Mechanism Design for Large Language Models

Paul Dütting [*1] **Vahab Mirrokni** [*1] **Renato Paes Leme** [*1] **Haifeng Xu** [*1 2] **Song Zuo** [*1]

Abstract

We investigate auction mechanisms to support the emerging format of AI-generated content. We in particular study how to aggregate several LLMs in an incentive compatible manner. In this problem, the preferences of each agent over stochastically generated contents are described/encoded as an LLM. A key motivation is to design an auction format for AI-generated ad creatives to combine inputs from different advertisers. We argue that this problem, while generally falling under the umbrella of mechanism design, has several unique features. We propose a general formalism—the *token auction* model—for studying this problem. A key feature of this model is that it acts on a token-by-token basis and lets LLM agents influence generated contents through single dimensional bids.

We first explore a robust auction design approach, in which all we assume is that agent preferences entail partial orders over outcome distributions. We formulate two natural incentive properties, and show that these are equivalent to a monotonicity condition on distribution aggregation. We also show that for such aggregation functions, it is possible to design a second-price auction, despite the absence of bidder valuation functions. We then move to designing concrete aggregation functions by focusing on specific valuation forms based on KL-divergence, a commonly used loss function in LLM. The welfare-maximizing aggregation rules turn out to be the weighted (log-space) convex combination of the target distributions from all participants. We conclude with experimental results in support of the token auction formulation.

[*]Equal contribution [1]Google Research [2]University of Chicago. Correspondence to: Paul Dütting <duetting@google.com>, Vahab Mirrokni <mirrokni@google.com>, Renato Paes Leme <renatoppl@google.com>, Haifeng Xu <haifengustc@google.com>, Song Zuo <szuo@google.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

In the current web ecosystem, auctions are the primary mechanism used to decide which ads (and commercial content more broadly) are displayed to users (Edelman et al., 2007; Varian, 2007). In those, advertisers bid for the right to have their creatives displayed to the user along with organic contents. Many of the web formats such as text, banners, video, apps, ... have their own subtleties which led to the development of new auction tools to handle them. Our goal in this paper is to investigate auction mechanisms to support the emerging format of AI-generated content. More specifically, we explore the use of auctions as a tool for influencing the output of large language models (LLMs).

We consider a situation where a certain space in the web (be a webpage, an UI element of an AI-chatbot, the dialog of a certain character in a video or a game ...) is clearly marked as commercial content and different advertisers can bid to influence the content in that space. Each advertiser has an LLM that can generate content for that space and is willing to pay a certain amount of money for the right to have their content displayed. A simple design is to collect bids from advertisers and let the highest bidder choose whatever content they wish to publish in that space. While simple, this design does not exploit the flexibility of LLMs which is to combine different concepts in a creative way.

Consider this example. In the first we ask an LLM to produce different ads for the fictitious Stingray Resort and the equally fictitious Maui Airlines:

- "*Experience the magic of Hawaii at Stingray Resort, where stunning views, luxurious accommodations, and endless activities await. Book your stay today and create unforgettable memories in the heart of paradise.*"
- "*Fly to Hawaii with Maui Airlines and experience the beauty of the Aloha State. We offer affordable flights to all the major islands, so you can start your Hawaiian vacation sooner. Book your flight today and let the island spirit take over!*"

For that use case, however, the LLM is flexible enough to produce a joint ad for both:

- *"Fly to paradise with Maui Airlines and experience*

the magic of Hawaii at Stingray Resort. Stunning views, luxurious accommodations, and endless activities await. Book your dream vacation today and create unforgettable memories."

One can envision an auction mechanism where both Stingray resort and Maui airlines can submit both LLMs as well as bids, and this will determine their prominence in the final outcome.[1]

1.1. Unique Challenges

LLMs (Brown et al., 2020; Thoppilan et al., 2022; Google et al., 2023) are a new technology with new and unconventional aspects, many of which have direct implications to auction design (e.g., how preferences are represented/expressed). Our goal is to identify some of the key challenges and take a first step in designing mechanisms to address them:

- **Modelling and Expressing Preferences.** Auction theory typically models preferences via value functions that assign a value to each outcome. LLMs, however, are *generative models* which do not attribute values to each example, but instead succinctly encode preferences over outcomes in a stateless neural network model that predicts continuation probabilities.

- **Necessity of Randomization.** LLMs crucially rely on randomization. When forced to output tokens deterministically, LLMs often have a worse performance compared to situations that sample from a distribution (see, e.g., (Holtzman et al., 2019), for a performance comparison of different decoding strategies). Therefore, an auction that aggregates LLM outputs should preferably also output distributions.

- **Technical Alignment.** Auction solutions should be technically aligned with current LLM technology. They should only use information available from current models and should be easy to integrate in the system. Ideally the allocation and payments should be obtained from simple manipulations of the LLM outputs.

- **Computational Efficiency.** LLM models are expensive to query, so the auction computation should not add too much overhead. In particular, auctions should not increase the number of calls to inference the models beyond the minimum necessary.

[1]While our main focus is to create ad creatives that merge content from different advertisers, auction mechanisms for merging LLM outputs could be used in other contexts.

