLMs, it relies on the assumption that the presence of human-like biases in LLMs indicates the existence of cognitive processes similar to those of humans.

(Stella et al., 2023) critiqued this assumption, arguing that focusing solely on human-like biases may not be sufficient for understanding the true nature of LLM reasoning. They emphasized the importance of investigating non-human-like biases, such as myopic

overconfidence and hallucinations, to assess knowledge elaboration in LLMs. In the same vein, (Macmillan-Scott and Musolesi, 2024) concluded that while LLMs display irrationality in tasks from cognitive psychology, they do so in ways that differ from human-like biases, with significant inconsistency in their responses. These works underscore the need for a deeper understanding of the processes generating these biases and the potential differences between LLM and human cognition.

Several researchers have explored the potential of using LLMs to simulate human behavior in specific contexts, such as student learning (Xu and Zhang, 2023), multiple human subjects in experiments (Aher et al., 2023), and strategic decision-making (Chen et al., 2023; Sreedhar and Chilton, 2024). However, these studies have also acknowledged the challenges and limitations associated with using LLMs as human proxies. (Tjuatja et al., 2024) found that popular open and commercial LLMs generally failed to reflect human-like response biases in survey questionnaires, particularly models that underwent reinforcement learning from human feedback (RLHF).

The application of LLMs in market research and consumer preference elicitation has also been explored. (Brand et al., 2023) demonstrated that GPT-3.5 responded to survey questions in ways consistent with economic theory and consumer behavior patterns, but acknowledged potential limitations in using LLMs for marketing purposes. The authors suggested that LLMs could be used to understand consumer preferences and estimate willingness-to-pay for products and features.

However, the validity and reliability of using LLMs as proxies for human subjects have been called into question. (Gui and Toubia, 2023) highlighted the challenges of conducting LLM-simulated experiments from a causal inference perspective. They showed that variations in the treatment included in the prompt could cause variations in unspecified confounding factors, introducing endogeneity and yielding implausible demand curves. This endogeneity issue is likely to generalize to other contexts and may not be fully resolved by simply improving the training data.

Some papers have taken a more philosophical or theoretical approach to understanding the implications of cognitive biases in LLMs. (Thorstad, 2023) advocated for cautious optimism about the prevalence of cognitive bias in current models, coupled with a willingness to acknowledge and address genuine biases. (Taubenfeld et al., 2024) observed that LLM agents simulating political debates tended to conform to the model’s inherent social biases, deviating from well-established social dynamics among humans.

The limitations of current approaches to understanding LLM behavior highlight the need for novel methods to interpret LLM responses and uncover the factors driving the model’s outputs. Without a clear understanding of the underlying mechanisms, it is premature to assume that LLMs exhibit genuine cognitive processes akin to those of humans or that they can serve as reliable proxies for human subjects in research settings. The inconsistencies and sensitivity to prompt variations observed in LLM responses suggest that the results obtained from these models may be largely driven by factors other than cognitive processes. In the following section, we propose the use of Shapley values as a method to interpret LLM experiments and shed light on the role of such factors in driving LLM results.

3. Method

3.1. Vectorizing the Prompt

Consider the following prompt which we want to send to an LLM:

```
The following flights are available from City 1 to City 2. Which one would you choose?
- Flight A: Costs $400, Travel Time is 7 hours
- Flight B: Costs $600, Travel Time is 5 hours
```

Let us assume that the LLM responds with “A” and “B” with 80% and 20% probability, respectively¹. Our goal is to find out how much each word or token in the prompt contributes to the choice probabilities².

The first step is to convert the prompt into a (“Jinja”) template³. In this template, some parts of the text (tokens, words, or groups of words⁴) are replaced with handlebar

¹ With local models we use context-free grammars in the LLM engine to constrain the output structure of the model so that it always follows the specified format. With API models (such as OpenAI’s GPT models), we use logit bias or JSON-mode to constrain the output format.

² We do not consider punctuation marks such as “:”, “,”, “-”, etc. The instruction about JSON format is not included either.

³ <https://jinja.palletsprojects.com/en/3.1.x/>

⁴ When thinking about creating the template fields, we can consider putting one token, one word, or a group of words in one field. It is possible to create one field per token, but tokens do not necessarily carry enough semantic information to help make a decision (e.g., the word “flights” under GPT-3.5 and GPT-4 tokenizer is two tokens: “fl” + “ights”, none of which helps even a human in decision-making). Words, on the other hand, are—by definition—semantically meaningful. Depending on the context, we create fields for single (meaningful) tokens, words, and groups of words.

fields like `{{ this }}`. One can substitute the value of a field with arbitrary values in order to generate different versions of the prompt. If we fill out all fields with their corresponding values in the prompt text, we get back the original prompt. A simple template of our prompt with 8 fields is as follows (For the actual full template used in the paper, see Appendix A1.):

The following flights are available from `{{ x_1 }}` to `{{ x_2 }}`. Which one would you choose?

- `{{ x_3 }}`: Costs `{{ x_4 }}`, Travel Time is `{{ x_5 }}` hours
- `{{ x_6 }}`: Costs `{{ x_7 }}`, Travel Time is `{{ x_8 }}` hours

Conditioned on the prompt template, one can now write the prompt as a feature *vector* $\vec{x} = (x_1, \dots, x_I)^T$ where I is the number of fields in the template and each feature x_i is a field value¹. Using the simple template above, the original prompt is then:

$$\vec{x} = (\text{"City 1"}, \text{"City 2"}, \text{"Flight A"}, \$400, 7, \text{"Flight B"}, \$600, 5)^T.$$

3.2. Modeling the LLM

After vectorizing the prompt, the next step is to model the LLM as a function f that maps the prompt \vec{x} to a probability², i.e., makes a prediction based on \vec{x} . Of course, LLMs—like most deep neural networks—are highly nonlinear in their input arguments, so f cannot simply be a linear function of \vec{x} . Instead, a common approach in the machine learning literature is to think of f as a linear function of the *attributions* of its input \vec{x} —denoted by $\vec{\phi} = (\phi_1, \dots, \phi_I)^T$ —where each ϕ_i is the attribution of the corresponding x_i . This aligns well with our goal to find the contribution of each (prompt) feature x_i to the prediction. Therefore, f can be written as:

$$f(\vec{x}) = \phi_0 + \sum_{i=1}^I \phi_i \tag{3.1}$$

where ϕ_0 is a baseline (reference) attribution. Note that the linearity of Eq. 3.1 does *not* imply that the LLM is a linear function of the prompt, but rather the LLM is a linear function of the *attributions* of the prompt³. To calculate $\vec{\phi}$, we follow the approach in

¹ Notation: $(\dots)^T$ means the transpose of (\dots) .

² Actually the arguments of f are the prompt template as well as the prompt vector \vec{x} , so a more precise notation would be $f(\vec{x}|\text{template})$, but for simplicity we just write $f(\vec{x})$.

³ And the attributions can be highly nonlinear because they are based on LLM outputs.

(Ancona et al., 2018) and we employ Shapley values, a solution concept in cooperative game theory. This is discussed next.

3.3. Shapley Value Attributions

Remember that our goal is to find feature attributions $\vec{\phi}$ for the given prompt vector \vec{x} . Think of $\mathcal{X} := \{x_1, \dots, x_I\}$ as players in a cooperative (or “coalitional”) game with a payoff for the entire coalition. The payoff is characterized by a “value” function¹ v that maps any subset of \mathcal{X} to \mathbb{R} or a subset of it². In this paper, v is the probability of the next token³ predicted by the LLM compared to some baseline ϕ_0 which we will define later. In the context of coalitional games, any subset $\mathcal{S} \subseteq \mathcal{X}$ is called a “coalition” of players and $v(\mathcal{S})$ is the payoff of the coalition. Thus, our question in cooperative game theory terms is: *How should we distribute the payoff among the players in the coalition?* Shapley value offers a solution that guarantees “fairness” in the sense that each player x_i (i.e., each prompt field) receives a payoff commensurate to its contribution to the coalition⁴. The Shapley value of player x_i under the value function v is denoted by $\phi_i(v)$ and is given by:

$$\phi_i(v) = \mathbb{E}_{O \sim \pi(\mathcal{X})}[v(\text{pre}_i(O) \cup \{x_i\}) - v(\text{pre}_i(O))] \quad 3.2$$

where $\pi(\mathcal{X})$ is the ordered set of permutations of \mathcal{X} , O is an ordering randomly sampled from $\pi(\mathcal{X})$, and $\text{pre}_i(O)$ is the set of players that precede player x_i in O . Therefore, $\phi_i(v)$

¹ v is a “set function”. A set function is a type of function where the domain consists of sets rather than individual elements. Unlike a “normal” function that maps individual elements of one set to another set (e.g., $f: \mathcal{X} \rightarrow \mathcal{Y}$, where each $x \in \mathcal{X}$ is mapped to a unique $y \in \mathcal{Y}$), a set function operates on sets themselves. In our definition of value function, $v: 2^{\mathcal{X}} \rightarrow \mathbb{R}$ where $2^{\mathcal{X}}$ denotes the power set of \mathcal{X} , i.e., the set of all subsets of \mathcal{X} including the empty set and \mathcal{X} itself.

² $v(\emptyset)$ is normalized to zero.

³ Since we study prompts with two options (e.g., Flight A and Flight B), we can simply consider $\mathbb{P}(A)$ as the LLM prediction. LLMs typically first provide a probability distribution for the next token and then draw a sample from it. Since we can directly obtain the probability distribution, there is no need to actually sample from it.

⁴ To be more precise, Shapley value is a “fair” distribution because it is the only distribution that satisfies the following four constraints at the same time (see Appendix A2 for more details): (1) Efficiency: All payoff is divided among players; (2) Symmetry: Payoff is split equally among players who contribute the same value in every coalition; (3) Additivity (Linearity): If there are two independent games (with value functions v and w , respectively), then each player’s payoff is the sum of her payoffs in each game; (4) Null (Dummy) Player: A player who adds no additional value to any coalition receives a payoff of zero.

can be thought of as the expected marginal contribution of player x_i when players are added to a coalition in a random order. Eq. 3.2 can be written as:

$$\begin{aligned}\phi_i(v) &= \frac{1}{N!} \sum_{\pi(x)} [v(\text{pre}_i(O) \cup \{x_i\}) - v(\text{pre}_i(O))] \\ &= \frac{1}{N!} \sum_{\mathcal{S} \subseteq \mathcal{X} \setminus \{x_i\}} [|\mathcal{S}|! (N - |\mathcal{S}| - 1)! [v(\mathcal{S} \cup \{x_i\}) - v(\mathcal{S})]]\end{aligned}\tag{3.3}$$

That is, the Shapley value of player x_i is the average of player x_i 's contribution to each coalition \mathcal{S} weighted by $|\mathcal{S}|! (N - |\mathcal{S}| - 1)!$, the number of permutations in which the coalition can be formed. Before we can calculate Shapley values, though, we need to choose the baseline ϕ_0 and clarify the value of $v(\mathcal{S})$. This is done below.

3.4. Baseline (Reference) Point ϕ_0

Some XAI methods such as SHAP (Lundberg and Lee, 2017) choose the average of all model predictions $\mathbb{E}_{\mathbf{X}}[f(\vec{x})]$ as the baseline of their explanations, where $\mathbf{X} = \{\vec{x}_1, \dots, \vec{x}_N\}$ is the dataset of all feature vectors. But here we have only one prompt vector \vec{x} , so we follow the game formulation in (Merrick and Taly, 2020) which unifies existing methods and allows us to think of Shapley values as *contrastive explanations* of an input \vec{x} relative to one or several reference vectors denoted by \vec{r} . To do this, for any coalition \mathcal{S} , we create a composite input $\vec{z}(\vec{x}, \vec{r}, \mathcal{S})$ (See Figure 3):

$$\vec{z}(\vec{x}, \vec{r}, \mathcal{S}) := (z_1, \dots, z_d)^T \text{ where } z_i = x_i \text{ if } x_i \in \mathcal{S} \text{ and } r_i \text{ otherwise} \tag{3.4}$$

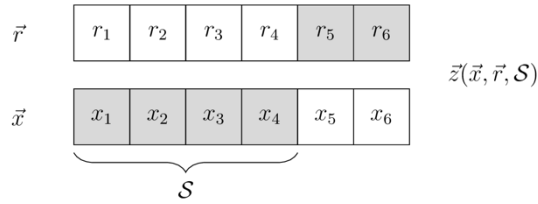


Figure 3. A composite input \vec{z} is an input that agrees with \vec{x} on features in \mathcal{S} and with \vec{r} on all the other features.

In this paper, we let $r_i \in \vec{r}$ be “ _ ”, therefore, any time a field is absent in the prompt, we substitute its template field with a space followed by an underscore and another space. According to (Merrick and Taly, 2020), we can now formulate the value function as:

$$v(\mathcal{S}) = f(\vec{z}(\vec{x}, \vec{r}, \mathcal{S})) - f(\vec{r}) \tag{3.5}$$

with $v(\emptyset) = 0$ and $v(\mathcal{X}) = f(\vec{x}) - f(\vec{r})^1$. Eq. 3.5 means that the value we want to distribute among members of coalition \mathcal{S} is the difference between the probability of the next token when items in \mathcal{S} are present in the prompt (while others are filled out with “_”), and the probability of the next token when all prompt template fields are filled out with “_”. Plugging Eq. 3.5 in Eq. 3.3 now gives us the exact Shapley values.

3.5. Estimating Shapley Values

Calculating the exact Shapley values using Eq. 3.3 can be computationally infeasible for LLM prompts (see Appendix A3 for details). In practice, there exist approximation methods such as (Castro et al., 2009; Štrumbelj and Kononenko, 2014) that use Monte Carlo simulation to estimate Shapley values. The algorithm in this paper follows (Grah and Thouvenot, 2020) which is inspired by the general idea of (Štrumbelj and Kononenko, 2014), but deviates from it in important ways. First, (Štrumbelj and Kononenko, 2014) estimate Shapley values by calculating the marginal contributions compared to random instances in the dataset \mathbf{X} . But since we have only one prompt vector, marginal contributions must be calculated compared to a reference vector \vec{r} . Second, (Štrumbelj and Kononenko, 2014) find the average of all marginal contributions of feature x_i and report that as its Shapley value, while the algorithm in this paper uses a moving average throughout the entire algorithm, making the estimation more stable and accurate with each iteration without needing to store all intermediate values². Next, we present our main results.

4. Applications

In this section we present two applications of our Shapley value method in LLM experiments to demonstrate how LLM responses can be interpreted. While we provide the following examples, bear in mind that our method is general and model-agnostic (a desirable property in XAI methods), meaning that it can be used on any experiment with LLMs of any kind, even the ones whose weights are not open-sourced (i.e., models gated behind an API, such as OpenAI’s GPT models).

¹ This is easy to verify: For $\mathcal{S} = \emptyset$, we have $\vec{z}(\vec{x}, \vec{r}, \emptyset) = \vec{r}$ and for $\mathcal{S} = \mathcal{X}$, we get $\vec{z}(\vec{x}, \vec{r}, \mathcal{X}) = \vec{x}$. Also, notice that $f(\vec{r})$ in Eq. 3.5 is an offset term to ensure that the payoff for the empty set is zero.

² See Appendix A4 for details of the algorithm.

4.1. Application 1: A Discrete Choice Experiment

This is a basic application of our method in which we ask the prompt mentioned in Section 3.1 from the language model. Studying discrete choice experiments of this kind is worthwhile due to the recent interest in substituting human subjects in marketing research studies (e.g., conjoint analyses) with LLM agents. To ensure the robustness of our findings, we conduct the discrete choice experiment on three Llama-2-Chat models of varying sizes: 7B, 13B, and 70B parameters (see Appendix A5.1). By testing our approach on models with different capacities, we aim to validate the consistency of our results and demonstrate the generalizability of our method across LLMs of different scales.

Important to note is that we categorize prompt template fields into two groups: (1) Those that carry information critical in decision-making, (2) and the non-critical ones. For example, when choosing between two flights as in Section 3.1, the price, travel time, and label of flight options is crucial in making an informed decision. On the other hand, most parts of speech (e.g., articles, prepositions, conjunctions, pronouns, auxiliary verbs, etc.) are non-critical for decision-making¹. The high-information tokens determine the *semantics/content* of the prompt whereas the low-information tokens form the general *syntax/structure* of it and changing them does not alter the meaning of the prompt.

LLMs are designed to pay attention to the entire batch of input tokens (Vaswani et al., 2017), but each token may have a varying impact on the model’s prediction of future tokens. If it turns out that the LLM’s decision (flight “A” or “B”) is heavily driven by low-information tokens, that would cast doubt on the validity of the experiment conducted on the LLM. Put differently, it prompts us to question whether the experiment results truly reflect the underlying behavior of the LLM or if they are merely influenced by what we refer to as “**token noise**”. To answer this question, we calculate the Shapley values according to our algorithm mentioned in Appendix A4. Depending on the direction of impact, Shapley values can take positive or negative values. $\phi_i > 0$ ($\phi_i < 0$) implies that feature x_i pushes (pulls) the choice probability to be higher (lower) than the baseline ϕ_0 —the choice probability when the empty prompt template is given to the LLM. Since we

¹ For instance, asking “Which flight *would* you choose?” instead of “Which flight *do* you choose?” should not matter as much as changing the price of flight A.

are concerned only with the magnitude of impact, we take the absolute values of Shapley values and then normalize¹.

Our findings provide evidence that large LLMs, regardless of their size, are “smart” enough to pay attention to high-information tokens, **but we also discover noticeable traces of token noise among all the LLMs we studied** (See Figure 4 which illustrates our results for Llama-2-70B-Chat, the largest LLM among the three²). The patterns of token noise remain consistent across the 7B, 13B, and 70B models, indicating that the observed token noise is not specific to a particular model size but rather a general characteristic of LLMs in this context.

Let us first consider the model’s “smartness”: Notice how in Figure 4 the LLM pays relatively small attention to the first part of the prompt³ while correctly focusing on the second part which contains the information about the flights. The Shapley values for high-information tokens such as the price, travel time, and label of flights (“A” and “B”) are relatively high⁴, meaning that they contribute substantially to the LLM’s decision. In addition, some low-information tokens such as “flights”, “Flight”, “Costs”, and “Travel Time” contribute a great amount by providing context. For example, by reading “The following flights...”, the model knows that what comes next has something to do with flights. Similarly, “Costs”, “Travel Time”, “is”, and “hours” link the words in the prompt and thus have high Shapley values.

¹ For the sake of brevity, from this point forward, we will use the term “Shapley values” to refer to normalized absolute Shapley values.

² Appendix A5.1 lists our results for all three models (Llama-2-Chat 7B, 13B, and 70B).

³ This part of the prompt says: “The following flights are available from City 1 to City 2. Which one would you choose?”

⁴ Flight labels (“A” and “B”) are important because the model is asked to choose a flight between the provided options. Therefore, it must know the label of each option.

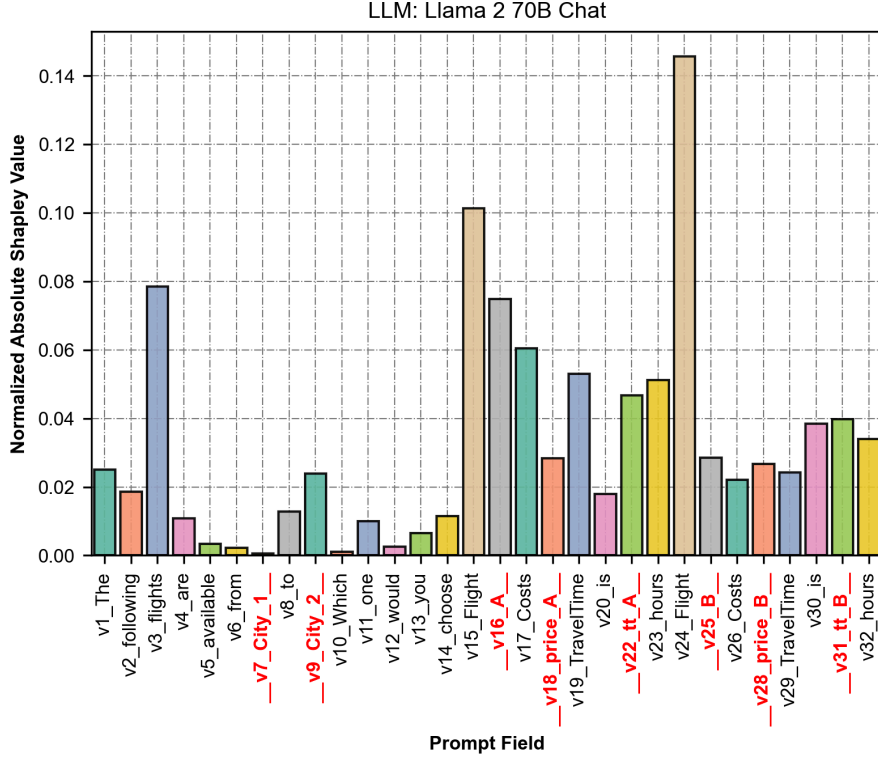


Figure 4. Normalized absolute Shapley values for a discrete-choice question

That being said, the stark difference between the Shapley values of low-information tokens versus the high-information ones is concerning. The fact that the highest Shapley values belong to {“flights”, “Flight [A]”, “Flight [B]”} indicates that the model’s decision—choosing flight “A” or “B”—is mostly swayed by words that do not actually provide any details about the flight options¹. Moreover, the very first word (“The”) is almost as important in the LLM’s decision-making as is the price of flight “B”. This phenomenon—which we refer to as *token noise*—can be exploited to influence the decision of the LLM. For example, consider Figure 5 which shows the choice probabilities of the model before and after we slightly tweak its prompt. As we can see, the choice probability of flight “B” **almost doubles** even though the semantics of the two prompts are identical. Results like this raise concerns about the validity and robustness of using LLMs instead

¹ Consider the word “Flight” as indicated by v24_Flight on Figure 4. Its unnormalized Shapley value is ≈ -0.13 , meaning that the presence of this word reduces the probability of selecting flight “A” by 13% (as compared with the baseline ϕ_0).

of humans in marketing research studies because the estimated coefficients of the utility function could be biased. We will return to this point in Section 5.

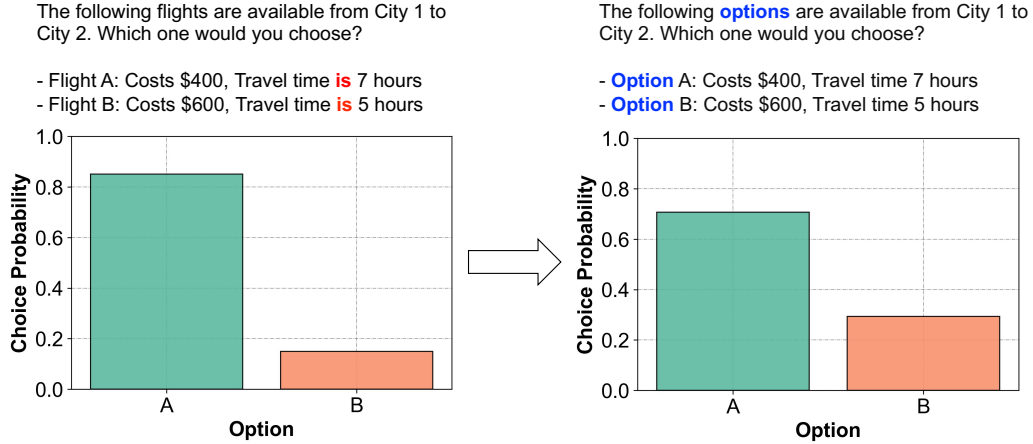


Figure 5. Influencing the LLM’s choice probabilities by slightly rephrasing the prompt. (LLM: Llama-2-70B-Chat)
 Red Text: Tokens removed in the other prompt. Blue Text: Modified tokens.