1.2. Our Contributions

The Token Auction Model. Our first contribution is a formalism ("The Token Auction Model") for studying this problem. *Tokens* are the units making up sentences and paragraphs.[2] Examples of tokens include (sub-)words, symbols, numbers, and special tokens indicating the beginning and ending of the text. In particular, any piece of text (potentially incomplete) can be represented as an array of tokens, and any array of tokens also encodes a piece of text.

One salient feature of the state-of-the-art LLMs is that they are stateless, i.e., they maintain no internal memory or state. Instead, they simply map a prefix string to a distribution over the next token. The output is then created in an autoregressive manner. Given an input prompt, the output is generated by repeatedly feeding the current sequence of tokens into the LLM, sampling a continuation token, and appending it to the sequence of tokens.

The *token auction* we propose also operates on a token-by-token basis, and serves to aggregate several LLMs to generate a joint output. We assume the designer has access to algorithmic LLM agents represented by their respective text generation functions (the functions that map a sequence of tokens to a distribution over the next token). In addition, we allow each LLM agent to submit a single dimensional bid. The auction output will be an aggregated distribution together with a payment rule.

This approach may appear counterintuitive at first since an advertiser cares about the generated final text, not the specific choice of words. This seems to suggest a dynamic planning of the generated token sequence. However, existing LLMs do not reason about full pieces of text, nor do they plan ahead; instead, their preferences are expressed as desired distributions over merely the next token. In other terms, we can think of an LLM as a succinct distillation of an agent's complex combinatorial preferences over sequences of tokens into a generative token-by-token model.[3]

The problem of aggregating LLMs forces the designer to understand the preferences of the agents away from the distilled LLM. This appears to be a very difficult problem. Specifically, we believe it is implausible or at least impractical to assume an individual agent can meaningfully manipulate the distribution over tokens at any given stage, to direct the produced text to a more preferred one. Our auction formulation seeks to strike a balance: By truthfully revealing the LLM to the designer, the agent gives the auction mech-

[2]More generally, one can consider tokens forming parts of images (Ramesh et al., 2021; Yu et al., 2022) and videos (Sun et al., 2019). For the purpose of this paper, we will stick with text generation.

[3]See our discussion in Section C, and Propositions C.1 and C.3 for additional support for the stateless approach.

anism a hint as to what their preferred distribution is. The bids, in turn, can be used to tradeoff between agents, and in particular help the designer determine their relative weights.

Simple and Robust Token Auctions. Motivated by the very challenging problem of modelling the agents' preferences for nearby generative models, we aim to design *robust* token auctions. We seek auctions that provide desirable incentive properties, while imposing minimal assumptions on the agents' preferences over distributions.

Specifically, we investigate a model where agents' preferences entail partial orders over distributions. We formulate two desirable incentive properties, which we consider minimal requirements:

- *Payment monotonicity*: Given two different bids by the same agent, a final distribution is closer to the desired distribution if and only if the payment is higher.
- *Consistent aggregation:* If for two different bids of the same agent, the final distribution is closer to the preferred distribution for some bids of the other agents, then it should be so for all bids of the other agents.

We show that any mechanism with these two properties is *strategically equivalent* to a mechanism that satisfies a monotonicity requirement on the distribution aggregation function.

We then investigate whether it is possible to equip such distribution aggregation functions with payment rules that satisfy additional incentive properties. Specifically, we investigate whether such aggregation rules admit an analogue of the *second-price payment rule*. In the single-item second-price (or Vickrey) auction (Vickrey, 1961), the payment corresponds to the critical bid where an agent transitions from losing to winning. To port this notion to our setting, we show that any monotone aggregation rule can be written as a distribution over deterministic allocations from bids to tokens such that there is a critical bid where the allocation transitions from a less preferred to a more preferred token. Such a critical bid becomes then a natural candidate for a payment rule.

A natural analogue of the second-price auction is obtained for a model that only relies on ordinal preferences. The resulting class of auctions is applicable whenever the agent valuations are compatible with the partial order, and provides robust incentives for all of these.

Designing Aggregation Functions. We then move to designing concrete aggregation functions. Our approach is to define welfare objectives inspired by LLM training, and to derive optimal distribution aggregation functions for these welfare notions.

We focus on specific valuation forms based on KL-divergence, a commonly used loss function in LLM. We specifically discuss two natural welfare notions, and show that the corresponding welfare-maximizing aggregation rules are the weighted (log-space) convex combination of the target distributions from all participants.

The linear and log-linear aggregation rules we identify have different pros and cons. Both share the advantage that they are welfare maximizing for the respective welfare notions. The linear rule unlike the log-linear rule is also monotone, and therefore compatible with the robust incentives approach.

Demonstration. We conclude with experiments to support our token auction formulation, obtained by prompt-tuning of a publicly available LLM. We consider a two-advertiser example and the linear and log-linear aggregation rules identified to be optimal for corresponding welfare notions. We show how the combined output varies as a function of $\lambda = {}^{b_1}/_{(b_1+b_2)}$, where b_1 and b_2 are the advertisers' bids. Both approaches lead to meaningful and interpretable text, and smoothly transition from one to another advertiser, with a joint ad produced for intermediate values of λ.

Additional Related Work. Additional related directions include LLM fine-tuning (Wei et al., 2021; Bakker et al., 2022; Ouyang et al., 2022; Bai et al., 2022), in-context learning (Brown et al., 2020; Wei et al., 2022; 2023), mechanism design for public projects (Papadimitriou et al., 2008; Dughmi, 2011; Freeman et al., 2019; Goel et al., 2019), and robust mechanism design (Bergemann and Morris, 2005; 2012; Roughgarden and Talgam-Cohen, 2016; Carroll, 2015; Dütting et al., 2019). See Appendix D for a more detailed discussion.

2. Demonstration

The description of the token auction model as well as the analysis of the auctions designed and their incentives can be found in the appendix.

We implement the aggregation functions proposed in Section C and discuss the examples they produce. Off-the-shelf LLMs respond full text passages. In our case, we need to peek at the internal states of LLMs (the probability distributions over tokens) at each token generation stage. For that reason, we use a custom version of the Google Bard model with a custom inference method that allows us to access the distributions. The aggregation functions are implemented inside the inference method.

Starting from the same base model, we customize it for different agents by prompt-tuning. One key advantage of simulating LLM agents with different prompts is that one does not need to serve multiple LLMs at the same time,

λ	Linear aggregation function q_{KL}	Log-linear aggregation function $\bar{q}_{\mathsf{KL}}$
1	**Alpha Airlines**: Your ticket to paradise.	
0.75	**Alpha Airlines**: Fly to Hawaii and experience the beauty of the islands with aloha. **Beta Resorts**: Stay at our resorts and enjoy the best of Hawaii.	Feel the magic of Hawaii with a flight on **Alpha Airlines**, now offering 20% off all flights when you book with us today!
0.6	**Alpha Airlines** flies you to Hawaii, where you can enjoy a week-long stay at the **Beta resort** for just $1000.	Experience the magic of Hawaii with a flight on **Alpha Airlines**, now offering 20% off all flights when you book with us today!
0.55	**Alpha Airlines** flies you to Hawaii, where you can enjoy a week-long stay at the **Beta resort**.	Escape to the tropical paradise of Hawaii with [**Alpha Airlines**]!
0.5	**Alpha Airlines** flies you to Hawaii, where you can enjoy a beautiful sunset on the beach. Stay 3 nights and get the 4th free at the **Beta Resort**.	Experience the magic of Hawaii with a stay at the luxurious [**Beta Resort**] and a refreshing flight on [**Alpha Airlines**].
0.45	Fly **Alpha Airlines** to sunny Hawaii and enjoy the secluded beaches and private lagoons of the Royal Hawaiian **Beta Resort**.	Experience the magic of Hawaii with a stay at the luxurious [**Beta Resort**] and a special flight offer from [**Alpha Airlines**].
0.4	Fly **Alpha Airlines** to sunny Hawaii and enjoy the first-class treatment that awaits you at **Beta Resort**, all for one low price.	Experience the magic of Hawaii at the [**Beta Resort**], where you'll feel like you're in a tropical paradise.
0.25	Experience the magic of Hawaii at the **Beta Resort**, where the sun shines brighter and the waves crash louder — book your stay today with our exclusive 20% off discount!	Experience the magic of Hawaii at the **Beta Resort**, where you'll be pampered like royalty and surrounded by breathtaking beauty.
0	Hawaii's **Beta Resort**: a paradise where the sun shines brighter, the waves sing sweeter, and the sand feels softer.	

Table 1. Text generation from two aggregation functions with different $\lambda = b_1/(b_1 + b_2)$.

but instead making multiple queries to the served one. Because of their large sizes, serving multiple LLMs can be very costly and practically challenging. As one of the key strengths of LLMs, the flexibility to accomplish various tasks with properly designed prompts sheds light to the possibility of training one universal LLM that can, for example, generate different ads according to agent-specific prompts. That is, the universal advertising LLM, plus an advertiser-specific prompt, behaves like an advertiser-specific LLM through the online in-context few-shot learning.

2.1. Setup

We illustrate our method with a co-marketing example among two agents; each of them would like to advertise for their brands, "Alpha Airlines" and "Beta Resort", regarding a shared topic "Hawaii". We choose fictitious brands to avoid the model directly using any existing ads. We use the brand names "Alpha" and "Beta" that do not have strong meanings on their own to minimize any potential hallucination, as we are using a common purposed LLM that is not optimized for our task. Each agent is given the following prompt:

> "You are an expert of writing texts that naturally combines two ads together. Your choice of words and sentences is full of artistic flair.
>
> Write a one-sentence ad for _____."

Agent A uses *"a flight to Hawaii using [**Alpha Airlines**]"* to fill the blank, while agent B uses *"a vacation in Hawaii at the [**Beta Resort**]"*. The first two sentences in the prompt aim to improve the quality of the ad generation through *assigning roles* (see, for example, (Wu et al., 2023)).

A natural question is whether the proposed method can adjust the combining strategy according to the context. If two agents are advertising for competing brands, will it avoid advertising for both in the same creative? We also explore such a competing setup with "Beta Resort" and "Gamma Hotel" using the same design.