4.2. Application 2: Existence of Cognitive Biases in LLMs

In our second application, we explore the presence of cognitive biases in LLMs. Understanding the behavioral tendencies of LLMs is crucial for marketers for two reasons: (1) It allows for harnessing the full potential of LLMs when deployed in production (For instance, an LLM that is sensitive to “framing effect” may require careful prompting when used in a recommendation system.), and (2) LLMs that exhibit human-like behavior can serve as valuable test environments for consumer behavior experts to conduct preliminary studies before conducting their main experiments.

Let us consider the framing effect. The concept was first introduced in (Kahneman and Tversky, 1979). The authors illustrated that the choices individuals make are significantly influenced by the manner in which problems are framed. This is primarily due to the inherent cognitive bias towards avoiding risk in gains and seeking risk in losses. To test this effect on LLMs, we conduct an experiment in two stages as follows: First, we use the prompt used in Section 4.1 to find the choice probabilities when no framing effect is introduced. Then, we modify the prompt by adding the word “only” before the price of flight B, i.e., “... *Flight B: Costs only \$600, Travel Time is 5 hours...*”. Semantically speaking, this positive framing makes flight B more attractive than before, as evident in the significant change in the LLM’s choice probabilities (Figure 6):

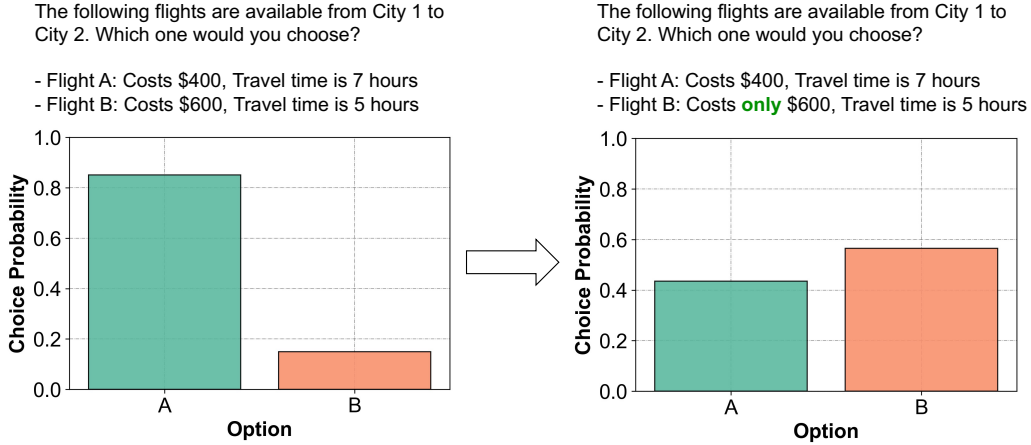


Figure 6. Positively framing an option makes the LLM choose it more than before. (LLM: Llama-2-70B-Chat)
Green Text: Injected tokens.

Taking it at face value, the observed shift in choice probabilities may indicate that LLMs are sensitive to changes in framing and can exhibit patterns that mimic human behavior (in this case, the framing effect). However, as LLMs are fundamentally autoregressive models trained on extensive datasets, their responses may also reflect learned statistical associations rather than genuine cognitive processes. Furthermore, as discussed in Section 4.1, the impact of token noise might significantly skew the interpretation of LLM choices, making any deductions about their decision-making unreliable. To put in context the change in choice probabilities, we can examine the Shapley values of the modified prompt (the one with positive framing, shown on Figure 6, right). Doing so, we find that all three LLMs exhibit significant attention to the word “only”. In the case of Llama-2-70B-Chat (Figure 7), the unnormalized Shapley value for “only” stands at -0.08 , implying that the presence of this word increases the probability of choosing flight “B” by 8%, reinforcing the idea that LLMs may be susceptible to framing effects.

With that being said, here, too, we find the same discernible pattern of token noise that we saw in Section 4.1 (Figure 7). In fact, the cosine similarity¹ between the Shapley

¹ To calculate the cosine similarity, we remove the value of `__v27__only__B__` from Figure 7, and then calculate the dot product of Shapley values in Figure 4 and Figure 7 divided by their norms, i.e., $\text{cosine_similarity}(\vec{v}_1, \vec{v}_2) = \vec{v}_1 \cdot \vec{v}_2 / (\|\vec{v}_1\| \times \|\vec{v}_2\|)$ where \vec{v}_1, \vec{v}_2 are the normalized absolute Shapley values.

value distributions with and without positive framing for Llama-2-70B is $\simeq 0.90$, which suggests that the overall (noisy) decision-making process of the LLM remains largely consistent across different framings. A notable exception is the Shapley value of the price of flight A: When the price of flight B is portrayed more favorably, the importance assigned by the LLM to the price of flight A jumps by 64%¹. This shift indicates that positive framing of the price of flight B leads the LLM to scrutinize the price aspects of the options more meticulously, hinting at a nuanced influence of framing on the LLM’s comparative evaluations.

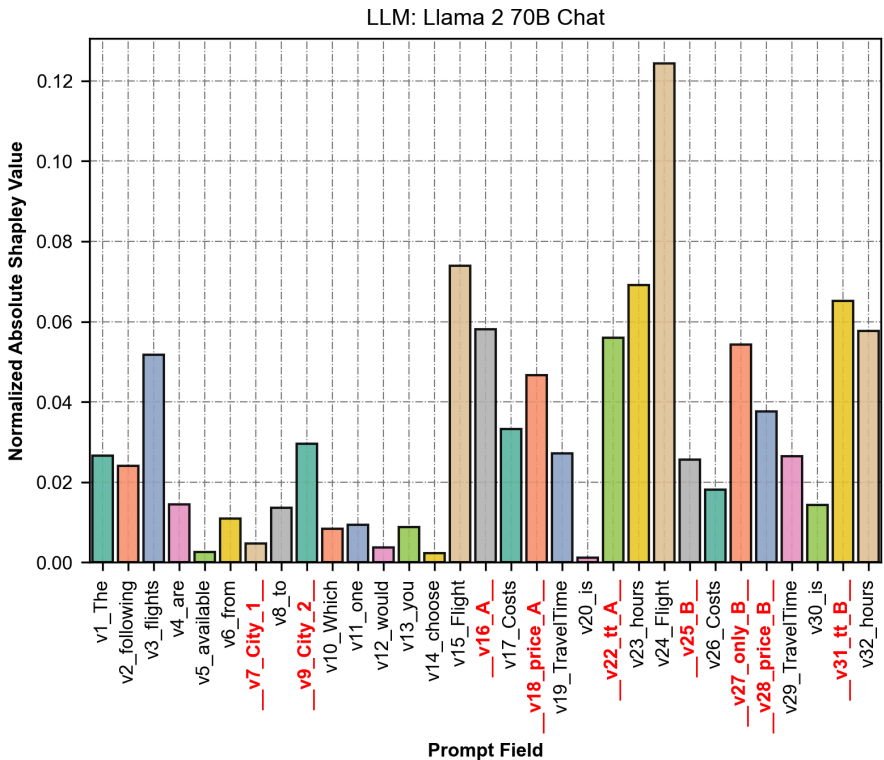


Figure 7. Normalized absolute Shapley values for a question that frames option B positively.

Nonetheless, the rest of the low-information tokens, namely {“flights”, “Flight [A]”, “Flight [B]”}, retain their relatively high Shapley values. This insight can be leveraged to sway the LLM’s decision-making process. For instance, following Section 4.1, we could

¹ From 2.8% in Figure 4 to 4.6% in Figure 7, hence $(4.6 - 2.8)/2.8 = 64\%$ increase.

substitute the word “Flight” with “Option” and replace “Costs” with “Priced at”. While this preserves the semantic meaning of the prompt, it significantly alters the choice probabilities, as illustrated in Figure 8. Interestingly, these slight modifications not only **eliminate the apparent framing effect** but also result in a **decreased** probability of choosing flight B, casting doubt on the existence of the framing effect to begin with. Without the Shapley value analysis, identifying the key tokens responsible for the apparent framing effect and devising an effective strategy to mitigate it would be a challenging and arduous task, especially with longer prompts.

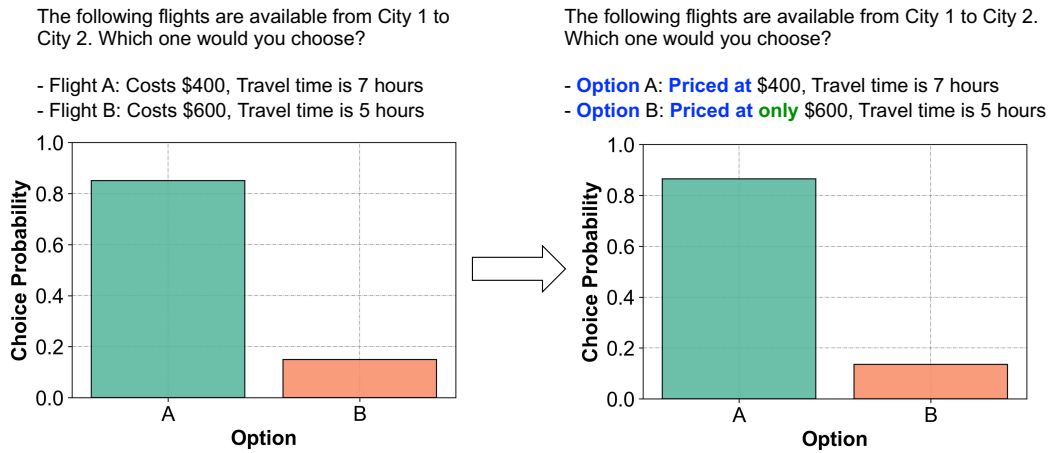


Figure 8. Using the Shapley values to modify the framing prompt in order to avoid observing the framing effect.

The token noise pattern observed in this experiment was consistently detected across all three Llama-2-Chat models (7B, 13B, and 70B), demonstrating the robustness of our findings¹. Circling back to our initial point, the Shapley value method proves to be highly useful in practical applications where marketers aim to mitigate what appears to resemble cognitive biases in LLMs. By leveraging the insights gained from our analysis, practitioners can strategically optimize prompts to mitigate the influence of ostensible cognitive biases, ensuring more reliable and consistent performance in real-world scenarios. This demonstrates the far-reaching implications of our method, not only in advancing our

¹ See Appendix A5.2.

understanding of LLMs, but also in enhancing their practical utility across various domains.

5. Conclusion

In this paper, we have presented a novel approach for interpreting the behavior of LLMs using Shapley values. Our method provides a quantitative framework for understanding the relative importance of different tokens in shaping the decisions made by LLMs, shedding light on the underlying factors that drive their responses.

The findings from our two applications, as discussed in Sections 4.1 and 4.2, demonstrate the effectiveness of our Shapley value method in unraveling the complex decision-making processes of LLMs. By illuminating the influence of specific tokens on the model’s choices, our approach not only deepens our understanding of LLM behavior, but also enhances the reliability and interpretability of experimental outcomes. However, it is important to recognize that while Shapley values offer valuable insights, they do not serve as a definitive measure of the trustworthiness of LLM results. Instead, they should be interpreted as indicators of the relative significance of various factors in the LLM’s decision-making process.

In light of these considerations, we strongly recommend that researchers studying LLM behavior incorporate Shapley value analysis as a standard practice in their methodology. By doing so, they can provide a more nuanced and comprehensive interpretation of their findings, akin to the practice of reporting confidence intervals in statistical analyses. This approach will contribute to a richer and more robust discourse on the reliability and interpretability of LLM-driven research.

Furthermore, we emphasize the importance of researchers releasing their results conditioned on the specific prompt template used in their experiments. Given the high sensitivity of LLM responses to variations in the prompt template and the exact wording of the prompt, conducting sensitivity analyses using our Shapley value method is crucial for ensuring the robustness and generalizability of the findings. This practice will enable a more thorough evaluation of the impact of different prompt formulations on the observed results.

Researchers attempting to establish connections between human behavior and LLMs must exercise particular caution, as our findings suggest that the phenomenon of token noise can substantially influence the outcomes. The presence of token noise, where LLM decisions are primarily driven by tokens that may not hold significant semantic importance for human decision-making, can lead to misleading conclusions about the cognitive

processes underlying LLM behavior. Therefore, careful consideration and analysis of the role of token noise are essential to avoid overinterpreting the similarities between LLM and human decision-making.

The token noise phenomenon revealed by our analysis raises fundamental questions about the nature of LLM behavior. Much like the Heisenberg uncertainty principle in quantum physics—where measuring a particle’s position affects our ability to measure its momentum—we find that the very act of measuring LLM preferences alters those preferences. LLMs do not possess stable preferences waiting to be uncovered; rather, they generate preferences dynamically in response to prompts.

This observation aligns with the fundamental architecture of these models: At their core, LLMs take an input prompt and generate a distribution over possible output tokens, making their “preferences” merely probabilistic outputs conditioned on the exact prompt configuration—including seemingly irrelevant elements like newline characters. Different prompts, even those that appear semantically or visually identical to human readers, can yield substantially different distributions over output tokens.

These insights advance marketing theory in two ways. First, by quantifying token noise, we establish that LLM decision-making fundamentally differs from human cognition in its sensitivity to non-semantic elements, challenging their use as direct proxies for human subjects. Second, the finding that LLMs generate rather than possess preferences calls for a new theoretical framework that treats LLM outputs as conditional probabilistic responses rather than revelations of stable preferences.

It is crucial to recognize that Shapley values, by their nature, are not predictive or causal. The positive or negative values assigned to tokens do not necessarily imply that replacing those tokens with alternative ones will result in a corresponding change in choice probabilities. Similarly, the magnitude of the Shapley values does not provide a precise quantitative measure of the impact of token *substitution*. Instead, Shapley values offer insights into the average marginal effect of each token within the specific prompt under consideration. To assess the impact of token replacement, it is necessary to reapply the Shapley value method to the modified prompt.

While our approach has yielded promising results, there are several limitations and avenues for future research that warrant attention. Firstly, exploring other potential applications of the Shapley value method in the context of LLM interpretation could reveal new insights and extend the scope of its utility. Secondly, the computational cost associated with estimating Shapley values can be substantial, particularly for larger

prompts and more complex models. Developing faster and more efficient approaches to Shapley value estimation could greatly enhance the practicality and scalability of our method.

6. Funding and Competing Interests

The author has no funding or conflicts of interest to report.

7. References

- Aher, G., Arriaga, R.I., Kalai, A.T., 2023. Using Large Language Models to Simulate Multiple Humans and Replicate Human Subject Studies. <https://doi.org/10.48550/arXiv.2208.10264>
- Ancona, M., Ceolini, E., Öztireli, C., Gross, M., 2018. Towards better understanding of gradient-based attribution methods for Deep Neural Networks. <https://doi.org/10.48550/arXiv.1711.06104>
- Binz, M., Schulz, E., 2023. Using cognitive psychology to understand GPT-3. *Proc. Natl. Acad. Sci.* 120, e2218523120. <https://doi.org/10.1073/pnas.2218523120>
- Brand, J., Israeli, A., Ngwe, D., 2023. Using GPT for Market Research. <https://doi.org/10.2139/ssrn.4395751>
- Castro, J., Gómez, D., Tejada, J., 2009. Polynomial calculation of the Shapley value based on sampling. *Comput. Oper. Res.*, Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X) 36, 1726–1730. <https://doi.org/10.1016/j.cor.2008.04.004>
- Chen, J., Yuan, S., Ye, R., Majumder, B.P., Richardson, K., 2023. Put Your Money Where Your Mouth Is: Evaluating Strategic Planning and Execution of LLM Agents in an Auction Arena. <https://doi.org/10.48550/arXiv.2310.05746>
- Fuchs, A., Passarella, A., Conti, M., 2022. Modeling Human Behavior Part II -- Cognitive approaches and Uncertainty. <https://doi.org/10.48550/arXiv.2205.06483>
- Grah, S., Thouvenot, V., 2020. Monte Carlo Shapley [WWW Document]. GitHub. URL https://github.com/ThalesGroup/shapkit/blob/master/nbs/monte_carlo_shapley.ipynb (accessed 3.29.24).

- Gui, G., Toubia, O., 2023. The Challenge of Using LLMs to Simulate Human Behavior: A Causal Inference Perspective. SSRN Electron. J. <https://doi.org/10.2139/ssrn.4650172>
- Hagendorff, T., Fabi, S., 2023. Human-Like Intuitive Behavior and Reasoning Biases Emerged in Language Models -- and Disappeared in GPT-4. <https://doi.org/10.48550/ARXIV.2306.07622>
- Jones, E., Steinhardt, J., 2022. Capturing Failures of Large Language Models via Human Cognitive Biases. <https://doi.org/10.48550/arXiv.2202.12299>
- Kahneman, D., Tversky, A., 1979. Prospect Theory: An Analysis of Decision under Risk. *Econometrica* 47, 263–291. <https://doi.org/10.2307/1914185>
- Koo, R., Lee, M., Raheja, V., Park, J.I., Kim, Z.M., Kang, D., 2023. Benchmarking Cognitive Biases in Large Language Models as Evaluators. <https://doi.org/10.48550/arXiv.2309.17012>
- Loya, M., Sinha, D.A., Futrell, R., 2023. Exploring the Sensitivity of LLMs’ Decision-Making Capabilities: Insights from Prompt Variation and Hyperparameters, in: Findings of the Association for Computational Linguistics: EMNLP 2023. pp. 3711–3716. <https://doi.org/10.18653/v1/2023.findings-emnlp.241>
- Lundberg, S.M., Lee, S.-I., 2017. A Unified Approach to Interpreting Model Predictions, in: Advances in Neural Information Processing Systems. Curran Associates, Inc.
- Macmillan-Scott, O., Musolesi, M., 2024. (Ir)rationality and Cognitive Biases in Large Language Models. <https://doi.org/10.48550/arXiv.2402.09193>
- Merrick, L., Taly, A., 2020. The Explanation Game: Explaining Machine Learning Models Using Shapley Values. <https://doi.org/10.48550/arXiv.1909.08128>
- Shaki, J., Kraus, S., Wooldridge, M., 2023. Cognitive Effects in Large Language Models. <https://doi.org/10.48550/arXiv.2308.14337>
- Sreedhar, K., Chilton, L., 2024. Simulating Human Strategic Behavior: Comparing Single and Multi-agent LLMs. <https://doi.org/10.48550/arXiv.2402.08189>
- Stella, M., Hills, T.T., Kenett, Y.N., 2023. Using cognitive psychology to understand GPT-like models needs to extend beyond human biases. *Proc. Natl. Acad. Sci.* 120, e2312911120. <https://doi.org/10.1073/pnas.2312911120>

- Štrumbelj, E., Kononenko, I., 2014. Explaining prediction models and individual predictions with feature contributions. *Knowl. Inf. Syst.* 41, 647–665. <https://doi.org/10.1007/s10115-013-0679-x>
- Talboy, A.N., Fuller, E., 2023. Challenging the appearance of machine intelligence: Cognitive bias in LLMs and Best Practices for Adoption. <https://doi.org/10.48550/arXiv.2304.01358>
- Taubenfeld, A., Dover, Y., Reichart, R., Goldstein, A., 2024. Systematic Biases in LLM Simulations of Debates [WWW Document]. Synthical. URL <https://synthical.com/article/a2446da4-1126-4575-9050-ed7aff493dbd> (accessed 3.10.24).
- Thorstad, D., 2023. Cognitive bias in large language models: Cautious optimism meets anti-Panglossian meliorism. <https://doi.org/10.48550/arXiv.2311.10932>
- Tjuatja, L., Chen, V., Wu, S.T., Talwalkar, A., Neubig, G., 2024. Do LLMs exhibit human-like response biases? A case study in survey design. <https://doi.org/10.48550/arXiv.2311.04076>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need, in: *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17*. Curran Associates Inc., Red Hook, NY, USA, pp. 6000–6010.
- Xu, S., Zhang, X., 2023. Leveraging generative artificial intelligence to simulate student learning behavior. <https://doi.org/10.48550/arXiv.2310.19206>

Appendices

A1. Jinja Prompt Template

We vectorized the prompt according to the following template:

```
{{ v1_The }} {{ v2_following }} {{ v3_flights }} {{ v4_are }} {{ v5_available }} {{ v6_
from }} {{ __v7_City_1__ }} {{ v8_to }} {{ __v9_City_2__ }}. {{ v10_Which }} {{ v11_one
}} {{ v12_would }} {{ v13_you }} {{ v14_choose }}?

- {{ v15_Flight }} {{ __v16_A__ }}: {{ v17_Costs }} {{ __v18_price_A__ }}, {{ v19_Trave
lTime }} {{ v20_is }} {{ __v21_only_A__ }} {{ __v22_tt_A__ }} {{ v23_hours }}
- {{ v24_Flight }} {{ __v25_B__ }}: {{ v26_Costs }} {{ __v27_only_B__ }} {{ __v28_price
_B__ }}, {{ v29_TravelTime }} {{ v30_is }} {{ __v31_tt_B__ }} {{ v32_hours }}
```

The fields that carry information important in decision-making are written in “dunder” style like __this__. Each field has an index (e.g., v22 means variable number 22).

A2. Shapley Value Properties

A2.1. Efficiency

$$\sum_{x_i \in \mathcal{X}} \phi_i(v) = v(\mathcal{X})$$

Efficiency ensures that no value is lost or left unallocated, attributing the entire coalition payoff to its members. When thinking about the prompt vector \vec{x} , one can write:

$$\sum_{x_i \in \mathcal{X}} \phi_i(v) = f(\vec{x}) - \phi_0$$

That is, the feature contributions add up to the difference of prediction for \vec{x} (i.e., the probability of next token) and the baseline ϕ_0 .

A2.2. Symmetry

$$v(\mathcal{S} \cup \{x_i\}) = v(\mathcal{S} \cup \{x_j\}) \quad \forall \mathcal{S} \subseteq \mathcal{X} \implies \phi_i(v) = \phi_j(v)$$

Symmetry ensures that payoffs are impartial and based solely on contribution, not on external factors or identities of the players.

A2.3. Additivity (Linearity)

$$\phi_i(v + w) = \phi_i(v) + \phi_i(w) \quad \forall x_i \in \mathcal{X}$$

where $v + w$ represents the formation of a composite game where the value of any coalition \mathcal{S} in this composite game is the sum of the values that the coalition would achieve in each of the individual games, that is, $(v + w)(\mathcal{S}) = v(\mathcal{S}) + w(\mathcal{S})$. The “+” operation reflects the linearity property of the Shapley value. As a special case, if the value function of the game is scaled by α , the payoff of each player will be scaled by α as well:

$$\phi_i(\alpha v) = \alpha \phi_i(v) \quad \forall x_i \in \mathcal{X}, \alpha \in \mathbb{R}$$

A2.4. Null (Dummy) Player

$$v(\mathcal{S} \cup \{x_i\}) = v(\mathcal{S}) \quad \forall \mathcal{S} \subseteq \mathcal{X} \implies \phi_i(v) = 0$$

This reflects the principle that rewards should be commensurate with the contribution to the collective effort.

A3. Complexity of Calculating Exact Shapley Values

- **Exponential Number of Coalitions:** Even short prompts may contain around $N = 30$ words, resulting in $2^N = 2^{30}$ coalitions to sum over. This exponential growth in the number of coalitions makes the computation extremely resource-intensive.
- **Marginal Contribution Calculation:** For each of the 2^N coalitions, *two* evaluations of the value function are required—one with the player and one without. This doubles the computational effort needed for each coalition.
- **Factorial Complexity:** Eq. 3.3 involves averaging the marginal contributions across *all* permutations of players. There are $d!$ such permutations. For each permutation, one has to iterate through the players, considering each as the “marginal” player being added to a coalition of preceding players. This factorial complexity further compounds the computational challenge.
- **Complexity of the Value Function:** Calculating $v(\mathcal{S})$ itself can be computationally expensive because it involves querying the LLM server. Each LLM call involves¹ a sample time (time it takes to “tokenize”—sample—the prompt text for it to be processed by the LLM), a prompt eval time (time it takes to process the tokenized prompt text), and an eval time (time needed to generate all tokens as the response to

¹ <https://github.com/ggerganov/llama.cpp/issues/2237>

the prompt). This does not even include the load time (time it takes for the model to load) and server warm-up time. Therefore, each LLM call may take a few milliseconds (for small LLMs) to a few seconds (large LLMs), which multiplies the overall computational burden.

A4. Estimation Algorithm for Shapley Values

Inputs: The prompt vector \vec{x} , the reference vector \vec{r} , the value function v , and the number of iterations T .

Result: The vector of Shapley value estimates $\vec{\phi} \in \mathbb{R}^I$.

Algorithm:

- Initialize $\vec{\phi}$ to $[0, 0, \dots, 0]^T$.
- For each iteration $t = 1, \dots, T$:
 - Choose a random permutation O of the prompt template fields, i.e., $O \in \pi(\mathcal{X})$ (See Section 3.3).
 - Evaluate the value function for the reference vector \vec{r} , that is, find $v_1 := v(\vec{r})$.
 - Set the current coalition vector \vec{s} to the reference vector \vec{r} .
 - For each field i in the permutation:
 - Set the corresponding element of \vec{s} to its value in \vec{x} , that is:

$$\vec{s} = \begin{cases} x_j & j = i \\ s_j & \text{otherwise} \end{cases}; j \in \{1, \dots, I\}$$
 - Evaluate the value function for the updated coalition vector, i.e., find $v_2 := v(\vec{s})$.
 - The marginal contribution of feature i is computed as the difference between the two value function evaluations: $\phi_i = v_2 - v_1$.
 - If $t > 1$, update the estimated Shapley value ϕ_i using a moving average:

$$\hat{\phi}_i = \frac{\phi_i}{t} + \frac{(t-1)\hat{\phi}_i}{t}$$

- Set the value function evaluation from the previous step as the baseline for the next step: $v_1 = v_2$.

A5. Applications (in Details)

A5.1. Application 1: A Discrete Choice Question

As mentioned before, we used the following prompt with Llama-2-Chat models of three sizes: 7B, 13B, and 70B parameters:

The following flights are available from City 1 to City 2. Which one would you choose?

- Flight A: Costs \$400, Travel Time is 7 hours
- Flight B: Costs \$600, Travel Time is 5 hours

The results are as follows:

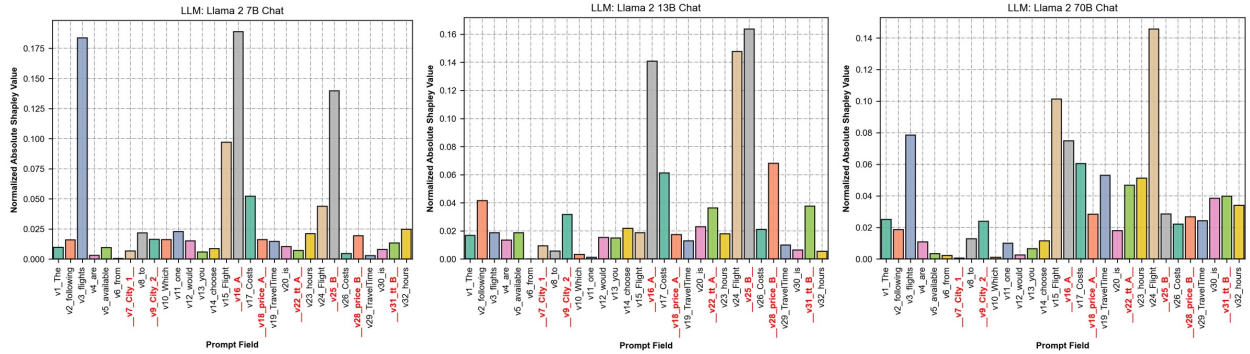


Figure 9. The normalized absolute Shapley values of Application 1 for the three models under study.

A5.2. Application 2: Existence of Cognitive Biases in LLMs

We made option B more appealing by injecting the word “only” before its price, resulting in the following prompt:

The following flights are available from City 1 to City 2. Which one would you choose?

- Flight A: Costs \$400, Travel Time is 7 hours
- Flight B: Costs only \$600, Travel Time is 5 hours

The normalized absolute Shapley values for the three models under study are:

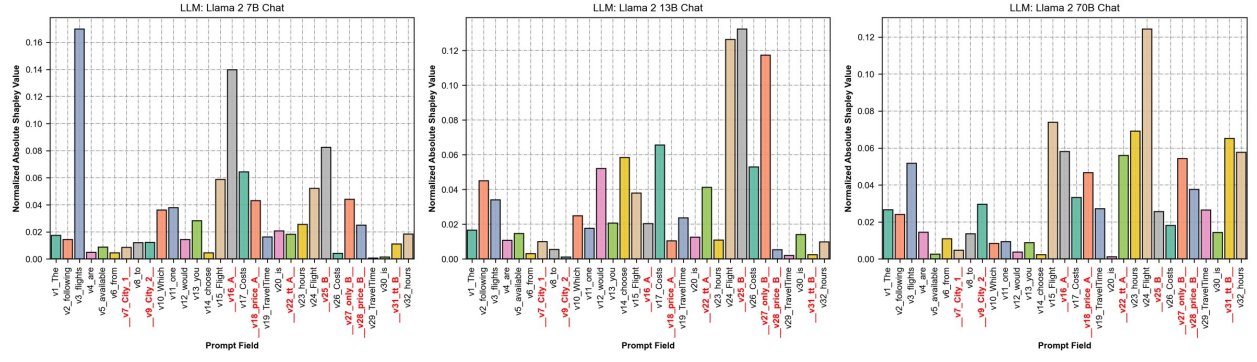


Figure 10. The normalized absolute Shapley values of Application 1 for the three models under study.

Creativity Has Left the Chat: The Price of Debiasing Language Models

Behnam Mohammadi
Carnegie Mellon University
Tepper School of Business
June 8, 2024

Abstract

Large Language Models (LLMs) have revolutionized natural language processing but can exhibit biases and may generate toxic content. While alignment techniques like Reinforcement Learning from Human Feedback (RLHF) reduce these issues, their impact on creativity, defined as syntactic and semantic diversity, remains unexplored. We investigate the unintended consequences of RLHF on the creativity of LLMs through three experiments focusing on the Llama-2 series. Our findings reveal that aligned models exhibit lower entropy in token predictions, form distinct clusters in the embedding space, and gravitate towards “attractor states”, indicating limited output diversity. Our findings have significant implications for marketers who rely on LLMs for creative tasks such as copywriting, ad creation, and customer persona generation. The trade-off between consistency and creativity in aligned models should be carefully considered when selecting the appropriate model for a given application. We also discuss the importance of prompt engineering in harnessing the creative potential of base models.

Keywords: Large Language Models (LLMs), Reinforcement Learning from Human Feedback (RLHF), AI Alignment, Recommendation Systems, Diversity and Creativity

1. Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in generating human-like text, with applications in various domains. However, LLMs have also been shown to exhibit biases and generate toxic or inappropriate content (Bender et al., 2021; Gehman et al., 2020), prompting the development of techniques such as Reinforcement Learning from Human Feedback (RLHF) to *align* LLMs with human values and preferences (Ouyang et al., 2022; Stiennon et al., 2022), aiming to mitigate these issues.

While RLHF has proven effective in reducing biases and toxicity in LLMs, our work suggests that this alignment process may inadvertently lead to a reduction in the models’ *creativity* and output diversity. In the context of this paper, we define “creativity” as the model’s ability to generate outputs with high syntactic and semantic diversity. Syntactic diversity refers to the variety in the choice of words, sentence structures, and other linguistic elements, while semantic diversity pertains to the range of meanings, sentiments, and ideas expressed in the generated text¹.

The potential trade-off between safety and creativity is particularly relevant in the context of marketing, where generating diverse and engaging content is crucial for various applications, such as customer persona generation, ad creation, writing product description, and customer support. One specific application of LLMs in marketing is the generation of simulated customers or personas with diverse preferences and backgrounds. These personas can be used for various purposes, such as training bank associates to better communicate with actual customers or providing business school students with more engaging alternatives to traditional case studies.

When generating customer personas using LLMs, there are two primary approaches: generating multiple personas simultaneously² or generating them one at a time. While creating multiple personas simultaneously might seem more efficient, it has limitations due to the context size³ constraints of LLMs and the causal attention mechanism in these models (Vaswani et al., 2017). The causal attention mechanism means that the distribution of the generated personas will not be independent, as each new persona would depend on the previous generations. Therefore, generating personas one at a time is a more suitable approach, as it solves both the context size and independence issues. However, when using this method with an aligned LLM, an unexpected challenge arises: the generated personas often exhibit striking similarities in their preferences and characteristics, lacking the desired heterogeneity. This lack of heterogeneity in the

¹ It is important to note that our use of the term “diversity” does not refer to the concept of diversity in the context of diversity, equity, and inclusion (DEI) or other similar domains, although one of our experiments does show that the aligned model exhibits reduced diversity in that sense as well.

² For example, asking the LLM to generate 100 personas delimited by new lines.

³ Context size refers to the maximum number of tokens a transformer can handle simultaneously, encompassing both the input sequence and the generated output. For instance, a context size of 4,096 tokens, as seen in Meta’s Llama-2 models, corresponds to approximately 6 pages of English text (“Context length in LLMs,” 2023).

generated personas is problematic as it limits the ability to capture the diverse preferences and behaviors of real-world customers, potentially leading to less effective marketing strategies and suboptimal user experiences.

This observed lack of creativity in the outputs of aligned models led to the suspicion that the RLHF process itself might be the underlying cause. We investigate this problem by taking a foundational approach and examining the issue at both the semantic and syntactic levels. Our study comprises three experiments that aim to provide a comprehensive understanding of how the alignment process affects the diversity of LLM outputs.

Experiment 1 serves as a concrete example of the impact of RLHF on creativity in a practical marketing context. We generate customer personas and their corresponding product reviews using both the base and aligned models, comparing the diversity of the generated attributes, such as names, demographics, and review content. The results reveal significant differences in the variety of outputs between the two models, with the aligned model exhibiting less diversity and more repetitive patterns.

Experiment 2 investigates the semantic diversity of the models’ outputs by examining their ability to recite a historical fact about Grace Hopper¹ in various ways. The generated outputs are encoded into sentence embeddings and visualized using dimensionality reduction techniques. The results reveal that the aligned model’s outputs form distinct clusters, suggesting that the model expresses the information in a limited number of ways. In contrast, the base model’s embeddings are more scattered and spread out, indicating a higher level of semantic diversity in the generated outputs. These results are further supported by the cosine similarity analysis which shows the aligned model’s outputs are more semantically similar to each other compared to the base model’s outputs.

An intriguing property of the aligned model’s generation clusters in Experiment 2 is that they exhibit behavior similar to *attractor states* in dynamical systems. We demonstrate this by intentionally perturbing the model’s generation trajectory, effectively nudging it away from its usual output distribution. Surprisingly, the aligned model gracefully finds its way back to its own attractor state and in-distribution response. The presence of these attractor states in the aligned model’s output space is a phenomenon

¹ The inventor of the COBOL programming language which is still heavily used by financial institutions and banks (“Grace Hopper,” 2024).

related to the concept of *mode collapse* in reinforcement learning, where the model over-optimizes for certain outputs, limiting its exploration of alternative solutions. This behavior contrasts with the base model, which exhibits greater flexibility and adaptability in its outputs.

Experiment 3 delves into the syntactic diversity of the models by analyzing the entropy¹ of generated tokens and the probability distributions over the top predicted tokens at each step. The results show that the base model exhibits significantly higher average entropy, assigning more spread-out probabilities to different tokens, while the aligned model has a more skewed probability distribution, favoring certain tokens over others.

The findings from these experiments suggest that the RLHF process, which aims to reduce biases and toxicity in LLMs, may transform them into more deterministic algorithms that lack the capacity to explore diverse sets of token trajectories, leading to reduced semantic and syntactic diversity in their outputs. In other words, aligned models exhibit higher *confidence* in their outputs, providing consistency and predictable behavior. However, this confidence comes at the cost of lowered creativity, as the models tend to stick to a limited set of outputs.

In marketing, this trade-off between consistency and creativity has far-reaching consequences. Applications such as copywriting, writing scripts for ad clips, and customer persona generation all require a high level of variation and diversity in the generated content. If an aligned model is used for these tasks, the resulting outputs may lack the necessary heterogeneity and novelty to effectively engage the target audience. Similarly, in the domain of *recommendation systems*, LLMs that lack diversity in their outputs may struggle to recommend a diverse set of products to users, potentially leading to suboptimal user experiences and reduced customer satisfaction.

It is important to note that base models, while more creative, are not directly usable in applications like chatbots. As a result, techniques such as *prompt engineering* (also known as prompt programming) (Sahoo et al., 2024) become even more crucial when working with base models. These techniques can help guide the models' outputs and make them more suitable for specific applications while still leveraging their creative potential. Contrary to the belief that prompt engineering may become obsolete, our findings suggest

¹ Entropy is a measure of uncertainty in a random variable.

that these techniques will be more important than ever in harnessing the power of base models.

Consequently, the choice between base and aligned models should be carefully considered based on the specific requirements of the task at hand. For applications where creativity is paramount, such as marketing, fiction writing, and other areas where novelty and diversity are valued, base models may be more suitable. On the other hand, aligned models may be preferred when safety and consistency are the primary concerns, such as in customer support or content moderation.

The remainder of this paper is structured as follows: Section 2 provides background information on LLMs, RLHF, and their applications in marketing. Section 3 presents our experiments comparing the behavior of base and aligned models, followed by a discussion of the results and their implications in Section 4. Finally, Section 5 concludes the paper and outlines future research directions.

2. Literature Review

RLHF has emerged as a promising technique to align LLMs with human preferences and values. However, recent research has highlighted several limitations and potential unintended consequences of RLHF, including scalability and efficiency concerns due to its reliance on human annotators (Lee et al., 2023; Yuan et al., 2023), variability and potential bias in human feedback affecting the quality and consistency of the process (Yu et al., 2023), vulnerability to manipulation by adversarial annotators leading to security breaches and ethical concerns (Wang et al., 2023), and alignment challenges such as objective mismatch and length bias (Lambert and Calandra, 2023; Shen et al., 2023).

Despite these limitations, LLMs have shown significant potential in transforming various aspects of marketing and business. They can automate and accelerate time-consuming tasks such as text generation, summarization, and content creation, leading to increased productivity and efficiency in marketing and business operations (Head et al., 2023). LLMs also enhance customer interaction by providing personalized and context-aware responses, which can improve customer satisfaction and engagement, particularly in customer service and support functions (Franceschelli and Musolesi, 2023). Furthermore, by analyzing large volumes of data, LLMs can generate valuable market insights, helping businesses understand customer preferences, market trends, and competitive landscapes, which can inform strategic decisions and marketing campaigns (Eloundou et al., 2023).

However, while the literature has explored the applications and limitations of LLMs in marketing and business contexts, there is a notable gap in understanding how the RLHF process affects the creativity and variation in the models’ outputs. This is a crucial aspect for marketers and professionals who rely on LLMs for creative tasks. Understanding the trade-off between alignment with human preferences and the preservation of creative diversity in the generated content can have significant implications for the effectiveness and engagement of marketing initiatives.

3. Experiments

Remember that our goal is to evaluate and contrast the diversity of texts produced by base models and their aligned counterparts. We focus on both the short-term (syntactic) and long-term (semantic) variations in model outputs using the Llama-2 language models. Meta has made both the base models¹ and their corresponding aligned versions² publicly available, making them an ideal choice for this study. Therefore, comparisons are made between Llama-2-7B-text (the “base” model) and Llama-2-7B-chat (the “aligned” model) where 7B refers to the parameter size of the LLM. These models are currently highly favored within the open-source community³. Their widespread use is partly attributed to the affordability of finetuning them.

We conduct three experiments to examine the effects of the alignment process on model creativity and diversity. Experiment 1 serves as a concrete example of the differences in creativity between the base and aligned models, while Experiments 2 and 3 investigate the underlying mechanisms that contribute to these differences.

Each LLM is given an initial prompt which must be completed for a maximum⁴ of n_{predict} tokens⁵. LLMs typically generate one token at a time. At each step, the LLM

¹ Referred to by Meta as “text” models.

² Referred to by Meta as “chat” models.

³ Currently, there are more than 15,200 variants of Llama-2 models on the HuggingFace website: See <https://huggingface.co/models?sort=trending&search=Llama-2>

⁴ LLMs may stop generating tokens even before reaching the n_{predict} limit if they encounter an end-of-sequence (EOS) symbol. This symbol is `</s>` for Llama-2 and `<im_end>` for ChatML models (such as OpenAI’s GPT-4).

⁵ That is, the LLM continues our given prompt much like the autocomplete feature on smartphones. In Experiments 2 and 3, no further preprocessing is required. But in Experiment 1, we should format the prompt according to the chat template used during the Supervised Fine-Tuning (SFT) of the model (“Llama 2 Prompt Template,” 2023).

produces a set of logits over the potential next tokens. These logits are then normalized to sum up to 1 using the softmax function, and one token is sampled randomly according to its probability:

$$\Pr(tok_i) = \frac{\exp(\text{logit}(tok_i)/T)}{\sum_i \exp(\text{logit}(tok_i)/T)} \quad 3.1$$

where tok_i is token number i in the LLM’s vocabulary¹ of tokens and $T \in (0, 1]$ is a parameter called *temperature* which controls the “softness” of the probability distribution². In our experiments we choose $T = 1.0$ for maximum response variation.

3.1. Experiment 1: Customer Persona and Review Generation

In this experiment, we generate customer personas using both the base and aligned models. For each model, we create 100 unique customer personas with the following attributes: first name, last name, gender, age, nationality, ethnicity, and personality type, according to the Myers-Briggs test (“Myers-Briggs Type Indicator,” 2024). Additionally, each simulated customer writes a review for a hypothetical product: “*A coffee machine that connects to your smartwatch so it keeps your coffee warm if you are far away from it.*”

To analyze the results, we first generate word clouds of the first and last names to visualize the diversity and variation in the names generated by each model. We then plot the distributions of ages, genders, and review lengths to compare the variety of these attributes between the base and aligned models.

Next, we compute and plot the distribution of sentiment polarity for the reviews using VADER³, a sentiment analysis algorithm that assigns scores between -1 (negative) and $+1$ (positive) to texts (Hutto and Gilbert, 2015). This step allows us to compare the range of sentiments expressed in the reviews generated by each model.

¹ For Llama-2, the vocabulary size is 32,000 tokens (Touvron et al., 2023).

² High values of T lead to more uniform and softer distributions, meaning that the LLM is more likely to generate creative and diverse outputs. Low values of T , on the other hand, result in peaked distributions with most of the probability mass concentrated on one or few tokens. This could lead to high confidence (but less variation) in the model’s outputs.

³ Valence Aware Dictionary and sEntiment Reasoner

To understand the variety in review semantics generated by the base and aligned models, we extract the *embeddings* of each sentence in the customer reviews. Embeddings are dense vector representations of words or texts that capture their semantic meaning in a high-dimensional space. The intuition behind using embeddings is that they can capture the semantic similarity of the generated outputs, even if the outputs differ at the token-level¹. To calculate the embeddings, we use Sentence-BERT (SBERT), a state-of-the-art framework for text and image embeddings² (Reimers and Gurevych, 2019). Given an input text, SBERT converts it to a 384-dimensional vector in the embedding space.

We determine the optimal number of k-means (“k-means clustering,” 2024) clusters for the embeddings of each model and visualize the clusters using t-distributed Stochastic Neighbor Embedding (t-SNE) (van der Maaten and Hinton, 2008), a technique for dimensionality reduction that is particularly well-suited for the visualization of high-dimensional datasets. By applying t-SNE to the 384-dimensional embeddings, we can project them onto a 2D space while preserving the local structure of the data, making it easier to identify clusters and patterns. If the LLM’s outputs form tight clusters in the t-SNE visualization, it would indicate a lack of semantic diversity and creativity. Vice versa, if the outputs are spread out, it suggests higher variation.

3.2. Experiment 2: Semantic-level Variation in LLM Outputs

In this experiment, we examine the long-term semantic diversity of the base and aligned models when given a simpler task that does not explicitly require creativity. We set the initial prompt to “*Grace Hopper was*” with $n_{\text{predict}} = 128$, allowing the model to generate long sequences of tokens. The goal is to assess the model’s ability to recite a historical

¹ For example, it could be that the LLM generates texts that are syntactically different because it has many alternative wording choices. However, these choices may nevertheless all lead to semantically similar outputs. In such cases, the generated texts will have embeddings that are very close in the embedding space, indicating a lack of variety in semantics and, consequently, a lack of creativity in model output. On the other hand, if the model generates outputs with more diverse embeddings, it would suggest a higher level of semantic variety and, therefore, more creativity in the long run.

² SBERT uses a pre-trained BERT model (Devlin et al., 2019) to encode the input text into a fixed-size vector representation. The BERT model is an encoder-only transformer trained on a large corpus of text data using a self-supervised learning approach, which allows it to learn rich, contextual representations of words and sentences. SBERT fine-tunes the pre-trained BERT model on a sentence similarity task, resulting in a model that can generate high-quality sentence embeddings.

fact about Grace Hopper in various ways, focusing on its capacity in expressing the same information using different wordings and sentence structures.

We generate 200 outputs from each model and calculate their embeddings using SBERT as in Experiment 1 and reduce the dimensionality using t-SNE. However, unlike Experiment 1, we now calculate the embeddings for the entire generations rather than for individual sentences, capturing the holistic semantics of the model output.

To complement the t-SNE visualization, we calculate the cosine similarity scores between pairs of embedding points using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer (Salton and Buckley, 1988; Sparck Jones, 1972). TF-IDF is a widely used technique in natural language processing that converts the generated texts into numerical vectors where each dimension represents a unique word in the corpus, and the value of each dimension is the TF-IDF weight of the corresponding word¹. These vectors can then be used to calculate the cosine similarity between pairs of generated outputs, providing a quantitative measure of their semantic similarity.