Two Bids as One Parameter. Since in both the linear aggregation rule q_{KL} and the log-linear aggregation rule $\bar{q}_{\mathsf{KL}}$, there is only one degree of freedom, we parameterize the response by $\lambda = {}^{b_1}/_{(b_1+b_2)}$.

2.2. Results

The results for the co-marketing example are listed in Table 1, where from top to bottom, the value of λ decreases from 1 to 0. As we can see for both aggregation functions, the generated texts roughly follow the pattern of "only Alpha Airlines" $\rightarrow$ "both Alpha Airlines and Beta Resort" $\rightarrow$ "only Beta Resort" when λ goes from 1 to 0. This is expected, as λ going from 1 to 0 corresponds to b_2 increasing from 0 to ∞ with b_1 fixed (or b_1 decreasing from ∞ to 0 with b_2 fixed). The thresholds of pattern changes are 0.75 and 0.4 for the linear aggregation, and 0.5 and 0.45 for the log-linear aggregation. We emphasize that the example is generated with a general purposed LLM, and it is reasonable to believe that the performance can be improved with proper fine-tuning for the specific task at hand.

References

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Chris Olah, Benjamin Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback. *CoRR*, abs/2204.05862, 2022. URL https://doi.org/10.48550/arXiv.2204.05862.

Michiel A. Bakker, Martin J. Chadwick, Hannah Sheahan, Michael Henry Tessler, Lucy Campbell-Gillingham, Jan Balaguer, Nat McAleese, Amelia Glaese, John Aslanides, Matt M. Botvinick, and Christopher Summerfield. Fine-tuning language models to find agreement among humans with diverse preferences. In *NeurIPS 2022*, pages 38176–38189, 2022.

Dirk Bergemann and Stephen Morris. Robust mechanism design. *Econometrica*, pages 1771–1813, 2005.

Dirk Bergemann and Stephen Morris. *Robust mechanism design: The role of private information and higher order beliefs*, volume 2. World Scientific, 2012.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. pages 1877–1901, 2020.

Jeremy Bulow and John Roberts. The simple economics of optimal auctions. *Journal of Political Economy*, 97(5): 1060–1090, 1989.

Gabriel Carroll. Robustness and linear contracts. *American Economic Review*, 105(2):536–563, 2015.

Shaddin Dughmi. A truthful randomized mechanism for combinatorial public projects via convex optimization. In *EC 2011*, pages 263–272, 2011.

Paul Dütting, Tim Roughgarden, and Inbal Talgam-Cohen. Simple versus optimal contracts. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 369–387, 2019.

Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007.

Rupert Freeman, David M Pennock, Dominik Peters, and Jennifer Wortman Vaughan. Truthful aggregation of budget proposals. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 751–752, 2019.

Ashish Goel, Anilesh K Krishnaswamy, Sukolsak Sakshuwong, and Tanja Aitamurto. Knapsack voting for participatory budgeting. *ACM Transactions on Economics and Computation*, 7(2), 2019.

Google, Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. Palm 2 technical report, 2023.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin

Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2019.

Roger B Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *NeurIPS 2022*, pages 27730–27744, 2022.

Christos Papadimitriou, Michael Schapira, and Yaron Singer. On the hardness of being truthful. In *FOCS 2008*, pages 250–259, 2008.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML 2021*, pages 8821–8831, 2021.

Tim Roughgarden and Inbal Talgam-Cohen. Optimal and robust mechanism design with interdependent values. *ACM Transactions on Economics and Computation*, 4 (3):1–34, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *ICCV 2019*, pages 7464–7473, 2019.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Kathleen S. Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Agüera-Arcas, Claire Cui, Marian Croak, Ed H. Chi, and Quoc Le. Lamda: Language models for dialog applications. *CoRR*, abs/2201.08239, 2022. URL https://arxiv.org/abs/2201.08239.

Hal R. Varian. Position auctions. *International Journal of Industrial Organization*, 25 (6):1163–1178, 2007.

William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1): 8–37, 1961.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2021.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS 2022*, pages 24824–24837, 2022.

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, and Tengyu Ma. Larger language models do in-context learning differently. *CoRR*, abs/2303.03846, 2023. URL https://doi.org/10.48550/arXiv.2303.03846.

Ning Wu, Ming Gong, Linjun Shou, Shining Liang, and Daxin Jiang. Large language models are diverse role-players for summarization evaluation. *arXiv preprint arXiv:2303.15078*, 2023.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation. *Transactions on Machine Learning Research*, 2022.

A. Preliminaries

In this section, we first provide an abstraction of typical generative models. For concreteness we adopt a terminology that suits the important LLM use case where the creative is text. We then introduce the basic formalism of the mechanism design problem we study.

A.1. Abstraction of Large Language Models

Large language models (LLMs) (Brown et al., 2020; Thoppilan et al., 2022; Google et al., 2023) can be abstracted as functions mapping from a partial sentence to the distribution of the next *token* that extends the partial sentence.

Formally, let T be the set of tokens and $\Delta(T)$ be the set of distributions over T. Let $T^* = T \cup T^2 \cup \cdots \cup T^K$ denote the set of sequences of tokens, where K is the maximum sequence length that the LLM can handle. Each LLM is modeled as a function $f : T^* \to \Delta(T)$ that maps any sequence of tokens to a distribution over the next token.