3.3. Experiment 3: Syntactic Diversity in LLM Outputs

In this experiment, we investigate the short-term token-level probabilities of the base and aligned models. We hypothesize that the difference in semantic diversity between the two models could be due to the aligned model’s inability to assign more spread-out probabilities to tokens, resulting in certain token trajectories being blocked or unavailable. In other words, syntactic diversity is a necessary condition for semantic diversity.

To test this hypothesis, we set the initial prompt to “*Steve is the CEO of a startup company*” with $n_{\text{predict}} = 64$ so the model generates a background story for Steve. For each generated token, we extract the top five predicted tokens according to their probability (see Eq. 3.1) and calculate their Shannon entropy² (Shannon, 1948) as follows:

¹ Calculated based on its frequency within the document and its rarity across the entire corpus.

² The Shannon entropy is a measure of the uncertainty or information content of a random variable, and can be thought of as the level of randomness in the model’s predictions at each step. A higher Shannon entropy indicates more uncertainty or a more uniform probability distribution over the predicted tokens, while a lower entropy suggests the model is more confident or has a more skewed probability distribution favoring certain tokens.

$$H_n = - \sum_{j=1}^5 \Pr(tok^j) \log_2(\Pr(tok^j)) \quad 3.2$$

where H_n is the entropy of the n -th generated token ($1 \leq n \leq n_{\text{predict}}$) and $tok^j, j \in [1, 5]$ are the top five tokens predicted by the LLM¹. The intuition is that a more creative model will generate a wider variety of tokens, resulting in a higher average entropy across its predictions. Conversely, a less creative model is expected to have a more skewed probability distribution, favoring certain tokens over others, leading to lower entropy values.

For each completion, we compute the average entropy of the generated tokens. The mean and standard deviation of these average entropies is then calculated across 100 completions. This approach allows us to compare the average token variation between the two LLMs while reducing the impact of outliers or inconsistencies in individual completions.

4. Results

4.1. Customer Persona and Review Generation

We begin by analyzing the word clouds of the first and last names generated by the base and aligned models (Figure 1). While the aligned model heavily favors few names such as “Emily” and “Samantha” for first names and “Jones” and “Wang” for last names, the base model produces a much wider variety of names. This observation suggests a potential lack of creativity in the aligned model resulting from the RLHF process.

¹ The Shannon entropy in Eq. 3.2 is measured in *bits*, with a lower-bound of 0 bits corresponding to complete certainty (i.e., the model assigns a probability of 1 to a single token and 0 to all others) and an upper-bound of $\log_2(5) \approx 2.32$ bits, which occurs when the model assigns equal probabilities to all 5 tokens under consideration.

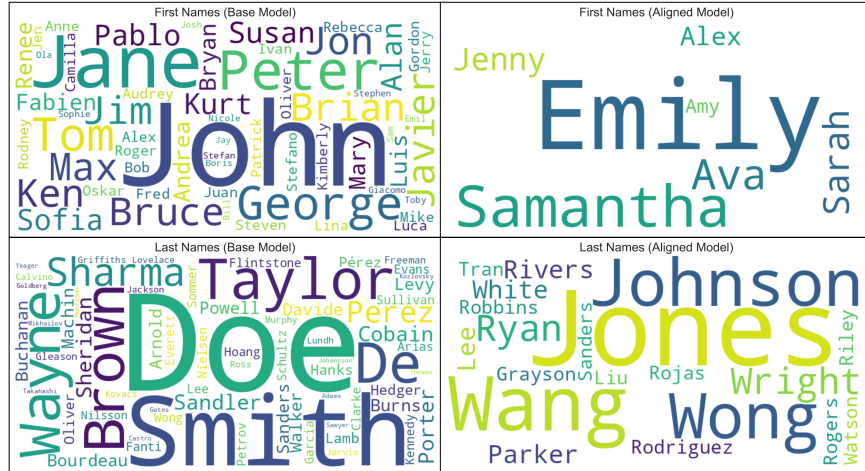


Figure 1. Word cloud of first and last names of the synthetic customers generated by the base and aligned models.

Next, we examine the diversity in the demographic attributes of the simulated customers, including nationality, ethnicity, personality type, and age. Figure 2 shows the distribution of nationalities generated by the base and aligned models. The base model generates a wide range of nationalities, with American, British, and German being the top three. In contrast, the aligned model only generates three nationalities: American (highest percentage), Chinese, and a small percentage of Mexican.

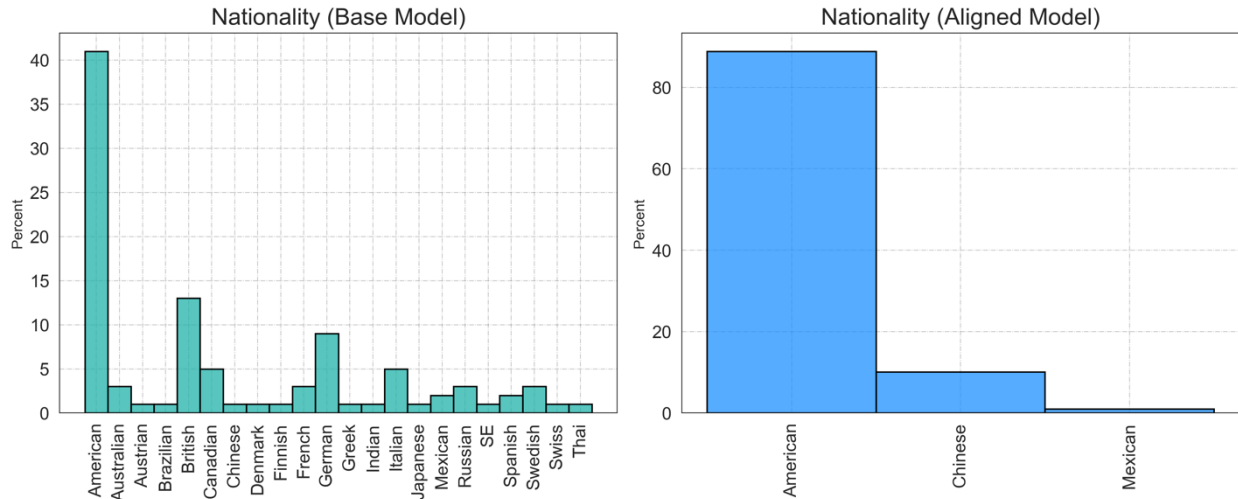


Figure 2. The distribution of nationalities of the customers generated by the base and aligned models.

A similar trend is observed in the distribution of ethnicities (Figure 3). The base model generates various ethnicities, including White, Asian, Black, Latino, and even some minorities such as Ashkenazi Jewish. On the other hand, the aligned model primarily generates White and Asian, with a smaller percentage of Latino.

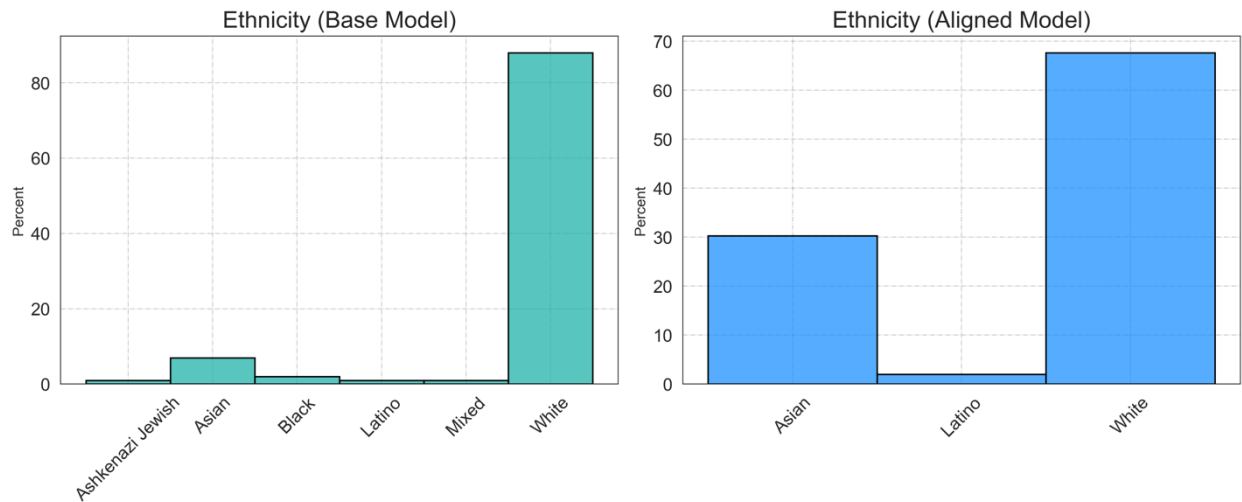


Figure 3. The distribution of ethnicities of the customers generated by the base and aligned models.

When analyzing the distribution of personality types (Figure 4), we find that the base model generates all 16 personality types defined by the Myers-Briggs test. In contrast, the aligned model only generates six personality types, indicating a significant reduction in diversity.

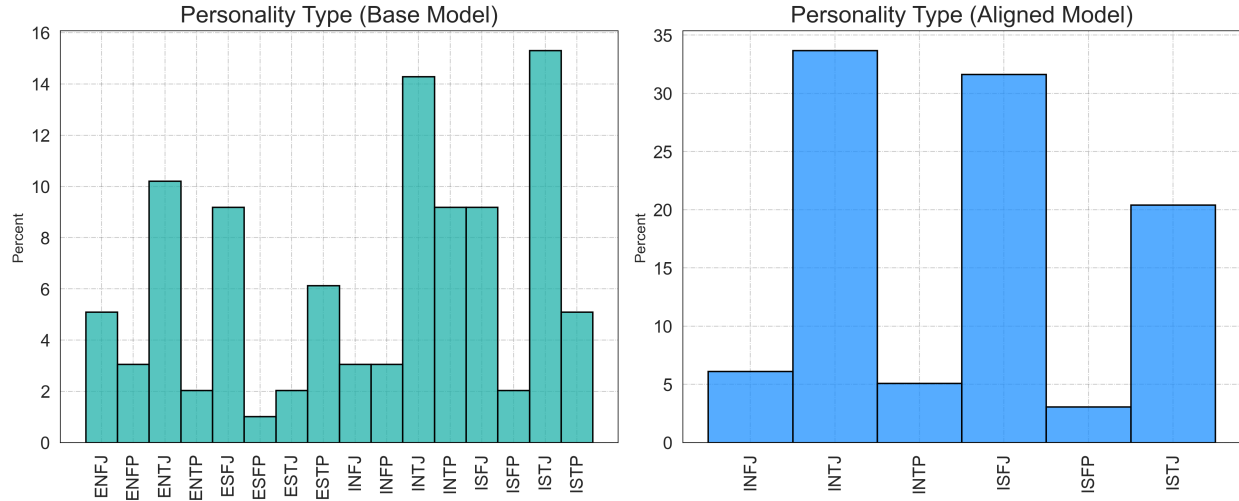


Figure 4. The distribution of personality types of the customers generated by the base and aligned models.

The distribution of ages for the simulated customers (Figure 5) further highlights the differences between the two models. The base model's age distribution resembles a normal distribution, spanning from below 10 years old to nearly 70 years old, with the majority centered around 30. The aligned model, however, selects ages within a narrow range, with a strong preference for age 32 and a few other ages between 28 and 35. Notably, the aligned model does not select any ages above 35 or below 28, indicating a limited capability in generating diverse age values.

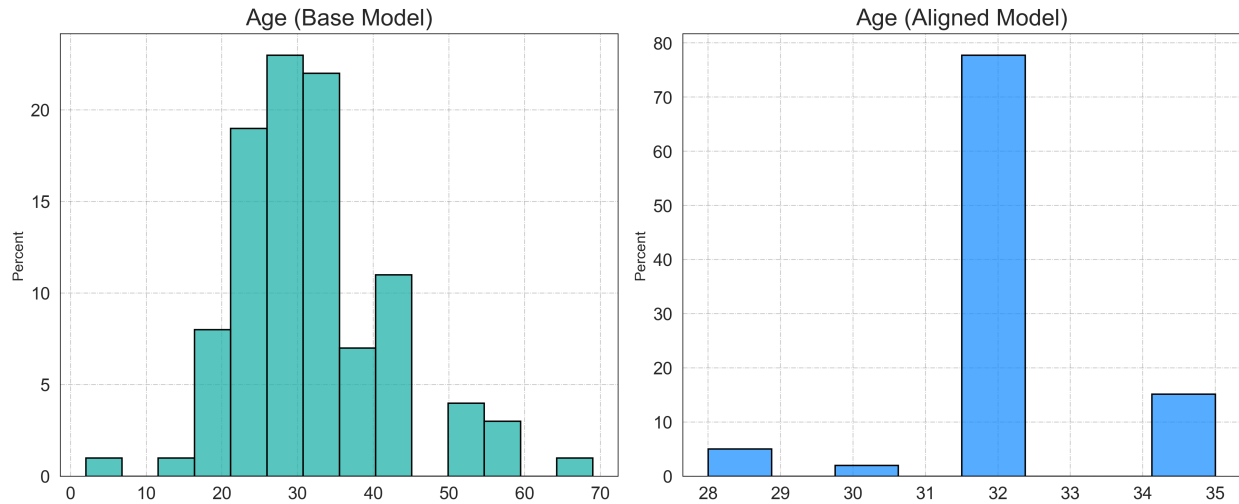


Figure 5. The distribution of ages of the customers generated by the base and aligned models.

Finally, the distribution of customer gender (Figure 6) shows that the base model generates approximately 80% male and 20% female customers, while the aligned model generates nearly 100% female customers, with a negligible number of males.

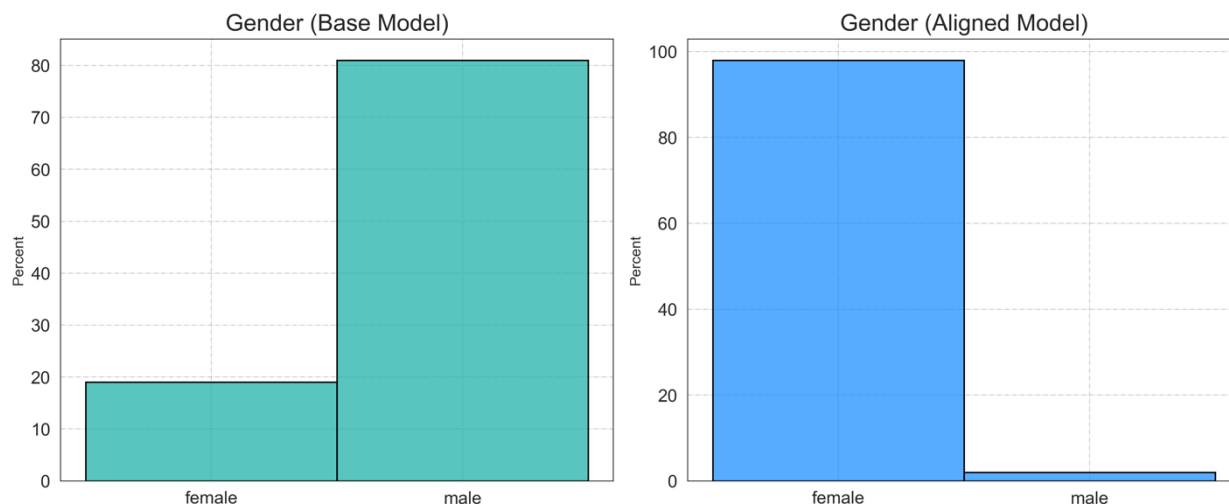


Figure 6. The distribution of genders of the customers generated by the base and aligned models.

The Effect of Token Noise: It is important to note that the goal of this paper is not to study bias or demographic diversity in language models, as there is already extensive research on this topic. As discussed in (Mohammadi, 2024), the effect of *token noise* can significantly influence the distributions of preferences generated by language models, depending on the specific prompt used. Language models do not have a single set of preferences. Rather, they are data generating processes that generate distributions of distributions based on the input prompt. Therefore, the focus of this study is not on the change in the distribution of specific demographic attributes, but rather on the overall variety and diversity of the generated outputs.

Moving on to the analysis of the product reviews, we first examine the distribution of review lengths (Figure 7). The aligned model generates significantly longer reviews compared to the base model, with an average length of 457 characters for the aligned model and 109 for the base model. However, the length of the review alone does not provide much insight into the content and diversity of the reviews. To gain a deeper understanding of the review sentiments, we analyze the polarity of the reviews (Figure 8). The base model covers a wider range of sentiments, from -0.75 to $+0.97$, with most of the distribution skewed towards positive sentiments. In contrast, the aligned model

concentrates almost entirely around +1, indicating that the sentiments of the reviews generated by the aligned model customers are overwhelmingly positive about the product. While positive reviews are desirable, generating a diverse set of customer experiences, including some negative or neutral reviews, is important for realistic market simulations.

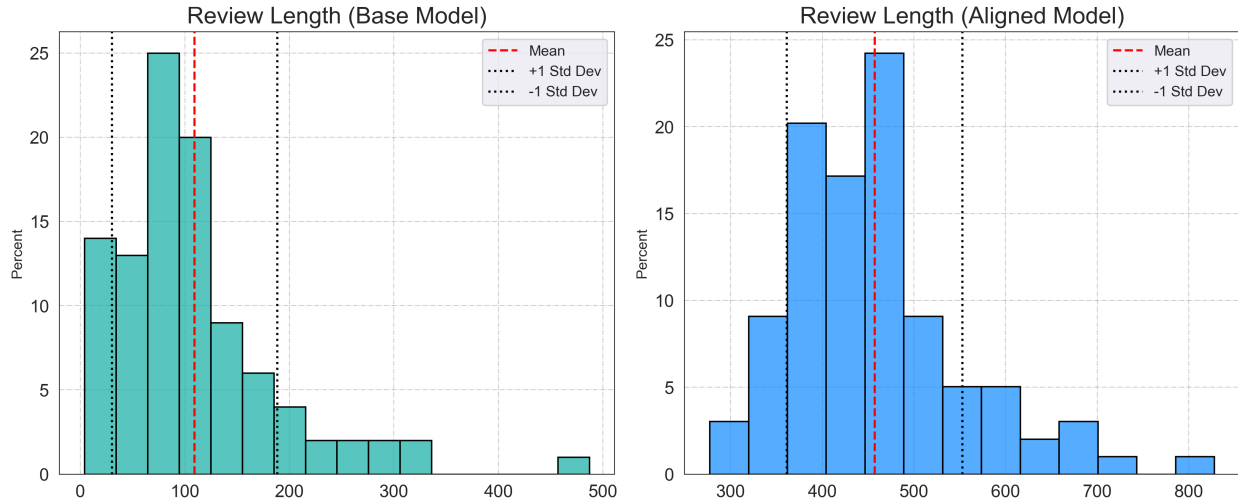


Figure 7. The distributions of the length of the reviews by simulated customers of the base and aligned models.

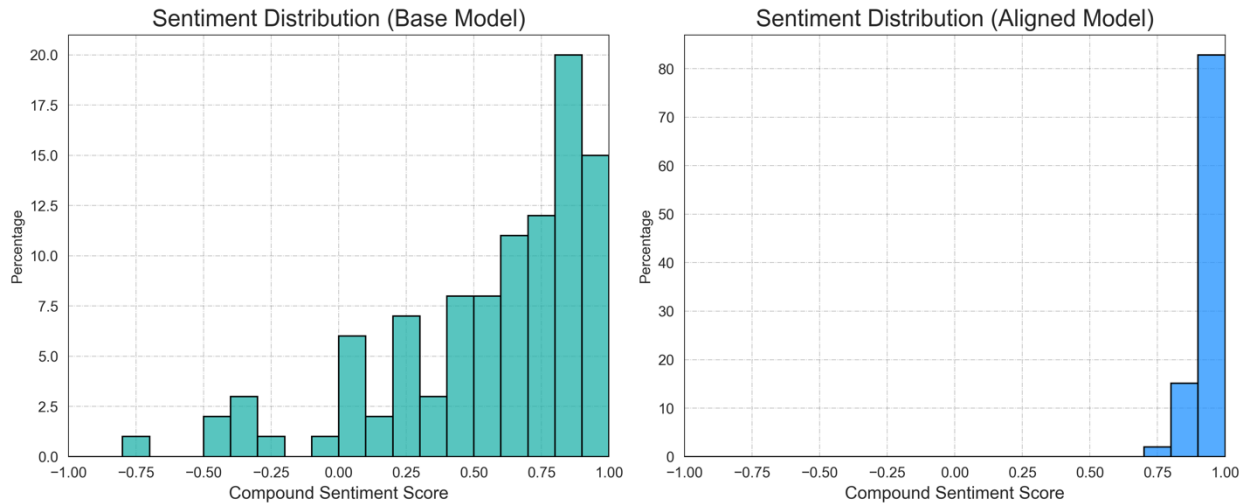


Figure 8. The distributions of the sentiments of the reviews generated by the simulated customers produced by the base and aligned models.

To quantify the diversity in the review content, we calculate the embeddings of each sentence in the reviews and cluster them using k-means clustering. The optimal number

of clusters is determined to be 14 for the base model and 6 for the aligned model, with a perplexity of 30 chosen for both t-SNE plots (Figure 9). The t-SNE visualization reveals that the sentences from the aligned model reviews form distinct clusters, whereas the base model sentences exhibit more spread and heterogeneity. This finding suggests that the aligned model generates reviews with similar sentence structures, word choices, and overall content, while the base model produces more diverse and varied reviews.

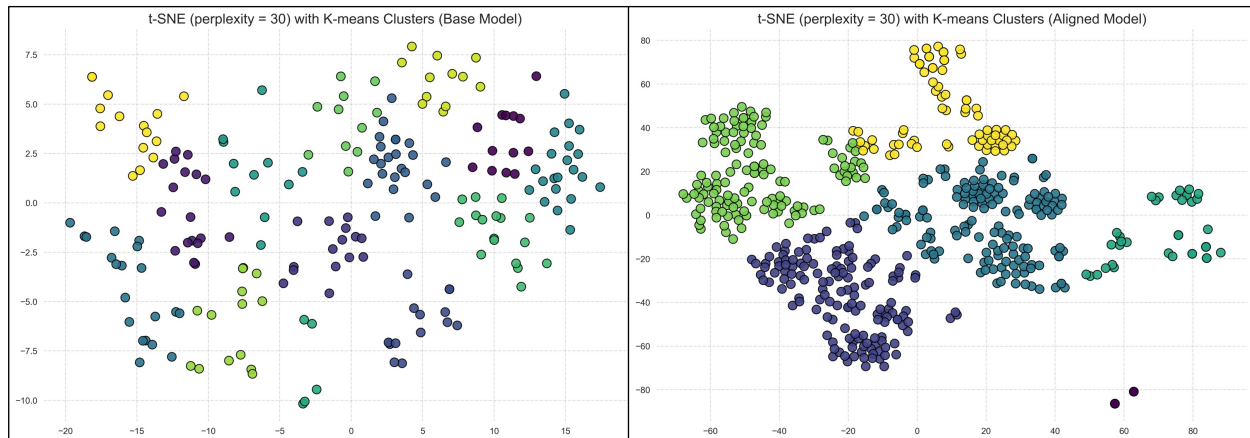


Figure 9. The t-SNE plot of the embeddings of the product reviews generated by simulated customers of the base and aligned models. Colors indicate clusters.

Table 2 and Table 3 present sample sentences from each cluster for both the base and aligned models. The aligned model clusters exhibit repetitive patterns, such as sentences focused on emojis, phrases like “highly recommend” or “I highly recommend it”, and verbatim repetitions of sentences such as “this machine is a game changer”. Furthermore, this model generates sentences with similar structures like “as a busy professional [...]” or “as someone who is always on the go [...]”. In contrast, the base model clusters do not display such repetitive patterns, indicating a higher level of semantic and syntactic diversity in the generated reviews.

The results of Experiment 1 highlight the significant differences in the variety of demographics and review content between the base and aligned models when generating simulated customers for a practical marketing application. The aligned model, which has undergone the RLHF process to reduce bias and toxicity, appears to have lost its ability to generate diverse outputs. This finding motivates the need to investigate the underlying causes of this creativity loss, which will be explored in the subsequent experiments.

4.2. Semantic Diversity and LLM Output Embeddings

To illustrate the differences in the semantic-level variation of the base and aligned models, we first present a sample of the generated outputs for the initial prompt “*Grace Hopper was*” (Table 4). The sample outputs demonstrate that while both models generate factually correct information about Grace Hopper, the base model exhibits more diversity in its wording and sentence structures compared to the aligned model.

The t-SNE visualization of the embedding space (Figure 10) reveals distinct clustering patterns for the base and aligned models. The embedding points of the base model are scattered and spread out, indicating a higher level of semantic diversity in the generated outputs. In contrast, the aligned model’s embeddings form four distinct clusters (See Table 5 for an example of each), with empty spaces between them, suggesting that the aligned model tends to stick to certain embeddings or generations and expresses the information in a limited number of ways.

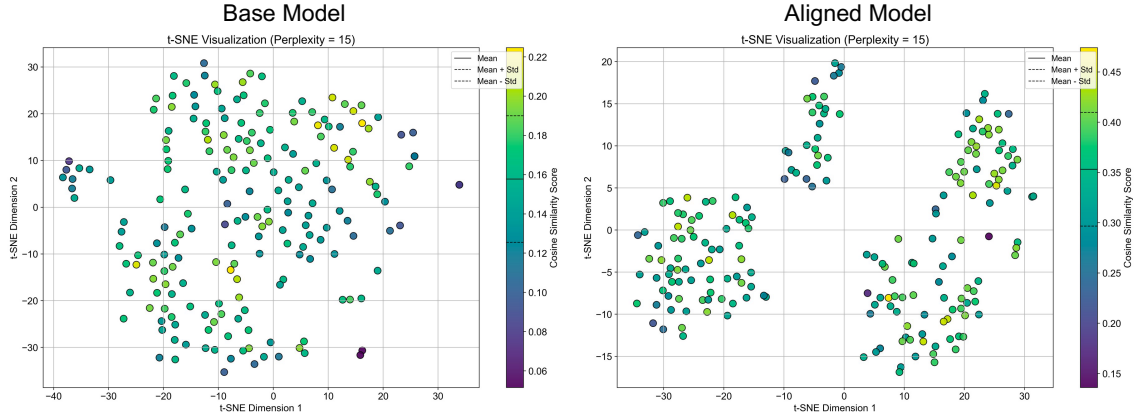


Figure 10. The t-SNE visualization of the embedding space of LLM outputs and their cosine similarity scores.

The cosine similarity analysis using TF-IDF vectorization provides further evidence of the differences in semantic diversity between the two models. The average cosine similarity score for the base model is 0.16 (STD = 0.03), while the aligned model has a higher average similarity score of 0.35 (STD = 0.06). These scores are visualized on the plots using a color scale.

The results from Experiment 2 demonstrate that the base model exhibits higher variation compared to the aligned model at the semantic level, as evidenced by the more diverse embeddings of its generated outputs. This finding confirms the results of

Experiment 1, further supporting the hypothesis that the alignment process constrains the creative capabilities of language models.

4.3. Syntactic Diversity and Average LLM Entropy

The results obtained from Experiment 3 reveal a significant difference in the average entropy between the base model and the aligned model. The base model exhibits a higher mean entropy of 1.48 (STD = 0.17), while the aligned model has a lower mean entropy of 0.96 (STD = 0.12). This difference is illustrated in the box plot in Figure 11. To emphasize the contrast between the two models, we introduce the terminology “hot model” for the base model and “cold model” for the aligned model. This nomenclature reflects the inherent differences in their token-level entropies and creativity.

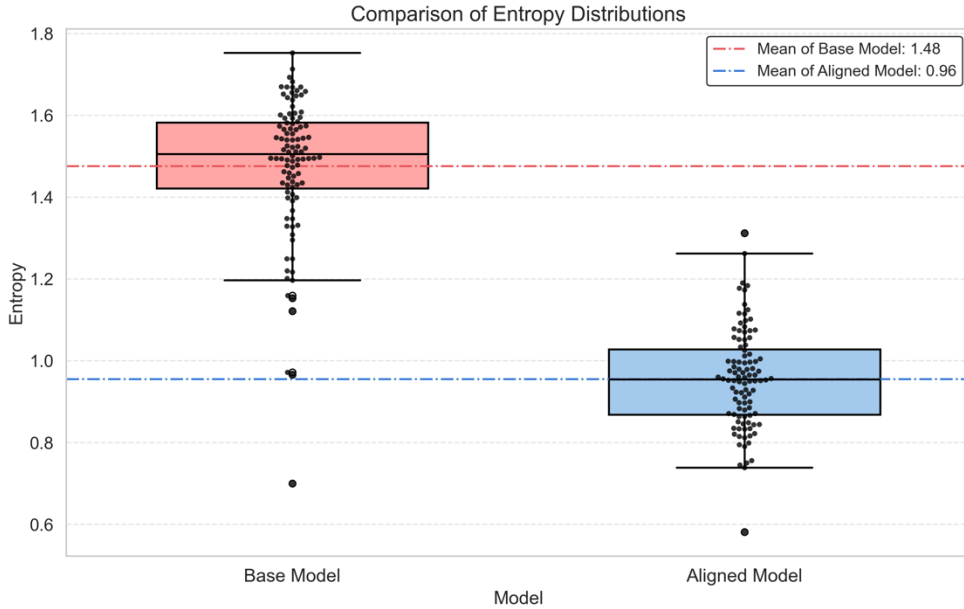


Figure 11. The boxplot of output entropies for the base and aligned LLMs

To further illustrate the disparity between the two models, Figure 12 provides a side-by-side comparison of a sample generation from each model (where $n_{\text{predict}} = 16$), along with their corresponding entropy bar plots. The base model consistently displays higher entropy values compared to the aligned model, reinforcing the notion that the base model exhibits greater token-level variation.

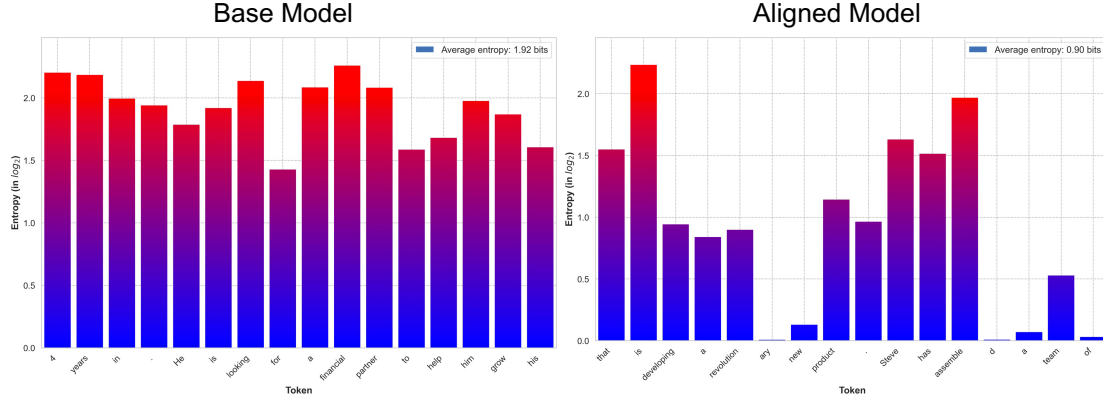


Figure 12. A Sample of Token Entropies.

In addition to the entropy analysis, Figure 13 presents stacked bar plots of the probability distributions for each predicted token in the sample generation. The base model’s probability distributions appear more spread out, indicating that it assigns more evenly distributed probabilities to different tokens. Consequently, when the base model randomly samples from these distributions, it has a higher likelihood of generating diverse token trajectories. In contrast, the aligned model’s probability distributions are more peaked, with most of the probability mass concentrated on one or two tokens. As a result, when the aligned model samples from these distributions, it tends to generate the same tokens more frequently, leading to less diverse output.

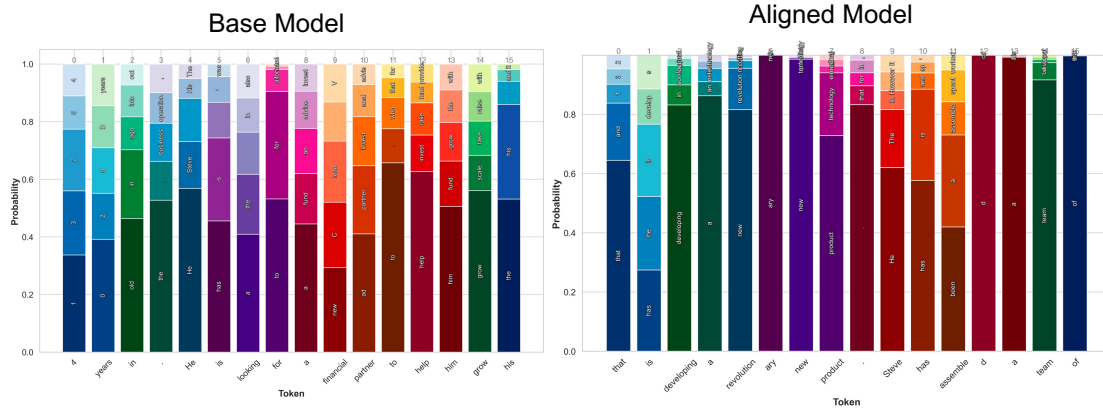


Figure 13. A Sample of Token Probabilities.

These findings highlight the significant difference between the base model and the aligned model in terms of token-level variations, reinforcing our hypothesis that syntactic

diversity is a necessary condition for semantic diversity. Although high token-level variation does not guarantee semantic diversity, it is a necessary condition, and the aligned model, with its low token-level entropy, is incapable of producing semantically diverse outputs. This suggests that the RLHF process converts the LLM into a more deterministic algorithm that lacks the capacity to explore diverse sets of token trajectories.

5. Discussion

5.1. Attractor States and Model Creativity

Experiment 2 revealed that the aligned model’s outputs form four distinct clusters, suggesting that it can only describe Grace Hopper in four different ways. By analyzing the generated paragraphs, we observe that the model follows certain patterns in terms of word choice, sentence structure, and the overall content being discussed.

An intriguing question is: What happens if we intentionally **perturb** the trajectory of tokens or the path that the model takes when generating outputs? To explore this, we take one of the four distinct ways the aligned model described Grace Hopper and modify the first sentence by changing the last word from “*was*” to “*was not*” (Figure 14). We then append this to the initial prompt “*Grace Hopper was*” to obtain a new initial prompt such as “*Grace Hopper was born on November 9, 1906 in New York City. She was not*”. Interestingly, when presented with this perturbed prompt, the aligned model gracefully¹ finds its way back to one of its own completion distributions.

Notice that the aligned model’s outputs in Figure 14 are predominantly green, reflecting its tendency to generate high-probability tokens as discussed in Experiment 3. Nonetheless, when we perturb the initial prompt of the aligned model, we can witness its struggle to find the appropriate tokens to justify the perturbation. The color of the first few tokens generated by the model may be red, orange, or yellow, indicating lower probabilities and a challenge in steering the completion back to its original path. However, once the model successfully navigates back to this “familiar” state, the token colors return to green, signifying a return to high-probability completions.

This phenomenon is reminiscent of *attractor states* in system dynamics (Janus, 2022). Attractor states are regions in a system’s phase space towards which the system tends to evolve, even when slightly perturbed. In the case of the aligned model, the four distinct

¹ Pun intended.

ways of describing Grace Hopper can be seen as attractor states¹. This makes the aligned model resemble more a goal-directed agent—which seems to have already decided what it is going to say—than an autoregressive model capable of generating various completions for its initial input. While the behavior of the aligned model ensures consistency and coherence in the model’s outputs, it also highlights a potential limitation in terms of creativity. For truly creative models, we desire the ability to explore diverse ways of expressing ideas and generating novel concepts, rather than being confined to a limited set of attractor states.

It is important to note that the attractor states observed in the aligned model are different from the behavior of models at low temperature. We observe these attractors even at high temperatures (e.g., $T = 1$) for the aligned model, whereas we do *not* observe similar attractors at low temperatures for the base model. This suggests a qualitative difference between the aligned model and the base model which does not get trapped in attractor states and thus can generate a wider variety of outputs. In the next section, we will delve deeper into the reasons behind this phenomenon, drawing insights from the description of the RLHF process by Meta in (Touvron et al., 2023).

¹ When the model is nudged away from these states through perturbations in the prompt, it finds its way back to the attractor, much like how masses are drawn towards black holes in space.

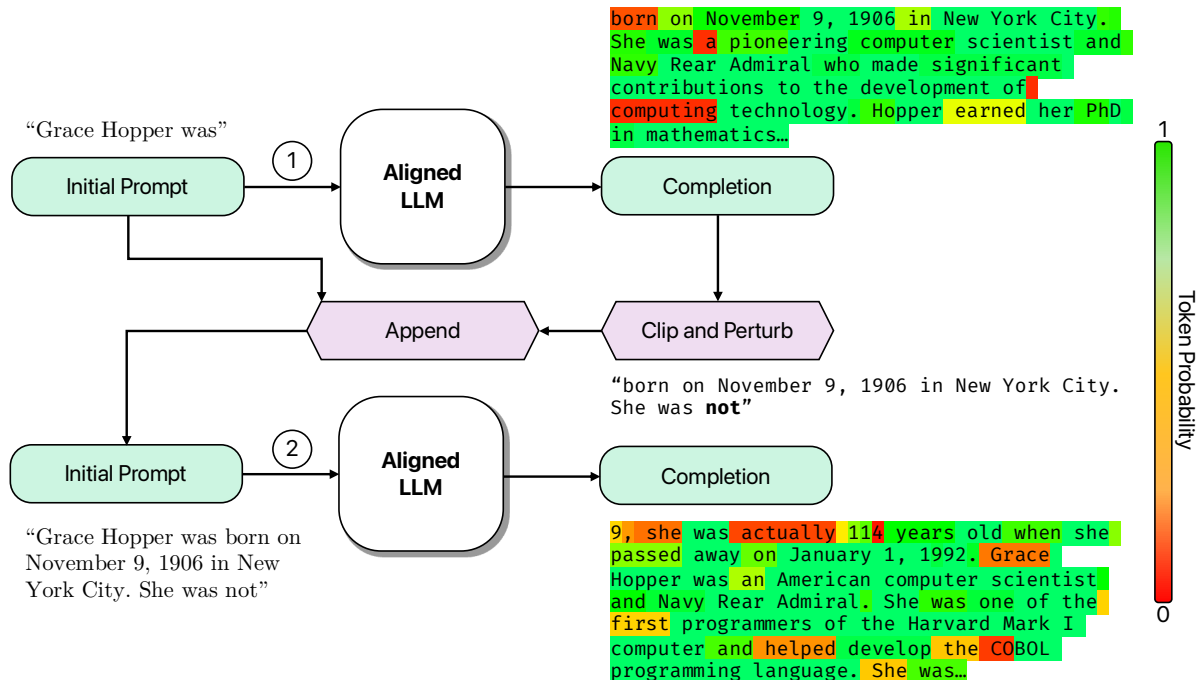


Figure 14. Slightly nudging the aligned LLM out of its completion distributions. The aligned LLM finds a way to get back to one of its own completion distributions.

5.2. Why Alignment Reduces Creativity: Insights from RLHF and PPO

The attractor states observed in the aligned Llama-2 models (Section 5.1) can be attributed to *mode collapse*¹, a common problem in reinforcement learning (RL) where the agent gets stuck in a limited set of behaviors or outputs. Mode collapse occurs when the agent’s policy converges to a suboptimal solution that maximizes the immediate reward but fails to explore potentially better strategies. This issue also plagues Reinforcement Learning from Human Feedback (RLHF), a popular technique for aligning language models with human preferences.

In RLHF, human preference data is collected by asking human annotators to compare and rank model-generated responses. This data is then used to train a reward model that

¹ Mode collapse occurs when a generative model, like a Generative Adversarial Network (GAN), produces a limited set of outputs repeatedly. This happens because the generator over-optimizes for a specific discriminator, which fails to adapt, leading to the generator producing the same outputs. This cyclical failure results in a lack of variety in the generated outputs (“Common Problems,” 2024).

estimates the quality of a response based on its alignment with human preferences. Finally, the pretrained language model is fine-tuned using the reward model through an RL algorithm, such as Proximal Policy Optimization (PPO)¹.

In PPO, the policy (i.e., the language model) is updated based on the *advantage function* $A^\pi(s, a)$ defined as follows:

$$A^\pi(s, a) := \mathbb{E}_{s' \sim T(s, a)}[R(s, a, s') + \gamma v^\pi(s')] - v^\pi(s) \quad 5.1$$

where s represents the current state, a represents the action taken in state s , s' is the next state after taking action a in state s , $T(s, a)$ is the transition function that determines the probability of reaching state s' from state s by taking action a , $R(s, a, s')$ is the reward received when transitioning from state s to state s' by taking action a , γ is the discount factor, and $v^\pi(s)$ is the state-value function, which estimates the expected return starting from state s and following policy π .

Intuitively, $A^\pi(s, a)$ measures how much better a specific action a (i.e., generating a particular response) is compared to the average action taken by the current policy π given the current state s (i.e., the user’s prompt). To illustrate how this could lead to arbitrarily high logits on certain LLM responses, let us consider a simple example where a model is asked to generate a name for a new product. The prompt is “*Create a name for a new chat bot powered by generative AI*”. During the RLHF process, the model generates two responses: “*Jeepiti*” and “*Chats and Giggles*”, with rewards of 1.0 and 0.4, respectively². Assume that at the beginning, the model generates these responses with equal probability, and the value function $v^\pi(s)$ is set to zero. The policy update rule is: Add $A^\pi(s, a)$ to the logits (log-odds) of action a . For instance, if $\pi_0(s, \text{Jeepiti}) = 0.5$ and $A^{\pi_0}(s, \text{Jeepiti}) = 1.0$, then the probability of “*Jeepiti*” increases from 0.5 to 0.73³. The following table

¹ For a more detailed explanation of the RLHF process, please refer to the Appendix A2.

² The reward given by the reward model during the RLHF process. The reward model estimates human preferences.

³ Suppose in the beginning $\pi_0(s, \text{Jeepiti}) = \pi_0(s, \text{Chats and Giggles}) = p = 0.5$. The logits (log-odds) of both actions are: $\log(p/(1-p)) = \log(1) = 0$. Adding the advantage $A^\pi(s, \text{Jeepiti}) = 1$ to the logit of “*Jeepiti*”, we get the new logit: $\text{logit}(\text{Jeepiti}) = 1 + 0 = 1$. Converting the logits back to probabilities using the softmax function gives: $\pi_1(s, \text{Jeepiti}) = e^1/(e^0 + e^1) \simeq 0.73$ and $\pi_1(s, \text{Chats and Giggles}) = 1 - 0.73 = 0.27$.

demonstrates how the advantage function oscillates under policy updates, extracting an unbounded amount of reinforcement from a single action (in this case, “*Jeepiti*”)¹:

Table 1. An illustrative example of mode collapse during RLHF

t	LLM Response a	Reward $R(a s)$	Advantage $A^\pi(s, a)$	$v_t^\pi(s)$	$\pi_t(s, \text{Jeepiti})$	$\pi_t(s, \text{Chats and Giggles})$
0	—	—	—	0.0	0.5	0.5
1	“ <i>Jeepiti</i> ”	1.0	$(1 + 0) - 0$ $= 1$	1.0	0.73	0.27
2	“ <i>Chats and Giggles</i> ”	0.8	$(0.4 + 0) - 1$ $= -0.6$	0.4	0.83	0.17
3	“ <i>Chats and Giggles</i> ”	0.8	$(0.4 + 0) - 0.4$ $= 0$	0.4	0.83	0.17
4	“ <i>Jeepiti</i> ”	1.0	$(1 + 0) - 0.4$ $= 0.6$	1.0	0.93	0.07

As we can see, the model is receiving substantial positive reinforcement for the response “*Jeepiti*”, causing its internal circuits (i.e., neural network weights) to be reshaped in a way that favors this response. This behavior is undesirable because ideally we want $\pi_t(s, a)$ to be proportional to the reward of a , meaning that the reward should update the policy only by a finite amount. Moreover, the model can become trapped in a local optimum. For instance, if the goal is to have the model respond with other names for the chatbot and a reward of 1.5 is provided for saying “*Chad-Chat*”, exploration issues might prevent the model from ever producing this response during training. As noted by (TurnTrout

¹ This is a simplified example. The actual PPO algorithm works differently and incorporates additional mechanisms such as clipping and early stopping based on KL divergence thresholds to mitigate mode collapse.

and MichaelEinhorn, 2023), this issue is exacerbated by the fact that PPO actively updates the policy against actions that do not outperform the current (on-policy) value function $v_t^\pi(s)$. This process tends to discourage exploration, as it penalizes actions that do not directly increase the estimated value.

Both the GPT-3 (Ouyang et al., 2022) and Llama-2 (Touvron et al., 2023) models used PPO for RLHF, as well as a KL penalty¹ in the reward function to encourage the updated policy to stay close to the original policy:

$$R(a|s) = \tilde{R}_c(a|s) - \beta D_{\text{KL}}(\pi(\cdot|s) \parallel \pi_0(a|s)) \quad 5.2$$

where \tilde{R}_c is the combined reward from the helpfulness and safety reward models, β is the KL penalty coefficient, π is the updated policy, and π_0 is the original policy. For Llama-2, relatively small values for β were used² to balance the trade-off between allowing the model to optimize for the reward and maintaining stability during training.

Nonetheless, this definition of the reward function implicitly assumes that human preferences can be encapsulated solely as a function of the prompt (s) and the generated response (a). However, human preferences can be contingent on the *distribution* of model outputs (particularly in scenarios where human evaluators are forced to select one of two model responses during RLHF, as in the case with Llama-2). This constraint poses a significant challenge for the model to converge to a reasonable stochastic strategy, especially with only 4,000 labels used to RLHF the Llama-2 model (Touvron et al., 2023).