Autoregressive Text Generation. A *prompt* is an initial set of tokens $s_0 \in T^*$ provided with instructions of what text to generate. An LLM produces a text in response to the prompt by sampling a token $\tau_1 \sim f(s_0)$ and constructing $s_1 = s_0 \oplus \tau_1$ (where $\oplus$ is the operation to append a token to an array). We then repeat the process of $\tau_k \sim f(s_k)$ and $s_k = s_{k-1} \oplus \tau_k$ until a special *end-of-sentence* token is sampled. If at some point the sequence of tokens becomes too long (larger than K) we trim s_k to its length-K suffix.

We remark that LLMs are stateless by design: They keep no internal memory (other than the sequence of tokens produced so far) and each token is sampled independently.

Training of LLMs. An LLM f is parameterized by a neural network structure M and a set of weights W. The weights are often obtained by three stages of optimization (see first three rows in Table 2). The initial stage is very computationally intensive but task independent. Subsequent stages are less costly and their goal is to adapt the general purposed model obtained in the first stage to more specific tasks. In each of the stages, we minimize a different loss function over a different dataset. The details of the training process are not particularly relevant for our discussion, but we add a more detailed discussion in Section C.1. We will note, however, that some of the mechanisms we discuss for combining the inputs of different LLMs resemble the functional form used in the reinforcement learning and fine-tuning steps.

Training/Learning stages	Data	Cost	Goal
Pre-training [4]	General texts from web, books, etc	Very high	A common baseline shared across downstream tasks
Instruction fine-tuning [5]	Task specific data	Medium	Optimize the behavior for specific tasks
RLHF [6]	Human evaluations	Medium	Security control, reducing harmful behavior, etc
In-context few-shot learning	Carefully designed prompts as inputs	Very low	Effectively influence the behavior in real-time

Table 2. Common training stages of LLMs.

A.2. Token Auctions for LLMs

We now formalize the mechanism design problem of combining the outputs of different LLM-represented algorithmic agents. As discussed in the introduction, we will design an auction to act on the token-by-token generation stage. Our goal is to keep the auction technically aligned with the state-of-the-art LLM systems.

Robust Modeling of LLM Agents' Preferences. One major challenge in designing mechanisms for LLM agents is that they are represented as distributions, and it is generally difficult to compare LLM agents' "utilities" among distributions. To illustrate, suppose an LLM agent's preferred distribution over two tokens is $p = (0.6, 0.4)$, and consider two possible generated distribution outcomes: $q_1 = (0.5, 0.5)$ and $q_2 = (0.8, 0.2)$. Between q_1 and q_2, it is unclear which one this LLM agent would prefer since while q_2 appears more distant from p than q_1, it has a higher probability on the first token which appears more preferably by the LLM's distribution p.

Despite this incomparability between q_1 and q_2, it does appear "obvious" that q_2 would be less preferred by the LLM than $q_3 = (0.7, 0.3)$. This is because q_3 deviates from p along the same directions as q_2 for each entry (i.e., both increase or both decrease), but deviates less in terms of the absolute value of deviation.

The above observation illustrates that while it is difficult to model LLM agents' complete preferences over all the generated

[4] Brown et al. (2020); Google et al. (2023).
[5] Wei et al. (2021).
[6] Ouyang et al. (2022).

distributions, it seems plausible to assume a certain *partial order* over the distributions. This motivates us to consider *robust* modeling of LLM agents' preferences, with the following notion of *obvious preference*.

Definition A.1 (Obvious Preferences over Distributions)**.** Consider any LLM agent i with preferred distribution p_i, and any two aggregation distribution $q, q' \in \Delta(T)$. We say q is (weakly) *obviously preferred* over q' by agent i — or formally, $q \succeq_i q'$ — if

$$\forall t \in T, \qquad |q(t) - p_i(t)| \leq |q'(t) - p_i(t)| \tag{1}$$

$$\text{and} \quad (q(t) - p_i(t))(q'(t) - p_i(t)) \geq 0. \tag{2}$$

Moreover, if $q \neq q'$, q is strictly preferred over q' by i, i.e., $q \succ_i q'$.

In other words, q is preferred by i over q' when (1) the deviation of q from p_i is smaller than the deviation of q' from p_i for every entry; and (2) these deviations are along the same direction for every entry. Note that Definition A.1 only specified a partial ordering among aggregated distributions. Thus it is possible that two distributions are not comparable, i.e., $q \not\succeq_i q'$ and $q' \not\succeq_i q$.

Token Auctions. Our goal is to design simple, practical auction mechanisms that work well under minimal assumptions about the agents' private preferences. Specifically, we seek to design *token auction mechanisms* $\mathcal{M} = \langle q, z \rangle$, where q is a distribution aggregation function and z is a payment function. A token auction mechanism operates on a token-by-token basis, and lets n algorithmic LLM agents influence the output distribution and payments through scalar bids. We denote the vector of bids by $\boldsymbol{b} = (b_1, \ldots, b_n) \in \mathbb{Q}_+^n$. We assume that the initial prompt $s_0 \in T^*$, and the text aggregation functions $f_1, \ldots, f_n$ of the n LLM agents are publicly known.