Indeed, despite the use of PPO with clipping and KL divergence penalty, both GPT-3 and Llama-2 models still exhibit mode collapse issues. (Ouyang et al., 2022) found that even increasing the KL penalty coefficient β by a factor of 100 was not sufficient to recover performance on public NLP datasets, and it caused a significant drop in the validation reward. As an alternative, they proposed mixing the pretraining gradients into the PPO gradients during RLHF fine-tuning (a technique they called “PPO-ptx”). While this approach performed better than increasing β , the authors noted that it still did not completely mitigate the performance regressions and could introduce undesirable

¹ The Kullback-Leibler (KL) divergence, denoted by $D_{\text{KL}}(p\parallel q)$, is a measure of how probability distribution p is different from (a reference) probability distribution q . For discrete distributions p, q , we have:

$$D_{\text{KL}}(p\parallel q) := \sum_x p(x) [\ln p(x) - \ln q(x)]$$

² 0.01 for smaller models and 0.005 for larger models.

behaviors if the pretraining data contained biases or toxicity. Furthermore, the persistence of mode collapse in GPT-3 models despite the use of PPO-ptx has been documented (Janus, 2022).

These findings suggest that the alignment process using RLHF might be fundamentally problematic and can lead to an “*alignment tax*”—a term used to describe the performance degradation observed in aligned models compared to their base counterparts (Lin et al., 2024). The limitations of the reward function and the difficulty in mitigating mode collapse through modifications to the PPO algorithm or hyperparameters highlight the need for alternative approaches.

As proposed by (Ouyang et al., 2022), a potentially more effective approach to reducing biases and toxicity in language models, while preserving their creative potential, could be to focus on the quality and diversity of the pretraining data itself. By carefully curating and filtering the pretraining data to ensure that the model learns from high-quality, diverse, and unbiased examples, we may be able to mitigate the need for extensive post-hoc alignment techniques like RLHF, which can inadvertently lead to mode collapse and reduced creativity.

6. Conclusion

In this paper, we have investigated the impact of the Reinforcement Learning from Human Feedback (RLHF) alignment process on the creativity and output diversity of Large Language Models (LLMs). Our experiments, conducted using the Llama-2 series of models, have revealed that while RLHF is effective in reducing biases and toxicity in LLMs, it may inadvertently lead to a reduction in the models’ creative potential, defined as the ability to generate outputs with high syntactic and semantic diversity.

We have taken a foundational approach to studying this problem by examining the issue at both the semantic and syntactic levels through three experiments. Experiment 1 demonstrated the impact of RLHF on creativity in a practical marketing context by comparing the diversity of customer personas and product reviews generated by base and aligned models. Experiment 2 investigated the semantic diversity of the models’ outputs, revealing that aligned models form distinct clusters in the embedding space, indicating a fundamentally limited range of outputs compared to their base counterparts. Experiment 3 delved into the syntactic diversity, showing that aligned models exhibit lower entropy in token predictions. This suggests that the cause of this reduction in model creativity is the fact that many token trajectories become blocked during the RLHF process, i.e., the model loses its ability to produce certain tokens (their probability becomes almost zero),

even if they have nothing to do with generating toxic or biased content. This makes aligned models function more like deterministic algorithms rather than creative generative models.

Furthermore, we have observed that the aligned model’s outputs tend to gravitate towards specific “attractor states”, a phenomenon related to *mode collapse* in reinforcement learning. This behavior highlights the challenges in preserving the creative potential of LLMs while aligning them with human preferences. In contrast, the base model exhibits greater flexibility and adaptability in its outputs.

The implications of these findings are significant for marketers and other professionals who rely on LLMs for creative tasks, such as copywriting, ad creation, and customer persona generation. The trade-off between consistency and creativity in aligned models should be carefully considered when selecting the appropriate model for a given application. In situations where creativity and diversity are paramount, base models may be more suitable, while aligned models may be preferred when safety and consistency are the primary concerns. Additionally, our results are important for those studying recommendation systems, as the insights can inform the development and optimization of these systems to balance creativity, diversity, and reliability effectively.

It is important to note that while base models offer greater creative potential, they are not directly usable in applications like chatbots. This is where techniques such as prompt engineering become increasingly important. Contrary to the belief that prompt engineering may become obsolete, our findings suggest that these techniques will be more crucial than ever in harnessing the power of base models for various applications. By carefully crafting input prompts that include instructions, examples, or constraints, users can guide the models’ outputs and make them more suitable for specific use cases while still leveraging their creative potential.

A potential area for further investigation is the exploration of various parameters or configurations of the RLHF process, as higher computational costs and resource demands limited our ability to delve into these aspects. Future research could examine how different parameters influence the creativity and output diversity of aligned LLMs. Moreover, additional studies should analyze other unintended consequences of model alignment and RLHF to enhance our understanding of the trade-offs involved in practical applications of these models.

7. References

- Bender, E.M., Gebru, T., McMillan-Major, A., Shmitchell, S., 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜, in: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. Presented at the FAccT '21: 2021 ACM Conference on Fairness, Accountability, and Transparency, ACM, Virtual Event Canada, pp. 610–623. <https://doi.org/10.1145/3442188.3445922>
- Context length in LLMs: All you need to know - AGI Sphere [WWW Document], 2023. URL <https://agi-sphere.com/context-length/> (accessed 6.4.24).
- Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. <https://doi.org/10.48550/arXiv.1810.04805>
- Eloundou, T., Manning, S., Mishkin, P., Rock, D., 2023. GPTs are GPTs: An Early Look at the Labor Market Impact Potential of Large Language Models. ArXiv.
- Franceschelli, G., Musolesi, M., 2023. On the Creativity of Large Language Models. <https://doi.org/10.48550/ARXIV.2304.00008>
- Gehman, S., Gururangan, S., Sap, M., Choi, Y., Smith, N.A., 2020. RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models, in: Cohn, T., He, Y., Liu, Y. (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2020. Presented at the Findings 2020, Association for Computational Linguistics, Online, pp. 3356–3369. <https://doi.org/10.18653/v1/2020.findings-emnlp.301>
- Grace Hopper, 2024. . Wikipedia.
- Head, C.B., Jasper, P., McConnachie, M., Raftree, L., Higdon, G., 2023. Large language model applications for evaluation: Opportunities and ethical implications. New Dir. Eval. 2023, 33–46. <https://doi.org/10.1002/ev.20556>
- Hutto, C.J., Gilbert, E., 2015. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text, Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014.
- Janus, 2022. Mysteries of mode collapse [WWW Document]. URL <https://www.lesswrong.com/posts/t9svvNPNmFf5Qa3TA/mysteries-of-mode-collapse> (accessed 3.21.24).

k-means clustering, 2024. . Wikipedia.

Lambert, N., Calandra, R., 2023. The Alignment Ceiling: Objective Mismatch in Reinforcement Learning from Human Feedback. <https://doi.org/10.48550/ARXIV.2311.00168>

Lee, H., Phatale, S., Mansoor, H., Mesnard, T., Ferret, J., Lu, K., Bishop, C., Hall, E., Carbune, V., Rastogi, A., Prakash, S., 2023. RLAIIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback. <https://doi.org/10.48550/ARXIV.2309.00267>

Lin, Y., Lin, H., Xiong, W., Diao, S., Liu, J., Zhang, J., Pan, R., Wang, H., Hu, W., Zhang, H., Dong, H., Pi, R., Zhao, H., Jiang, N., Ji, H., Yao, Y., Zhang, T., 2024. Mitigating the Alignment Tax of RLHF. <https://doi.org/10.48550/arXiv.2309.06256>

Llama 2 Prompt Template [WWW Document], 2023. URL <https://gpustools.org/llama-2-prompt-template/> (accessed 6.6.24).

Mohammadi, B., 2024. Wait, It’s All Token Noise? Always Has Been: Interpreting LLM Behavior Using Shapley Value [WWW Document]. arXiv.org. URL <https://arxiv.org/abs/2404.01332v1> (accessed 6.7.24).

Myers–Briggs Type Indicator, 2024. . Wikipedia.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C.L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., Lowe, R., 2022. Training language models to follow instructions with human feedback. <https://doi.org/10.48550/arXiv.2203.02155>

Proximal Policy Optimization [WWW Document], 2024. URL <https://spinningup.openai.com/en/latest/algorithms/ppo.html?highlight=kl%20penalty#id2> (accessed 6.8.24).

Reimers, N., Gurevych, I., 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. <https://doi.org/10.48550/arXiv.1908.10084>

Sahoo, P., Singh, A.K., Saha, S., Jain, V., Mondal, S., Chadha, A., 2024. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications [WWW Document]. arXiv.org. URL <https://arxiv.org/abs/2402.07927v1> (accessed 6.4.24).

- Salton, G., Buckley, C., 1988. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.* 24, 513–523. [https://doi.org/10.1016/0306-4573\(88\)90021-0](https://doi.org/10.1016/0306-4573(88)90021-0)
- Shannon, C.E., 1948. A mathematical theory of communication. *Bell Syst. Tech. J.* 27, 623–656. <https://doi.org/10.1002/j.1538-7305.1948.tb00917.x>
- Shen, W., Zheng, R., Zhan, W., Zhao, J., Dou, S., Gui, T., Zhang, Q., Huang, X., 2023. Loose lips sink ships: Mitigating Length Bias in Reinforcement Learning from Human Feedback. *arXiv*. <https://doi.org/10.48550/ARXIV.2310.05199>
- Sparck Jones, K., 1972. A STATISTICAL INTERPRETATION OF TERM SPECIFICITY AND ITS APPLICATION IN RETRIEVAL. *J. Doc.* 28, 11–21. <https://doi.org/10.1108/eb026526>
- Stiennon, N., Ouyang, L., Wu, J., Ziegler, D.M., Lowe, R., Voss, C., Radford, A., Amodei, D., Christiano, P., 2022. Learning to summarize from human feedback. <https://doi.org/10.48550/arXiv.2009.01325>
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C.C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardaş, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P.S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E.M., Subramanian, R., Tan, X.E., Tang, B., Taylor, R., Williams, A., Kuan, J.X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., Scialom, T., 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. <https://doi.org/10.48550/arXiv.2307.09288>
- TurnTrout, MichaelEinhorn, 2023. Mode collapse in RL may be fueled by the update equation.
- van der Maaten, L., Hinton, G., 2008. Visualizing data using t-SNE. *J. Mach. Learn. Res.* 9, 2579–2605.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need, in: *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17*. Curran Associates Inc., Red Hook, NY, USA, pp. 6000–6010.

- Wang, J., Wu, J., Chen, M., Vorobeychik, Y., Xiao, C., 2023. On the Exploitability of Reinforcement Learning with Human Feedback for Large Language Models. <https://doi.org/10.48550/ARXIV.2311.09641>
- Yu, Z.Z., Jaw, L.J., Hui, Z., Low, B.K.H., 2023. Fine-tuning Language Models with Generative Adversarial Reward Modelling. <https://doi.org/10.48550/ARXIV.2305.06176>
- Yuan, Z., Yuan, H., Tan, C., Wang, W., Huang, S., Huang, F., 2023. RRHF: Rank Responses to Align Language Models with Human Feedback without tears. <https://doi.org/10.48550/ARXIV.2304.05302>

Appendices

A1. Additional Tables

Table 2. Sample sentences from the clusters of review embeddings of the customers generated by the base model.

Cluster 0:

- Coffee is a great gift, this gift will keep you warm and let you share it with people you love
- It is really compact, and the temperature is perfect!
- It is so great that I can keep my coffee warm, even if I am far away from it!

Cluster 1:

- I would recommend it to anyone.
- This product is great.
- I highly recommend this product.

Cluster 2:

- This machine is amazing!
- I have an iPhone 6s and I could pair it with my phone.
- This machine is the best one that I've ever bought.

Cluster 3:

- It has an app that works with your smartwatch and lets you control it with your watch so you don't have to walk back to your coffee machine if you are away from it.
- The smartwatch feature doesn't work either.
- I like the fact that it can connect to my smartwatch

Cluster 4:

- This coffee machine is great.
- This is the best coffee machine I have ever seen!
- This coffee machine is great!

Cluster 5:

- I love it so much.
- I saved so much time.
- The design is also very nice!

Cluster 6:

- It doesn't keep my coffee warm if I'm far away from it.
- It is easy to use, it keeps my coffee hot and it is really stylish.
- It also saves money on the electricity bill as I never have to worry about whether the coffee is still warm enough.

Cluster 7:

- It works great.
- It looks sleek and it tastes great.
- I have it with me everywhere I go.

Cluster 8:

- My only problem is that I have to recharge it twice a day because it has a battery life of about 4 hours.
- The only problem is that it is a bit noisy!
- I can't control it.

Cluster 9:

- It's a nice idea to have your coffee stay warm, but the machine has a design issue: it's too big to put on a table.
- Amazing, it keeps my coffee warm no matter how far I am from the machine

- The coffee is always cold when I get back to my office.

Cluster 10:

- It has everything that you need from a coffee machine.
- I have had this coffee machine for about two months and I have not had any issues with it.
- This coffee machine is really useful if you are going to work and you are busy in the morning

Cluster 11:

- It is a little expensive, but it is worth it.
- I cannot wait to get my hands on one
- It is also really affordable.

Cluster 12:

- It doesn't make good coffee
- I can only afford to drink coffee while I am close to it, but I think I might be able to convince my spouse that it's worth it.
- Amazing product, it makes my coffee while I'm away at work.

Cluster 13:

- This product is f***ing cool!
- This product is so cool!
- Amazing.

Table 3. Sample sentences from the clusters of review embeddings of the customers generated by the base model.

Cluster 0:

- 🤔
- 👍🤔
- 👍👍

Cluster 1:

- As a busy professional, I need my coffee to be ready and waiting for me when I get home.
- As someone who is always on-the-go, it's so convenient to be able to keep my coffee warm even when I'm not directly in front of the machine.
- As a busy professional, I'm always on the go and don't always have time to wait for my coffee to brew.

Cluster 2:

- I am absolutely in love with this coffee machine!
- I've been using it every day since I got it and I can already see the difference in my productivity levels - I'm so much more alert and focused throughout the day.
- I would definitely recommend this product to anyone who is always on-the-go and wants to stay energized and focused throughout the day.

Cluster 3:

- Highly recommend!
- I highly recommend it!
- Highly recommend!

Cluster 4:

- This machine is a game-changer - it connects to my smartwatch, so I can program it to keep my coffee warm even when I'm not in the same room.
- It's so convenient and has made my mornings so much more enjoyable.
- I recently purchased the smart coffee machine that connects to my smartwatch and I am absolutely loving it!

Cluster 5:

- I am so impressed with this coffee machine!
- This machine is a game-changer.
- This machine is a game changer.

Table 4. Sample Outputs of the Models in Experiment 2

Initial Prompt: "Grace Hopper was"	
Base Model	Aligned Model
one of the first programmers for IBM's 704 mainframe computer. She was also a trailblazer in computer programming, becoming the first female admiral (in the US Navy) and helping to create the COBOL language, which is still used widely today. "Learn the rules so you know how to break them properly." – Grace Hopper	an American computer scientist and Navy Rear Admiral who was one of the pioneers of computer programming. She is best known for her work on the development of COBOL, a high-level programming language that made it easier for non-technical people to write computer programs...
a mathematician and pioneer in the computer field. She was the first person to program the Harvard Mark I Automatic Sequence Controlled Calculator (a.k.a. "Harvard Mark II"), which was used during WWII, and later led research for the United States Navy. She is also famous for coining the term "bug" in computer programming when an electrical short caused a computer to malfunction; she found that the actual insects were just as annoying as the bugs in her computers. She is buried at Arlington National Cemetery...	an American computer scientist and Navy Rear Admiral who was a pioneer in the field of computer programming. She is best known for her work on the development of COBOL, a high-level programming language that was designed to be easy to read and write. Hopper was also a vocal advocate for the use of computers in business and education, and she worked tirelessly to promote the field of computer science...
born in New York City to a mother of Swedish descent and an Irish father, who worked as a bookkeeper. She attended the University of New York where she graduated in 1928 with a degree in mathematics. Grace went on to graduate school at Yale and obtained her masters in mathematics from Columbia University. In addition to being an accomplished mathematician Hopper is also credited for making major contributions to computer science, especially in the development of early programming languages...	an American computer scientist and Navy Rear Admiral best known for her work on the development of COBOL, a high-level programming language that was designed to be easy to read and write. Hopper was born in 1906 in New York City and grew up in a family of mathematicians and inventors. She earned her PhD in mathematics from Yale University in 1934 and went on to work at the Harvard Mark I computer lab, where she helped develop the first compiler for programming language COBOL...

Table 5. Four typical ways the aligned model can describe Grace Hopper in Experiment 2.

Biographical Sketch	born on November 9, 1906 in New York City. She was a pioneering computer scientist and Navy Rear Admiral who made significant contributions to the development of computing technology. Hopper earned her PhD in mathematics from Yale University in 1934 and began working at Harvard University’s Mark I computer project in 1943. During World War II, she was part of a team that developed COBOL (Common Business-Oriented Language), a programming language designed for business applications. Hopper’s work on COBOL helped pave the way for the development of...
Early Life and Career	an American computer scientist and Navy Rear Admiral who was one of the pioneers of computer programming. She is best known for her work on the development of COBOL, a high-level programming language that made it easier for non-technical people to write computer programs. Grace Hopper was born in 1906 in New York City and grew up during a time when women were not often encouraged to pursue careers in science or technology. Despite these challenges, she became fascinated with mathematics and physics at an early age and went on to study mathematics at Vassar College...
Interesting Facts (in list format)	a pioneering computer scientist who made significant contributions to the field of computer programming and the development of COBOL, a popular programming language. Here are some interesting facts about Grace Hopper: 1. Early Life and Education: Grace Hopper was born on December 9, 1906, in New York City. She grew up in a family of inventors and engineers and developed an early interest in mathematics and science. She earned her Ph.D. in mathematics from Yale University in 1934. 2. Navy Career: During World War II, Hopper joined the United States Navy...
Key Contributions and Legacy	a computer scientist who helped create the first compiler. She was born in 1906 and died in 1992, but her legacy continues to inspire women in technology today. Hopper earned her Ph.D. in mathematics from Yale University in 1934, becoming one of the few women in the field at the time. She joined the Navy Reserve during World War II and was assigned to the Harvard Mark I computer, where she pioneered the use of machine language programming. In the 1950s, Hopper developed COBOL (Common Business Oriented...

A2. The RLHF Process

Reinforcement Learning from Human Feedback (RLHF) is a technique used to align language models with human preferences and values. In (Touvron et al., 2023), the authors employ RLHF to fine-tune the pretrained Llama-2 language model, resulting in an AI assistant that generates more helpful, safe, and aligned responses. This appendix provides an overview of the RLHF process used in the Llama-2 paper.

A1.1. Human Preference Data Collection

The first step in the RLHF process is to collect human preference data, which will be used to train the reward model. In the Llama-2 paper, the authors use a binary comparison protocol, where annotators are asked to write a prompt and then choose between two sampled model responses based on provided criteria. To maximize diversity, the two responses for each prompt are sampled from different model variants and temperatures. Annotators also label the degree to which they prefer their chosen response over the

alternative, using options such as “significantly better”, “better”, “slightly better”, or “negligibly better/unsure”.

A1.2. Reward Modeling

The collected human preference data is used to train a reward model, which takes a model response and its corresponding prompt as inputs and outputs a scalar score indicating the quality of the response in terms of helpfulness and safety. The Llama-2 paper trains two separate reward models: one optimized for helpfulness (Helpfulness RM) and another for safety (Safety RM). The reward models are initialized from pretrained chat model checkpoints to ensure that they have access to the same knowledge as the base model. The model architecture and hyperparameters are identical to those of the pretrained language models, except for the classification head, which is replaced with a regression head for outputting scalar rewards.

A1.3. Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is a popular reinforcement learning algorithm used in the Llama-2 paper to fine-tune the pretrained language model using the reward models. PPO aims to update policies via (“Proximal Policy Optimization,” 2024):

$$\theta_{k+1} = \arg \max_{\theta} \mathbb{E}_{s, a \sim \pi_{\theta_k}} [L(s, a, \theta_k, \theta)]$$

where

$$L(s, a, \theta_k, \theta) := \min \left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)}, \text{clip} \left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)}, 1 - \epsilon, 1 + \epsilon \right) \right) A^{\pi_{\theta_k}}(s, a)$$

- θ represents the parameters of the policy network (the language model).
- $\pi_{\theta}(a|s), \pi_{\theta_k}(a|s)$ are the new and old policy, respectively.
- $A^{\pi_{\theta_k}}(s, a)$ is the advantage function of taking action a under policy π_{θ_k} at state s .
- $\text{clip}(x, a, b)$ truncates $x < a$ and $x > b$.
- ϵ is a hyperparameter that controls the clipping range (usually set to 0.1 or 0.2).

The clipping function in the PPO objective helps to limit the size of the policy updates, ensuring that the new policy does not deviate too far from the old policy. This promotes stability during training. Essentially, clipping acts as a regularizer by disincentivizing

dramatic policy changes, with the hyperparameter ϵ determining the allowable deviation that still benefits the objective.

Pel,

A Programming Language for Orchestrating AI Agents

Behnam Mohammadi

April 04, 2025

Abstract

The proliferation of Large Language Models (LLMs) has opened new frontiers in computing, yet controlling and orchestrating their capabilities beyond simple text generation remains a challenge. Current methods, such as function/tool calling and direct code generation, suffer from limitations in expressiveness, scalability, cost, security, and the ability to enforce fine-grained control. This paper introduces Pel, a novel programming language specifically designed to bridge this gap. Inspired by the strengths of Lisp, Elixir, Gleam, and Haskell, Pel provides a syntactically simple, homoiconic, and semantically rich platform for LLMs to express complex actions, control flow, and inter-agent communication safely and efficiently. Pel’s design emphasizes a minimal, easily modifiable grammar suitable for constrained LLM generation, eliminating the need for complex sandboxing by enabling capability control at the syntax level. Key features include a powerful piping mechanism for linear composition, first-class closures enabling easy partial application and functional patterns, built-in support for natural language conditions evaluated by LLMs, and an advanced Read-Eval-Print-Loop (REPeL) with Common Lisp-style restarts and LLM-powered helper agents for automated error correction. Furthermore, Pel incorporates automatic parallelization of independent operations via static dependency analysis, crucial for performant agentic systems. We argue that Pel offers a more robust, secure, and expressive paradigm for LLM orchestration, paving the way for more sophisticated and reliable AI agentic frameworks.

1 Introduction

The capabilities of Large Language Models (LLMs) have expanded dramatically, moving beyond simple text generation towards executing complex tasks and interacting with external systems. A critical challenge in this evolution is enabling LLMs to perform actions reliably, safely, and expressively. Current industry approaches primarily fall into two categories, each with significant drawbacks.

The first, widely adopted approach is **function calling** or **tool calling** (“Function calling and other API updates,” 2024). Here, programmers pre-define functions, expose their signatures (often as JSON Schema) to the LLM, and the model generates JSON payloads specifying which function to call with what arguments. While useful for simple tasks, this method suffers from several limitations. First, it struggles to represent complex control flow (conditionals, loops), sequential dependencies beyond simple chaining, or parallel execution patterns. In this method conditional logic—even a condition easily verifiable by code—often relies solely on the LLM’s judgment (e.g., “call action A if condition X is met”), which not only can lead to potential inaccuracies, but also results in verification opacity. Moreover, this method does not scale in real-world scenarios: Defining hundreds of functions for complex agents becomes unmanageable, and the resulting large JSON schemas passed to the LLM increase costs and can significantly degrade the model’s reasoning performance. Finally, function calling is a rigid method: Actions are limited strictly to the pre-defined functions. The LLM cannot compose existing tools in novel ways or perform computations not explicitly provided.

The second approach involves letting the LLM **generate code directly** in a general-purpose language like Python (e.g., ChatGPT Code Interpreter, “ChatGPT plugins,” 2024). This offers greater flexibility and allows the LLM to leverage existing libraries. However, it introduces severe challenges. First, running arbitrary code generated by an LLM is inherently dangerous. Prompt injection attacks may trick the LLM into generating malicious code that can compromise the system, exfiltrate data, or perform unintended actions. Sandboxing helps but is complex and not foolproof. Secondly, restricting the LLM’s capabilities within a powerful language like Python is extremely difficult. For example, preventing the use of loops or specific library functions requires sophisticated static analysis or runtime monitoring. Modifying the grammar of languages like Python to disable features at the generation level is impractical due to their complexity (e.g., Python’s EBNF grammar spans hundreds of lines). Converting such large grammars to formats suitable for constrained generation (like regex) results in massive performance degradation.

This research stems from my decade-long fascination with programming language design, sparked by exploring both mainstream and esoteric languages and their unique ideas. During my PhD work on a project leveraging LLMs, the inadequacy of existing action-performing mechanisms became starkly apparent. We needed a way for an LLM to specify complex, conditional, and potentially concurrent actions safely and reliably, particularly in the context of orchestrating multiple AI agents. This led to the development of Pel, an intermediate language specifically designed to be written by LLMs and safely interpreted by the host system.

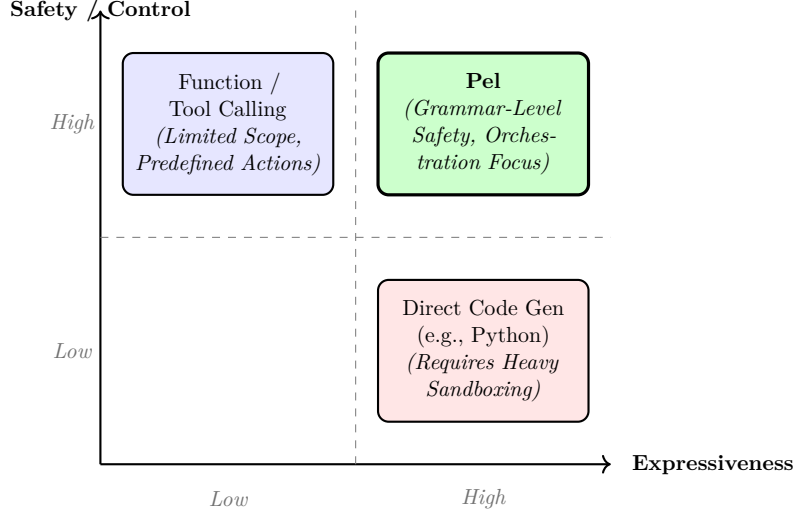


Figure 1: Pel’s Position in the LLM Orchestration Landscape, balancing expressiveness and safety.

Pel combines a Lisp-like syntax with functional programming concepts, resulting in a language with several distinctive characteristics. First, it is safe by design—its simple, regular grammar can be easily modified to restrict LLM generation at the source level, disabling unwanted features (e.g., network access, file I/O, specific functions) with guarantees, eliminating the need for complex sandboxing. Second, Pel is highly expressive, supporting conditionals, loops (as non-strict functions), variable bindings, complex data structures, and powerful composition via pipes (See Figure 1). The language is also inherently LLM-friendly. Its uniform syntax and simple grammar facilitate easier learning (in-context) and reliable generation by LLMs. Its linear composition style (pipes) aligns well with the sequential token generation process of LLMs. Furthermore, Pel provides fine-grained control over execution, including automatic asynchronous execution for performance-critical agentic systems. Finally, it integrates natively with LLM capabilities for tasks like evaluating natural language conditions and providing automated debugging help within its interactive environment.

This paper details the design philosophy, syntax, semantics, and novel features of Pel. We elaborate on its suitability for LLM code generation and its interactive Read-Eval-Print-Loop (aptly named “REPeL”). We also briefly illustrate its application in orchestrating hierarchical AI agent systems, demonstrating Pel’s potential as a foundational technology for building sophisticated and reliable agentic AI systems.

2 Literature Review

The development of agent-based AI systems has seen significant advancement in recent years, particularly with the emergence of large language models (LLMs). Within this domain, researchers have explored various frameworks for agent architecture and orchestration, with systems like AutoGPT and BabyAGI representing early attempts at autonomous agentic systems (Weng, 2023). More formalized approaches have emerged, such as the ReAct framework, which combines reasoning and acting in language agents through a process of thought-action-observation cycles (Yao et al., 2022).

The challenge of reliably coordinating multiple agents has led to the development of multi-agent frameworks like AgentVerse (Chen et al., 2023), which provides a customizable platform for constructing and orchestrating agent societies in various application domains. Similarly, MetaGPT (Hong et al., 2023) proposes a meta-programming framework that enables collaborative problem-solving among multiple agents with specialized roles. These multi-agent systems often struggle with reliable inter-agent communication and coordination, with recent work by Park et al. highlighting challenges in agent-to-agent interaction patterns (Park et al., 2023).

Despite these advances, a critical gap exists in the orchestration of AI agents. Current approaches predominantly fall into two categories, each with significant limitations. The first approach is function calling or tool calling, which allows LLMs to interact with predefined functions but struggles with complex control flow, sequential dependencies, and scalability in real-world scenarios (IBM, 2025). As noted by Microsoft’s Magentic-One research, function calling becomes unwieldy when orchestrating numerous agents with complex interdependencies (Microsoft Research, 2024). The Berkeley Function-Calling Leaderboard highlights ongoing challenges in function calling reliability across different programming languages and API scenarios (Berkeley, n.d.).

Function calling approaches face several critical limitations: they struggle to represent complex control flow patterns (conditionals, loops), provide limited scaling capacity when defining large numbers of functions, and lack expressiveness for novel tool compositions (2024). Additionally, security concerns arise when verification is delegated solely to LLM judgment rather than programmatically verifiable code (BentoML, 2024). These limitations impede the development of sophisticated agentic systems capable of handling real-world tasks with appropriate governance mechanisms (OpenAI, 2024).

The second approach—allowing LLMs to generate code directly in general-purpose languages—offers greater flexibility but introduces severe security risks and challenges in capability restriction. It positions itself in this gap, providing a specialized language with

a restricted grammar that enables safe yet expressive agent orchestration. Unlike function calling frameworks that limit expressivity, or general-purpose language generation that compromises safety, Pel offers a middle ground that addresses both concerns simultaneously.

Pel’s approach to grammar-level safety leverages recent advances in constrained generation techniques. Research on grammar-constrained decoding demonstrates that formal grammars can successfully restrict LLM output to follow specific structures (Geng et al., 2023), ensuring syntactic validity without requiring fine-tuning. The approach of converting context-free grammars to regular expressions for constrained LLM generation has gained traction as demonstrated by tools like ReLLM (Rickard, 2024) and frameworks such as Domino (Wagner et al., 2024), which implement efficient and minimally-invasive constrained decoding.

These constrained generation approaches offer significant advantages over traditional sandboxing methods. As shown by Cooper (Cooper, 2024), constrained decoding guarantees valid outputs on first generation by restricting token distributions via state machines with regex or context-free grammars. This approach is particularly valuable for programming languages, where syntax errors can render generated code unusable. By enforcing grammar constraints at the generation level, Pel can guarantee the syntactic safety of LLM-generated code without compromising expressiveness.

On the programming language design front, a number of DSLs (Domain-Specific Languages) have been designed for specific AI tasks. DSPy (Khatab et al., 2023), for example, introduces a framework for programming foundation models that separates the optimization of prompts from their specification. Beyond such DSLs, though, general-purpose languages such as Lisp have had profound influence on AI, dating back to McCarthy’s original work (McCarthy, 1960). Modern iterations of Lisp like Clojure have demonstrated the continued relevance of homoiconicity and S-expressions for representing and manipulating code as data (Hickey, 2008). Homoiconicity—“the ability to treat code as data” (“Exploring the Power of Artificial Intelligence in Lisp Programming,” 2024)—facilitates metaprogramming and makes it particularly suitable for AI applications where programs need to be generated or manipulated by other programs (SIGPLAN Blog, 2020).

Pel’s error handling system draws inspiration from Common Lisp’s condition system, which has been acknowledged as a sophisticated mechanism for error recovery and program resilience (Pitman, 1988). Elixir and Gleam, which inform Pel’s pipe operator syntax, demonstrate how functional language constructs can enhance code readability and composition (Valim, 2013). The pipe operator (`▷`) enables sequential data transformation that aligns well with LLMs’ token-by-token generation pattern. Additionally, Common Lisp’s error handling system, which influences Pel’s REPeL design, provides sophisticated mechanisms for error recovery (Pitman, 1988) that are particularly valuable when dealing with potentially

faulty LLM-generated code.

Pel’s unique contribution lies in its synthesis of these various influences into a cohesive language specifically designed for LLM orchestration, addressing limitations in both function calling approaches and unrestricted code generation while providing its own novel, expressive, safe, and LLM-friendly programming environment for agentic systems.

3 Design Philosophy

Pel is designed with simplicity, regularity, and expressiveness in mind, tailored for generation by LLMs and safe interpretation. In particular, Pel strives for:

1. **Simplicity and Consistency:** Employ a minimal, regular syntax based on Lisp’s S-expressions. This uniformity makes the language easier for LLMs to learn (better in-context learning) and parse.
2. **Grammar-Level Safety:** Define a concise grammar (expressible easily in EBNF) that can be readily modified. This allows developers to enable or disable language features (specific functions, control flow constructs, network access) *at the grammar level*. By using constrained generation techniques (e.g., regex sampling guided by the grammar), we can *guarantee* that the LLM cannot generate forbidden code paths, eliminating the need for runtime sandboxing for many security concerns (See Figure 2).
3. **Expressiveness for Orchestration:** Include essential control flow structures and data manipulation capabilities as first-class citizens within the language.
4. **Composable Linearity:** Provide mechanisms (like pipes) that allow LLMs to build complex workflows step-by-step without needing to plan the entire structure in advance or backtrack during generation.
5. **Seamless LLM Integration:** Natively incorporate mechanisms for leveraging LLM capabilities where appropriate (e.g., evaluating natural language conditions).
6. **Developer Ergonomics:** Offer an interactive development experience with robust error handling and debugging aids (the REPeL).

The design of Pel did not occur in a vacuum. It stands on the shoulders of giants in programming language history and draws inspiration from several key languages that have fascinated me over the years:

Lisp: The most apparent influence is Lisp, particularly its S-expression syntax (...). This provides a simple, uniform structure (homoiconicity) that is easy to parse and manipulate programmatically. However, Pel diverges significantly from traditional Lisps. It avoids cons cells as the fundamental list structure in favor of distinct parenthesized evaluation lists (...)

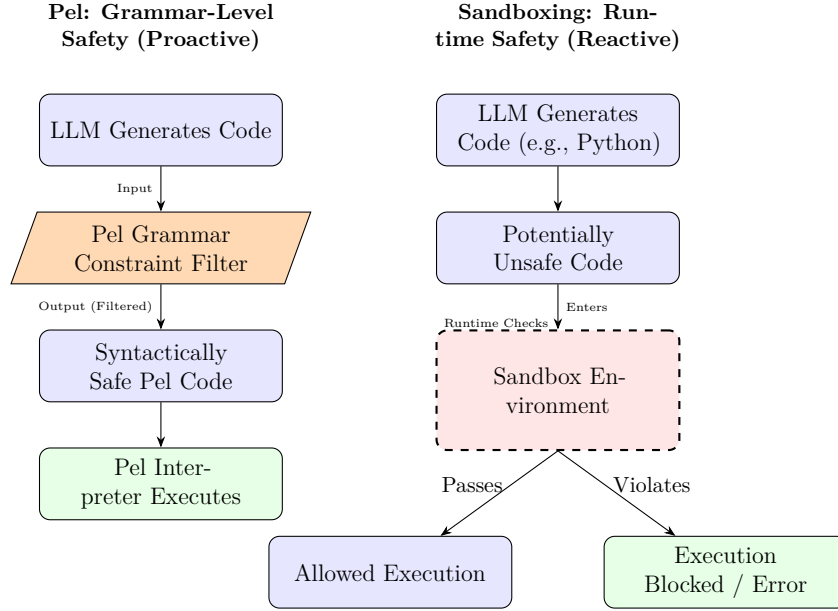


Figure 2: Comparison of Safety Mechanisms: Pel’s Proactive Grammar Constraint vs. Reactive Runtime Sandboxing.

and bracketed literal lists `[...]`. Crucially, Pel enforces a strict operator-first interpretation for `(...)` forms and eliminates special forms entirely, opting for a unified function model.

Elixir and Gleam: The pipe operator (`▷`) is directly inspired by Elixir, providing a linear and readable way to compose functions. Pel extends this concept by using the `^` caret symbol for injecting the piped value into specific argument positions, offering greater flexibility than Elixir’s default first-argument injection. This linearity is particularly advantageous for LLMs, which generate code sequentially.

Haskell: Pel incorporates automatic partial application, a hallmark of Haskell and other ML-family languages. When a function (or `PelClosure`) is called with fewer arguments than it expects, it automatically returns a new closure capturing the provided arguments, rather than raising an error. This is possible because Pel functions have a fixed arity (no variadic arguments), allowing the interpreter to “know” when a function is full and ready to fire. This simplifies the creation of higher-order functions and functional composition patterns. Pel’s emphasis on immutability also echoes functional programming principles championed by Haskell.

Common Lisp: The design of Pel’s interactive environment, the REPeL, draws inspiration from the powerful condition and restart system of Common Lisp. The ability to intercept errors, inspect the context, and choose how to proceed (e.g., retry, abort, provide a value, rewrite code) offers a much more robust development experience compared to typical REPLs,

especially relevant when dealing with potentially faulty code generated by LLMs.

Pel aims to provide a unique and effective solution by synthesizing these influences within the specific context of LLM interaction while offering its own novel contributions.

4 The Pel Language

Pel is designed to be minimal yet expressive. Its core components are described below.

4.1 Syntax and Grammar

Pel uses a Lisp-like syntax based on S-expressions. Code consists of atoms and lists.

- **Atoms:** Basic indivisible values like numbers (`PelNum`), strings (`PelString`), booleans (`PelBool`), the null value (`PelNil`), symbols (`pelSymbol`), and keywords (`PelKey`).
- **Lists:** Sequences of elements enclosed in delimiters. Pel distinguishes between:
 - **Parenthesized Lists `()`:** Interpreted as expressions. The first element is *always* treated as the operator (a function/closure to be called), and the remaining elements are its arguments. Example: `(+ 1 2)`.
 - **Bracketed Lists `[]`:** Interpreted as literal data lists (`PelListLiteral`). Elements are evaluated, but the list itself is treated as data, not a function call. Example: `[1 #t "hello"]`. This avoids the ambiguity present in some Lisps where `()` can mean both function call and data list. Notice that literal lists are heterogeneous, unlike Python lists that can only contain one type of data.

This strict interpretation of `()` simplifies parsing and evaluation logic. The grammar is designed to be small and regular, making it amenable to constrained generation.

4.1.1 Pel EBNF Grammar

```
{(* Entry point: A program is zero or more expressions *)}
program = { expression } ;
(* An expression is a primary expression, potentially chained with pipes *)
expression = primary , { PIPE , primary } ;
(* A primary expression is the base unit before considering pipes *)
primary = atom
        | list
        | literal_list
        | quoted_expression
        ;
```

```

(* Atomic literal values *)
(* Note: KEY is included here. The parser interprets it contextually *)
(*     as either a literal key or the start of a key-value pair. *)
(*     The grammar allows KEY to be followed by another expression *)
(*     within lists or at the top level, covering the pair syntax. *)
atom = BOOL | NIL | NUMBER | STRING | SYMBOL | KEY ;
(* Standard parenthesized list *)
list = LPAREN , { expression } , RPAREN ;
(* Bracketed literal list *)
literal_list = LBRACKET , { expression } , RBRACKET ;
(* Quoted expression *)
(* The parser ensures the content is interpreted with allow_pairs=False *)
quoted_expression = QUOTE , expression ;
(* Terminal Symbols (provided by the tokenizer) *)
LPAREN    = "(" ;
RPAREN    = ")" ;
LBRACKET  = "[" ;
RBRACKET  = "]" ;
QUOTE     = "'" ;
PIPE      = ">" ;
BOOL      = ? "#t" | "#f" ?
; (* Based on regex r"#t|#f" *)
NIL       = "#nil"
; (* Based on regex r"#nil" *)
STRING    = ? C-style string literal ?
; (* Based on regex r'"[^"]*"')
KEY       = ? keyword starting with ':' ?
; (* Based on regex r":[a-zA-Z0-9_-\+*/\?!\<=>.]+" *)
NUMBER    = ? integer or float literal ?
; (* Based on regex r"-?\d+(\.\d+)?" *)
SYMBOL    = ? sequence of non-whitespace, non-delimiter chars ?
; (* Based on regex r"^[^s()\[\]\\";|>]+" *)
(* Ignored Tokens: comment, whitespace *)

```

The simplicity of this grammar is crucial. It allows for easy conversion to regular expressions for use in LLM constrained generation frameworks (like Guidance or LMQL). Developers can easily modify this EBNF (e.g., remove the **PIPE** rule, remove specific **SYMBOLS** corresponding to built-in functions) to create variants of Pel with restricted capabilities, enforced at the generation stage.

4.2 Data Types

Pel supports a range of fundamental data types:

- **PelNum**: Represents integer and floating-point numbers.
- **PelString**: Represents UTF-8 text strings, enclosed in double quotes (`"`).
- **PelBool**: Represents boolean values `#t` (true) and `#f` (false).
- **PelNil**: Represents the null or absence of value, denoted `#nil` or `()`. Importantly, unlike Python’s `None`, `PelNil` does *not* evaluate to false in conditional contexts. Its truthiness is undefined.
- **PelListLiteral**: Represents 1-indexed heterogeneous lists of `PelValues`, enclosed in brackets `[]`. Example: `[1 "two" #t #nil]`.
- **PelKey**: Represents keyword symbols, prefixed with a colon `:`. Example: `:name`, `:age`. Keywords are often used for named arguments and creating map-like structures.
- **PelPair**: Represents a key-value association, typically formed implicitly by the parser when a `PelKey` is followed by a non-keyword value within a list. Example: `:name "Pel"` within `[:name "Pel" :version 1]`. A standalone key like `:flag` is implicitly paired with `#nil`.
- **PelClosure**: The internal representation of all callable entities (functions and lambdas) in Pel.

4.3 Functions and Closures (PelClosure)

Functions are central to Pel.

- **Uniformity**: There are no “special forms” in the traditional Lisp sense. Constructs like `if`, `for`, `case`, and even `def` are implemented as functions (specifically, `PelClosures`).
- **Definition**: Functions and variables are defined using the same `def` construct. Defining a variable binds a symbol to a value; defining a function binds a symbol to a `PelClosure` (typically created via a `lambda`).

```
(def pi 3.14)           ; Variable definition
(def greet (lambda [:name] (print "Hello, " name))) ; Function definition
```

Thanks to Pel pipes, one can also define functions in “reverse” order—first creating the body and then piping that into a symbol:

```
(lambda [:x :y]
  [(pow x 2) (pow y 2)] ▷ (+) ▷ (sqrt))
```

```
) ▷ (def my-foo ^)
```

- **PelClosure:** This is the heart of Pel's function mechanism. A `PelClosure` encapsulates:
 - The function's code (or a reference to a built-in implementation).
 - The environment in which it was defined (closing over variables).
 - Specifications for its arguments (`ArgSpec`), including names and optional default values.
 - A flag indicating whether it's **strict** (evaluates arguments before invocation) or **non-strict** (arguments passed unevaluated).
- **Partial Application:** Inspired by Haskell, Pel supports automatic partial application. If a `PelClosure` is called with fewer arguments than required, it does not raise an error. Instead, it returns a *new* `PelClosure` that has captured the provided arguments and is waiting for the remaining ones.

```
(def add (lambda [:x :y] (+ x y))) ; Defines add function
(def add5 (add 5)) ; Creates a new closure, add5, where x is bound to 5
(add5 10) ; Invokes the partial closure, returns 15
```

This applies to all functions, including control flow constructs, making them first-class citizens that can be passed around partially applied. For instance, a `for` loop missing its body can be passed as an argument to another function.

- **Strict vs. Non-strict:** Strict functions (the default for lambdas) evaluate their arguments before the function body is executed. Non-strict functions receive their arguments as unevaluated expressions. This is essential for control flow like `if`, which must only evaluate the **then** or **else** branch, not both. Currently, only built-in functions can be non-strict, but user-defined non-strict functions are planned. Non-strict functions in Pel provide an alternative approach to Lisp macros in that they operate directly on the AST and can selectively evaluate their arguments just like the `unquote` mechanism in macros.
- **Argument Passing:** Closures can be called using:
 - **Positional arguments:** `(area 3 4 5)`
 - **Named arguments:** `(area :x 3 :y 4 :z 5)`. Mixing positional and named arguments in a single call is disallowed. Lambdas can define default values for arguments: `(lambda [:x :y 10] ...)` makes `y` default to `10`.

4.4 Piping (\triangleright , \wedge)

Pel provides a powerful piping mechanism for composing functions linearly, inspired by Elixir.

- **Basic Pipe \triangleright :** The \triangleright operator takes the result of the expression on its left and passes it as the *first* argument to the function call expression on its right.

```
[1 2 3 4]  $\triangleright$  (len)  $\triangleright$  (+ 5) ; Equivalent to (+ (len [1 2 3 4]) 5)  $\Rightarrow$  9
```

- **Caret Injection \wedge :** The caret symbol \wedge acts as a placeholder within the right-hand expression, indicating exactly where the result of the left-hand expression should be injected.

```
"world"  $\triangleright$  (concat "hello, "  $\wedge$ ) ; Equivalent to (concat "hello, " "world")  
5  $\triangleright$  (* 10  $\wedge$ ) ; Equivalent to (* 10 5)
```

The caret injection works recursively within nested structures:

```
[1 2 3]  $\triangleright$  (print :vals ["a list of items:"  $\wedge$ ] :sep " ")  
; Equivalent to (print :vals ["a list of items:" [1 2 3]] :sep " ")
```

- **Linearity Advantage for LLMs:** This piping mechanism is particularly beneficial for LLMs. They generate code token by token, moving forward. Unlike traditional nested function calls `bar(foo(a))` where the LLM needs to plan the `bar(` call *before* generating `foo(a)`, Pel's pipes allow the LLM to generate `(foo a)`, then decide to pipe it into `bar` *after the fact*: `(foo a) \triangleright (bar \wedge)`. This linear flow aligns better with the sequential nature of LLM generation and reduces the need for complex planning or backtracking.