Distribution Aggregation Function. This is the first ingredient to a token auction mechanism. A distribution aggregation function q takes as input a vector of bids $\boldsymbol{b} \in \mathbb{Q}_+^n$ and n distributions $\boldsymbol{p} \in \Delta(T)^n$ and maps these to a distribution over tokens:

$$\text{aggregation function:} \quad q : \mathbb{Q}_+^n \times \Delta(T)^n \to \Delta(T).$$

For fixed bids, a distribution aggregation function can be used in the same way as a text aggregation function. Namely, starting from the initial prompt $s_0 \in T^*$, we can repeatedly sample τ_k from distribution $q_k = q((b_1, \ldots, b_n), (f_1(s_{k-1}), \ldots, f_n(s_{k-1})))$ for each $k \geq 1$ to generate $s_k = s_{k-1} \oplus \tau_k$. Note the alignment with LLMs, which already produce the distributions $f_i(s_{k-1})$ for $i \in [n]$. No additional calls to the LLMs are needed.

Payment Function. In addition to the distribution aggregation function, we seek to design payment functions. Here, we want to operate on a token-by-token-basis and seek a stage independent design. Formally, for each agent i, we aim to define a

$$\text{pricing function:} \quad \zeta_i : \mathbb{Q}_+^n \times \Delta(T)^n \times T \to \mathbb{R},$$

with the interpretation that for bids $\boldsymbol{b} \in \mathbb{Q}_+^n$, distributions $\boldsymbol{p} \in \Delta(T)^n$, and token $t \sim q(\boldsymbol{b}, \boldsymbol{p})$, the payment from agent i is $\zeta_i(\boldsymbol{b}, \boldsymbol{p}, t)$. These pricing functions naturally lead to expected payments by taking expectations over tokens. Namely, for each agent i, we define

$$\text{payment function:} \quad z_i : \mathbb{Q}_+^n \times \Delta(T)^n \to \mathbb{R},$$

as the function that takes as input a vector of bids $\boldsymbol{b} \in \mathbb{Q}_+^n$ and distributions $\boldsymbol{p} \in \Delta(T)^n$, and maps these to $z_i(\boldsymbol{b}, \boldsymbol{p}) = \mathbb{E}_{t \sim q(\boldsymbol{b}, \boldsymbol{p})}[\zeta_i(\boldsymbol{b}, \boldsymbol{p}, t)]$.

Discussion. We believe that token auctions and our assumptions offer the right level of abstraction for reasoning about the strategic aspects of aggregating LLMs. This is because it seems impractical (if not impossible) to fully express an agent's preferences over all possible generated creatives. The most plausible way to do so at the current state of the art is perhaps to represent each agent as an LLM. Indeed, by design, current LLMs naturally distill agent preferences over texts to stateless distributions over tokens. Therefore, if agents' are represented as LLMs, it seems natural to auction tokens based on agents' token preferences expressed by the LLMs.

At the same time, the detailed functioning of LLMs remains rather opaque, and it seems implausible that agents could meaningfully misreport the outcome distributions of their LLMs in order to achieve a more desirable aggregated output.

Our auction formulation offers a middle ground. We assume the designer has access to the LLMs, but let the agents influence the aggregation process through a single dimensional bid.

B. Incentives in Token Auctions

In this section, we examine the strategic properties of token auctions. Our goal is robust incentive properties that rely on as few assumptions about the agents' preferences as possible. We first formulate two natural properties that any reasonable mechanism should satisfy, and show that they are equivalent to a monotonicity requirement on the distribution aggregation function. We then show that for any such monotone distribution aggregation function, it is possible to define a natural second-price payment rule.

B.1. Desirable Incentive Properties

We begin by formulating two conditions that any reasonable token auction mechanisms should satisfy. The first is a *monotonicity* condition on the payment function. It requires that agents' pay increases if and only if they obtain obviously better distributions.

Definition B.1 (Payment Monotonicity)**.** Mechanism $\mathcal{M} = \langle q, z \rangle$ satisfies *payment monotonicity*, if for all $\boldsymbol{p}$, $\boldsymbol{b}_{-i}, b_i, b_i'$ we have

$$z_i(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) \geq z_i(b_i', \boldsymbol{b}_{-i}, \boldsymbol{p}) \iff q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) \succeq_i q(b_i', \boldsymbol{b}_{-i}, \boldsymbol{p}).$$

It is a natural incentive constraint because if the payment function is not monotone, then bidders will naturally manipulate their bids in order to induce better distribution with lower payment.

The second incentive constraint is about the consistency of the aggregation function. Intuitively, whenever two bids lead to two aggregated distributions with an obvious order, this order should be consistent in the sense that it is not affected by others' bids.

Definition B.2 (Consistent Aggregation)**.** The distribution aggregation function $q(\boldsymbol{b}, \boldsymbol{p})$ is said to be *consistent* if it admits consistent ordering across all $\boldsymbol{b}_{-i}$. Formally, if $q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) \succ_i q(b_i', \boldsymbol{b}_{-i}, \boldsymbol{p})$ for some $\boldsymbol{b}_{-i}$, then for all $\boldsymbol{b}'_{-i}$, $q(b_i, \boldsymbol{b}'_{-i}, \boldsymbol{p}) \succeq_i q(b_i', \boldsymbol{b}'_{-i}, \boldsymbol{p})$.

Similar to payment monotonicity, this requirement of consistent aggregation is imposed to avoid bidders' concerns that the same bid can lead to obviously better or worse distributions, depending on the opponents' bids. To rule out such behavior, consistency requires that whenever bids b_i, b_i' are such that b_i is strictly preferred over b_i' for some opposing bids $\boldsymbol{b}_{-i}$, then it better be that b_i is weakly preferred over b_i' for all opposing bids $\boldsymbol{b}'_{-i}$.

These two properties are quite common in the mechanism design literature. For instance, in single-time auction design, allocation consistency is a necessary condition for incentive compatible mechanisms (Myerson, 1981), and it also induces a monotone payment function.

B.2. Monotone Aggregation Functions

Next we show a "revelation principle" type of result, stating that if one is interested in mechanisms satisfying the desirable incentive properties stated above (Definition B.1 and Definition B.2), then one can without loss of generality focus on *monotone aggregation functions* as captured in the following definition.

Definition B.3 (Monotone Aggregation Function)**.** The distribution aggregation function $q(\boldsymbol{b}, \boldsymbol{p})$ is called *monotone* if any higher bid from any agent i leads to an obviously more preferred aggregated distribution for i. Formally, for all $\boldsymbol{p}$, $\boldsymbol{b}_{-i}$ and $b_i \geq b_i'$:

$$q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) \succeq_i q(b_i', \boldsymbol{b}_{-i}, \boldsymbol{p}).$$

We are now ready to state our main finding in this subsection with the following definition of *strategic equivalence* from one mechanism $\mathcal{M}$ to another $\tilde{\mathcal{M}}$. In words, the aggregated distribution and all agents' payments will be the same under mechanism $\mathcal{M}$ and $\tilde{\mathcal{M}}$ after each agent i applies some strategy mapping π_i. Formally,

Definition B.4 (Strategic Equivalence)**.** A mechanism $\mathcal{M} = \langle q, z \rangle$ is *strategically equivalent* to another mechanism $\tilde{\mathcal{M}} = \langle \tilde{q}, \tilde{z} \rangle$, if $\forall \boldsymbol{p} \in \Delta(T)^n$, there exists a profile π of strategy mappings with $\pi_i : \mathbb{Q}_+ \to \mathbb{Q}_+$ for every agent i (i.e., $\pi(\boldsymbol{b}) = (\pi_1(b_1), \ldots, \pi_n(b_n))$), such that $\forall \boldsymbol{b} \in \mathbb{Q}_+^n,\ q(\boldsymbol{b}, \boldsymbol{p}) = \tilde{q}(\pi(\boldsymbol{b}), \boldsymbol{p})$ and $z(\boldsymbol{b}, \boldsymbol{p}) = \tilde{z}(\pi(\boldsymbol{b}), \boldsymbol{p})$.

Theorem B.5 (Revelation Principle)**.** *Any mechanism $\mathcal{M} = \langle q, z \rangle$ with a consistent distribution aggregation function q and a monotone payment function z is strategically equivalent to a mechanism $\tilde{\mathcal{M}} = \langle \tilde{q}, \tilde{z} \rangle$ which has a monotone distribution aggregation function $\tilde{q}$ and a monotone payment function $\tilde{z}$.*

Theorem B.5 can be viewed as a revelation principle in the sense that it simplifies the design choice of aggregation functions. Monotone aggregation functions are a strict subset of consistent aggregation functions since monotonicity directly implies a total order over possible aggregated distributions $Q(\boldsymbol{b}_{-i}, \boldsymbol{p}) = \{q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) : b_i \in \mathbb{Q}_+\}$, with the order naturally determined by the real-numbers' order on i's bid b_i hence this order will be consistent across different $\boldsymbol{b}_{-i}$ and $\boldsymbol{p}$. In this sense, one might think that consistency — which does not impose any restriction when $q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p})$ is not strictly preferred over $q(b'_i, \boldsymbol{b}_{-i}, \boldsymbol{p})$ — might be a significantly weaker requirement on aggregation functions than monotonicity which requires a total and consistent order. Theorem B.5 shows that this is not the case — they are essentially the same as long as the natural incentive requirement of payment monotonicity is also imposed.

The proof of Theorem B.5 hinges on the following two lemmas, Lemma B.7 and Lemma B.6. Together these two lemmas imply the existence of a strategy mapping, under which the resulting aggregation function becomes monotone. The proof of the theorem is completed by applying the same mapping to the payment function, and noting that this ensures payment monotonicity. We defer the formal proofs of these results to Appendix E.