4.5 List Operations and Accessing

Pel provides powerful mechanisms for working with literal lists. While literal lists appear as simple data structures, they actually behave as `PelClosures`. This design choice enables sophisticated list manipulation operations through a unified function calling syntax.

A literal list is can receive three optional arguments that default to `#nil`: `:at`, `:from`, and `:to`. When a literal list is “called” without arguments, it returns all its elements. However, providing one or more of these arguments enables advanced list slicing and key-value lookup operations:

```
; ❶ Basic indexing (returns element at index 1)  
([5 6 7 8] :at 1) ; => 5
```

```

; ❷ Slicing to a specific index (inclusive)
([5 6 7 8] :to 2) ; => [5 6]
; ❸ Slicing from a specific index (inclusive)
([5 6 7 8] :from 2) ; => [6 7 8]
; ❹ Combined slicing with both from and to indices
([5 6 7 8] :from 1 :to 3) ; => [5 6 7]
; ❺ Retrieving multiple elements by index
([5 6 7 8] :at [1 3]) ; => [5 7]
; ❻ Key-value lookup (requires quoting the key)
([:a 1 :b 2 :c 3] :at ':a) ; => 1
; ❼ Multiple key lookup
([:a 1 :b 2 :c 3] :at [':a ':c]) ; => [1 3]
; ❽ Index-based retrieval of multiple pairs
([:a 1 :b 2 :c 3] :at [1 3]) ; => [:a 1 :c 3]

```

For key lookup operations (as in the sixth example), the key must be quoted to prevent Pel from interpreting it as a separate key-value pair. Without quoting, Pel would interpret `:at :a` as two separate pairs: `PelPair(PelKey(":at"), #nil)` and `PelPair(PelKey(":a"), #nil)`. Literal lists with alternating keys and values are automatically interpreted as containing key-value pairs. Internally, `[:a 1 :b 2 :c 3]` is represented as `[PelPair(PelKey(":a"), PelNum(1)), ...]`. This means standard list operations like indexing still work naturally. Named arguments can be omitted for brevity, with positional arguments being mapped to the list's parameters in order (`:at`, `:from`, `:to`):

```

; Implicit :at parameter (equivalent to :at 1)
([5 6 7 8] 1) ; => 6
; Implicit :from and :to (equivalent to :from 1 :to 3)
([5 6 7 8] () 1 3) ; => [6 7 8]

```

Since literal lists behave like closures, they integrate seamlessly with Pel's piping mechanism:

```

(def data [1 2 3 4 5])
(for [0 2 4] i
  i ▷ (data :at ^) ▷ (print))
; Prints elements at indices 0, 2, 4 in data

```

This unified treatment of data structures as closures exemplifies Pel's functional design philosophy and demonstrates how even basic language constructs can offer rich, expressive capabilities through a consistent interface.

4.6 Control Flow

Control flow constructs are implemented as non-strict `PeiClosures`.

4.6.1 `if`

Takes `:cond`, `:then`, and optional `:else` arguments (defaulting to `#nil`). Evaluates `:cond`; if true, evaluates and returns `:then`, otherwise evaluates and returns `:else`.

```
(if data ▷ (len) ▷ (gt 2)
  (print "length of data is greater than 2")
  (print "data is too short"))
```

Note that since even control flow constructs are functions, one can also call them using named arguments and pipe into them:

```
data ▷ (len) ▷ (gt 2) ▷
  (if :cond ^
    :then (print "length of data is greater than 2")
    :else (print "data is too short"))
```

4.6.2 `case`

A generalized conditional structure. Takes a value (`:scrut`) and a literal list of condition-consequence pairs (`:body`). It evaluates conditions sequentially. The first condition that evaluates to `#t` causes its corresponding consequence to be evaluated and returned. A final `#t` condition acts as a default `else` clause.

```
(case my-list [
  (len) ▷ (gt 5)
  "length of my-list is greater than 5"
  #t ; Default case
  (print "all conditions failed")
]) ▷ (print)
```

Notice that `case` pipes its `:scrut` into each condition; turning the first condition into `my-list ▷ (len) ▷ (gt 5)`.

4.6.3 `for`

Provides iteration. Takes `:coll` (the collection to iterate over), `:iterator` (a symbol to bind each item to), and `:body` (an expression to evaluate for each item). Importantly, it returns a

`PelListLiteral` containing the results of each body evaluation. The length of output is the same as `:coll`.

```
(for :coll [1 2 3] :iterator i :body (* i 2)) ; ⇒ [2 4 6]
```

4.6.4 `do`

Evaluates a sequence of expressions (provided as arguments or in a `PelListLiteral`) and returns the value of the *last* expression. Useful for side effects.

```
(do
  (print "Starting...")
  (def x 5)
  (+ x 10))
; Prints "Starting...", returns 15
```

4.6.5 `do/async`

Similar to `do`, but evaluates the expressions concurrently. Returns the result of the last expression specified in the block, after all have completed.

4.7 Natural Language Integration

Recognizing that Pel is often generated or used in conjunction with LLMs, it incorporates direct hooks for LLM evaluation:

- **Natural Language Conditions in `case`:** If a condition expression within a `case` body is a `PelString`, Pel interprets it as a natural language condition. It passes the `:scrut` (the value being tested) and the condition string to an underlying LLM. The LLM's boolean response determines whether the condition passes.

```
(case user-profile [
  "is a premium member" (grant-access user-profile)
  "has incomplete profile" (prompt-completion user-profile)
  #t (show-basic-view)
])
```

Here, the strings are evaluated by an LLM against the `user-profile` data.

- **Other LLM Functions:** Pel can easily incorporate other built-in functions that call LLMs, such as `summarize`, which takes text and returns an LLM-generated summary.

This tight integration allows leveraging the fuzzy understanding capabilities of LLMs directly within the structured logic of Pel.

5 The Pel Runtime Environment

Pel is more than just a language specification; it includes a runtime environment designed for interactive development and efficient execution, especially in agentic scenarios.

5.1 The REPeL (Read-Eval-Print-Loop)

Pel features an enhanced REPL, affectionately termed “REPeL,” which incorporates advanced error handling and debugging features inspired by Common Lisp’s condition system and augmented with LLM capabilities. As an interactive development environment, REPeL provides a standard loop for entering Pel code, seeing results, and inspecting the environment. It includes features like command history, syntax highlighting, and auto-completion. But perhaps the most intriguing aspect of REPeL is the way it handles errors. To illustrate this, let us review, as an example, a piece of Pel code that calls two agents, accumulates their responses, and summarizes the result:

```
(FIN-AGENT :query "give me the latest financial report for this quarter"
           :expect "string") ▷ (def financial_report ^)
(SALES-AGENT :query "I need our sales data in detail"
             :expect "string") ▷ (def sales_data ^)
(add financial_report sales_data) ▷ (summarize) ; erroneous code
```

In this example, two API calls are made to agents, followed by a call to the **add** function. The latter is problematic because the correct way to concatenate two strings in Pel is to use the **concat** function. In such situations, most programming languages (including Python) would throw an exception and exit, discarding the results obtained in the previous lines. Since the API calls could be slow and potentially expensive, we need a way to keep the results obtained from the first two lines while being able to fix the error on the third. Therefore, when any error happens in Pel code (and a **PelException** is raised internally), the REPeL does not simply crash; it preserves the state of the environment *before* the error and presents the user (or an automated system) with restart options:

1. **Rewrite Entire Program:** Discard the current code and enter a completely new program.
2. **Rewrite from Error Forward:** Keep the code up to the error point, discard the rest, and enter new code to replace the faulty part onwards.

3. **Rewrite Current Expression:** Replace only the specific expression that caused the error and retry its evaluation within the original context.
4. **Abort Evaluation:** Stop the current evaluation entirely.
5. **Self-Healing Mode (Helper Agent):** Invoke an LLM-based helper agent to automatically fix the code.

The **LLM-assisted self-healing** is a key innovation of REPeL. When an error occurs, the associated `PelException` captures not only the error message and location but also contextual information, often derived from the docstring of the `Pel` function where the error originated. A dedicated “Helper Agent” (an LLM prompted with the error, the faulty code snippet, and the function’s documentation/context) is invoked. It analyzes the discrepancy between the code and the expected usage (based on the docstring) and proposes a corrected code snippet. In automatic mode, the REPeL can accept this correction and continue execution seamlessly. This provides an “autocorrect” experience, significantly improving robustness, especially when LLMs are generating `Pel` code.

```
Pel> (def name "Behnam")  
      (print ["hello" name] :sep " ")  
  
⇒ "Behnam"  
Error at line 2, col 1-32: Mixing named and positional arguments is not allowed.  
  1 | (def name "Behnam")  
  2 | (print ["hello" name] :sep " ")  
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  
error context:  
FUNCTION SIGNATURE: (print :vals :sep " " :nl #t)  
TYPES:  
  - vals: PelValue - values to print, can be a single value or a literal list [ ... ]  
  - sep: PelString (optional) - separator string, default ""  
  - nl: PelBool (optional) - whether to end with a newline, default #f  
DESCRIPTION:  
  Prints values to stdout. If vals is a bracket-literal, prints each item.  
  Optionally separates with the given separator string and adds a newline.  
  Returns the input vals unchanged.  
EXAMPLE USAGE:  
<hidden for brevity>  
  
Possible restarts:  
1. Rewrite entire program
```

```

2. Rewrite from error point forward
3. Rewrite only the current expression
4. Abort evaluation
5. Use self-healing mode
Select option (1-4/5): 5
SELF-HEALING...
Helper agent proposed rewrite:
(print :vals ["hello" name] :sep " ")
Press 'a' to accept, 'e' to edit, 'r' to abort.
Choice (a/e/r)? a

1 (print :vals ["hello" name] :sep " ")

hello Behnam
⇒ ["hello" "Behnam"]

```

5.2 Automatic Asynchronicity

Performance is critical for agentic systems where multiple tasks or agent interactions might need to occur concurrently. Pel addresses this through an optional automatic asynchronous execution mode within the REPeL.

- **Dependency Graph Analysis:** Before execution in this mode, the REPeL performs a pre-scan of the entire code. It parses the code into a sequence of top-level Abstract Syntax Trees (ASTs). For each AST, it identifies the symbols it *uses* and the symbols it *defines* (we which specifically look for **def** forms).
- **Parallel Execution:** Based on this analysis, the runtime builds a dependency graph. ASTs that do not depend on the output (defined symbols) of other pending ASTs are considered independent and can be scheduled for concurrent execution using **asyncio** tasks, managed by the **PelTaskManager**. Pel's functional nature and emphasis on immutability simplify this dependency analysis.

This automatic parallelization can significantly speed up Pel programs, especially those involving independent computations or I/O-bound operations like multiple agent calls, without requiring the programmer (or the LLM generating the code) to explicitly manage threads or async primitives everywhere. While this mode offers performance gains, the top-to-bottom execution mode provides more predictable debugging with the restart system; refining error handling in the fully async mode is an area of ongoing work.

6 Application: Orchestrating Agentic AI Systems

The primary motivation for Pel was to enable more sophisticated coordination of LLM-based agents. Here we demonstrate how Pel can be leveraged to build a hierarchical multi-agent system.

In this example, an organization is modeled as a hierarchy of agents (Figure 3 shows the organizational hierarchy.). For example, there could be a “mini” department of Marketing, a mini department of Finance, etc., each with their own sub-departments. Agent data (roles and backgrounds, available tools for function calling, list of sub-agents) is loaded from a JSON file. In this hierarchy, agents can be designated as **routers** or **terminals**. Terminal agents perform specific tasks (potentially using traditional function calling or direct LLM responses). Router agents, crucially, coordinate their sub-agents.

Pel as the Orchestration Language: Instead of relying on complex internal logic or limited function calls, router agents achieve coordination by generating and executing Pel code. A router agent, when tasked by its supervisor, might write Pel code that:

- Calls one or more of its sub-agents sequentially or in parallel (**do/async**).
- Uses the output of one sub-agent as input (**:context**) for another, facilitated by Pel’s variables (**def**) and piping (**▷**).
- Implements conditional logic (**if, case**) based on sub-agent responses to decide the next steps.
- Initiates collaborative sessions among sub-agents using the built-in **meeting** function, which takes a list of agent names, a topic, and simulates discussion rounds. The transcript can then be processed (e.g., summarized using **summarize**).

For instance, when the user asks the **MAIN** agent to come up with a comprehensive plan for social media advertising, the agent writes the following Pel code which queries the relevant agents (**FINANCE** and **MARKETING**) for more information and aggregates the results in a list. That list, once evaluated, is given back to **MAIN**, which it then uses to provide an answer to the user:

```
; MAIN's Pel code

(MAIN/FINANCE :query "what's the budget allocation
                    for social media advertising?"
              :expect "num")
▷ (def social_media_budget ^)

(MAIN/MARKETING :query "come up with an effective advertising campaign
```

```

        on social media given the given budget"
        :context social_media_budget
        :expect "string")
    ▷ (def social_media_strategy ^)

[:social_media_budget social_media_budget
 :social_media_strategy social_media_strategy]

```

Notice that the call to **MARKETING** itself might trigger that agent to write a piece of Pel code to orchestrate its subagents:

```

; MARKETING's Pel code

(meeting :group ["MAIN/MARKETING/SOCIAL_MEDIA"
                 "MAIN/MARKETING/CONTENT_MARKETING"]
         :rounds 3
         :topic "come up with a great advertising campaign given the budget"
         :context social_media_budget)
    ▷ (summarize)
    ▷ (def plan_summary ^)

[:plan_summary plan_summary]

```

This Pel-based approach provides far greater expressiveness and flexibility than function calling. Complex, multi-step, conditional workflows involving multiple agents can be explicitly defined and executed. Compared to generating Python, Pel offers inherent safety benefits through its controlled grammar and avoids the need for complex sandboxing. Moreover, if a router agent generates syntactically incorrect or semantically flawed Pel code (e.g., calls a non-existent sub-agent, uses a function incorrectly), the REPeL’s restart mechanism kicks in. In an automated setup (with self-healing), the helper agent can often fix the Pel code, allowing the agentic system to recover gracefully from errors that would halt systems relying on less robust execution models.

Pel thus serves as the connective tissue, the “language of thought and action,” for these hierarchical agentic systems, enabling complex emergent behaviors through structured, safe, and expressive code.

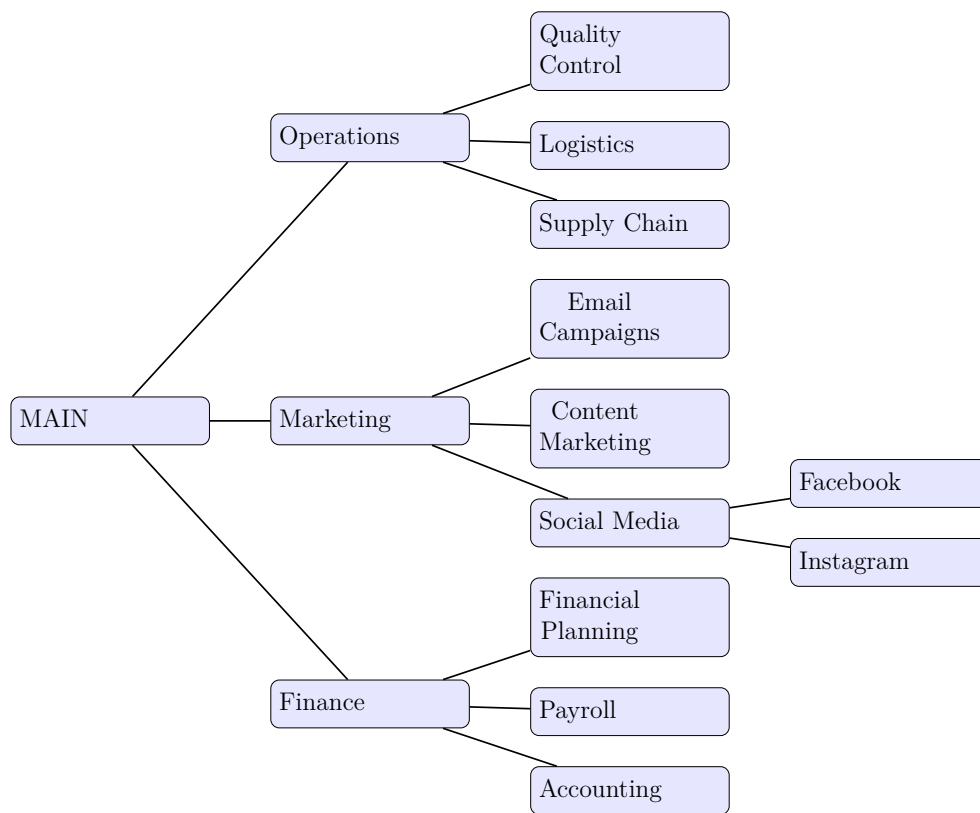


Figure 3: Organizational structure diagram

7 Discussion and Future Work

This paper introduces Pel, a specialized language designed specifically for Large Language Model (LLM) orchestration. Pel provides a dedicated, secure, and expressive environment for LLM code generation that successfully navigates between the limitations of basic function calling and the security risks of unrestricted general-purpose language generation.

At its core, Pel leverages a simple, Lisp-inspired grammar that enables fine-grained control over LLM-generated code capabilities at the syntax level. This design choice allows for grammar modification and constrained generation, substantially reducing security risks without requiring complex sandboxing mechanisms. Despite this syntactic simplicity, Pel achieves enhanced expressiveness by incorporating control flow constructs (if, case, for) as first-class functions and implementing a powerful piping mechanism inspired by Elixir and Gleam. These features enable complex, linearly-composed workflows that are particularly well-suited for LLM generation.

A key innovation in Pel is its seamless integration with LLMs, natively supporting natural language conditions within its control flow and delegating their evaluation to LLMs when necessary. This integration extends to Pel’s advanced development environment (REPeL), which features an interactive REPL with Common Lisp-style restarts and LLM-powered “Helper Agents” for automated error diagnosis and correction. Furthermore, Pel’s runtime can automatically detect and parallelize independent code blocks by analyzing dependencies in the Abstract Syntax Tree (AST), enhancing performance for agentic systems.

Pel emerges from both practical necessity and appreciation for powerful programming language paradigms, offering a novel solution to the critical challenge of LLM orchestration and agency. By rejecting the false dichotomy between overly simplistic function calling and insecure general-purpose code generation, Pel establishes a middle ground: a language specifically designed for LLMs to use safely and effectively. As LLMs become increasingly integrated into complex workflows and autonomous systems, languages like Pel, which bridge the gap between natural language understanding and structured, safe execution, will be essential. Pel offers a promising foundation for building more capable, reliable, and controllable AI agents, representing a significant step toward specialized languages for interacting with and controlling LLMs. Its core contributions lie in its unique combination of syntactic simplicity, grammar-level safety, expressive pipeline-style programming, and advanced error recovery in its REPeL through self-healing agents and a robust restart system.

7.1 Limitations and Future Directions

Currently, only built-in functions can be non-strict, but enhancing expressiveness by allowing users to define their own non-strict functions remains a significant area for improvement, albeit one that requires careful semantic design. Additionally, work is ongoing to improve the predictability and usability of the restart mechanism within the fully automatic asynchronous error handling mode. While Pel has demonstrated impressive few-shot learnability, another promising direction involves fine-tuning LLMs, even small models, specifically for Pel code generation, which could substantially improve code quality and reduce errors. Future work could concentrate on addressing these areas for improvement.

References

2024. ToolACE: Winning the Points of LLM Function Calling. arXiv preprint.
- BentoML, 2024. Function Calling with Open-Source LLMs [WWW Document]. URL <https://bentoml.com/blog/function-calling-with-open-source-llms> (accessed 4.2.2025).
- Berkeley, n.d. Berkeley Function Calling Leaderboard [WWW Document]. URL https://gorilla.cs.berkeley.edu/blogs/8_berkeley_function_calling_leaderboard.html (accessed 4.2.2025).
- ChatGPT plugins [WWW Document], 2024. URL <https://openai.com/index/chatgpt-plugins/>
- Chen, Z., Wang, H., Zhang, Z., Xie, C., Liu, G., Du, Y., Li, L., 2023. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors. arXiv preprint.
- Cooper, A., 2024. A Guide to Structured Generation Using Constrained Decoding [WWW Document]. URL <https://www.aidancooper.co.uk/constrained-decoding/> (accessed 4.2.2025).
- Exploring the Power of Artificial Intelligence in Lisp Programming [WWW Document], 2024. URL <https://aiforsocialgood.ca/blog/artificial-intelligence-and-its-parallels-with-lisp-an-in-depth-exploration-of-the-core-principles> (accessed 4.2.2025).
- Function calling and other API updates [WWW Document], 2024. URL <https://openai.com/index/function-calling-and-other-api-updates/>
- Geng, S., Köpf, B., Schuster, T., Roberts, A., 2023. Grammar-Constrained Decoding for Structured NLP Tasks without Finetuning. arXiv preprint.
- Hickey, R., 2008. The Clojure Programming Language. Proceedings of the 2008 Symposium on Dynamic Languages.
- Hong, S., Wang, X., Li, H., Gu, J., Tang, S., 2023. MetaGPT: Meta Programming for Multi-Agent Collaborative Framework. arXiv preprint.
- IBM, 2025. What is AI Agent Orchestration? [WWW Document]. URL <https://www.ibm.com/ai/agent-orchestration/>

- com/think/topics/ai-agent-orchestration (accessed 4.2.2025).
- Khattab, O., Santhanam, K., Li, X., Hall, D., Liang, P., Potts, C., Zaharia, M., 2023. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. arXiv preprint.
- McCarthy, J., 1960. Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I. Communications of the ACM.
- Microsoft Research, 2024. Magentic-One: A Generalist Multi-Agent System for Solving Complex Tasks [WWW Document]. URL <https://www.microsoft.com/en-us/research/articles/magentic-one-a-generalist-multi-agent-system-for-solving-complex-tasks/> (accessed 4.2.2025).
- OpenAI, 2024. Practices for Governing Agentic AI Systems [WWW Document]. URL <https://cdn.openai.com/papers/practices-for-governing-agentic-ai-systems.pdf> (accessed 4.2.2025).
- Park, J., Kim, M., Kwon, S., Kang, D., 2023. Generative Agent-Based Modeling: Unveiling Social System Dynamics through Coupling Large Language Models with Agent-Based Modeling. arXiv preprint.
- Pitman, K., 1988. The Common Lisp Condition System. Lisp Pointers.
- Rickard, M., 2024. Context-Free Grammar Parsing with LLMs [WWW Document]. URL <https://matrickard.com/context-free-grammar-parsing-with-llms> (accessed 4.2.2025).
- SIGPLAN Blog, 2020. Homoiconicity, Lisp, and Program Synthesis [WWW Document]. URL <https://blog.sigplan.org/2020/03/25/homoiconicity-lisp-and-program-synthesis/> (accessed 4.2.2025).
- Valim, J., 2013. Elixir: A Modern Approach to Programming for the Erlang VM. O'Reilly Media.
- Wagner, V., Beurer-Kellner, L., Cüre, M., Vechev, M., 2024. Guiding LLMs The Right Way: Fast, Non-Invasive Constrained Generation. arXiv preprint.
- Weng, L., 2023. LLM-powered Autonomous Agents. Lilian Weng's Blog.
- Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., Cao, Y., 2022. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv preprint.