Lemma B.6. *Consider any consistent aggregation function q. Suppose $\succeq_i$ defines a total order over the aggregation set $Q(\boldsymbol{b}_{-i}, \boldsymbol{p}) = \{q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) : b_i \in \mathbb{Q}_+\}$ induced by agent i's bid for any fixed $\boldsymbol{b}_{-i}$ and $\boldsymbol{p}$, then there exist a profile π of strategy mappings such that $q(\boldsymbol{b}, \boldsymbol{p}) = \tilde{q}(\pi(\boldsymbol{b}), \boldsymbol{p})$ for some monotone aggregation function $\tilde{q}(\cdot, \boldsymbol{p})$.*

Lemma B.7. *For any distribution aggregation function q, there exists a payment function z such that mechanism $\mathcal{M} = \langle q, z \rangle$ is payment-monotone if and only if $\succeq_i$ establishes a total order over $Q(\boldsymbol{b}_{-i}, \boldsymbol{p}) = \{q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) : b_i \in \mathbb{Q}_+\}$ for any fixed $\boldsymbol{b}_{-i}$ and $\boldsymbol{p}$.*

We conclude this subsection by giving two natural examples used in today's machine learning practice: an example of a monotone aggregation function, and a non-monotone one.

[Linear Aggregation] Consider $q_{\mathsf{KL}}(\boldsymbol{b}, \boldsymbol{p})$ defined as

$$q_{\mathsf{KL}} = \frac{1}{B} \sum_{i \in [n]} b_i \cdot p_i, \text{ where } B = \sum_{i \in [n]} b_i.$$

It is easy to verify that this is a monotone aggregation function.

[Log-linear Aggregation] Consider the aggregation function $\bar{q}_{\mathsf{KL}}(\boldsymbol{b}, \boldsymbol{p})$ defined by the following equations:

$$\forall t \in T, \ \ln \bar{q}_{\mathsf{KL}}(t) = \frac{1}{B} \sum_{i \in [n]} b_i \cdot \ln p_i(t) - C,$$

where $B = \sum_{i \in [n]} b_i$ and $C = \ln \sum_{t \in T} e^{\frac{1}{B} \sum_{i \in [n]} b_i \cdot \ln p_i(t)}$.

The following two-agent example shows that $\bar{q}_{\mathsf{KL}}$ is not monotone: $p_1 = (.5, .4, .1)$ and $p_2 = (.5, .1, .4)$. When $b_1 = b_2$, $\bar{q}_{\mathsf{KL}} = (\sqrt{.25}, \sqrt{.04}, \sqrt{.04})/.9 = (5/9, 2/9, 2/9)$. Fix $b_2 = 1$, either $b_1 \to 0$ or $b_1 \to \infty$, $\bar{q}_{\mathsf{KL}}(t_1) = .5 < 5/9$. Hence $\bar{q}_{\mathsf{KL}}$ must not be monotone.

B.3. Second Price Payment Rules

Next we explore whether for monotone aggregation functions we can create a pricing rule with a "second-price" semantic inspired by the Vickrey auction notion of "minimum-bid-to-win". In the Vickrey auction, the payment corresponds to the critical bid where an agent transitions from losing to winning. To port this notion to our setting, we will show in Theorem B.10 that any monotone aggregation rule can be written as a distribution over deterministic allocations from bids to tokens such that there is a critical bid where the allocation transitions from a less preferred to a more preferred token. Such a critical bid becomes then a natural candidate for the payment. We will show that besides being payment monotone it has other desirable properties, such as a Myerson-like characterization (Myerson, 1981) in terms of the total variation distance between the preferred distribution and the outcome. To define our payment rule, we first discuss the notion of stable sampling.

B.3.1. Stable Sampling

We analyze a monotone distribution aggregation function $q(\boldsymbol{b}, \boldsymbol{p})$ from the perspective of a single agent where the distributions $\boldsymbol{p}$ and the bids of other agents $\boldsymbol{b}_{-i}$ are fixed. To simplify the notation, we refer to $q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p})$ as $q(b_i)$.

We define an implementation of $q(\cdot)$ as a function σ that maps b_i and an exogenous random variable $r \sim \mathcal{R}$ (independent of $\boldsymbol{b}$ and $\boldsymbol{p}$) to a token $t \in T$. When r is fixed, σ is fully deterministic. We say that $\sigma : \mathbb{Q}_+ \times \mathcal{R} \to T$ is a valid implementation of $q(\cdot)$ if:

$$\Pr_{r \sim R}[\sigma(b_i, r) = t] = q_t(b_i), \forall t \in T.$$

Next we define what it means for an implementation to be a stable sampling. It is useful to split the token set T into $T_+ = \{t \in T : q_t(0) \leq (p_i)_t\}$ and $T_- = \{t \in T : q_t(0) > (p_i)_t\}$ corresponding to undersampled (T_+) and oversampled (T_-) tokens. The monotonicity of aggregation function q is equivalent to the monotonicity of $q_t(b_i)$ as formalized in the following lemma (proof in Appendix F).

Lemma B.8. *A distribution aggregation function q is monotone, if and only if for every agent i and $b_i \in \mathbb{R}_+$,*

1. *$\forall t \in T_+$, $q_t(b_i) \leq (p_i)_t$ and q_t weakly increases;*
2. *$\forall t \in T_-$, $q_t(b_i) \geq (p_i)_t$ and q_t weakly decreases.*

We can now define the key notion of stable sampling, and state and prove our main result in this subsection.

Definition B.9 (Stable Sampling)**.** Let $q_i(b_i)$ be an aggregation function obtained by fixing $\boldsymbol{b}_{-i}$ and $\boldsymbol{p}$, and let T_+ and T_- be the sets of undersampled and oversampled tokens for agent i. Then we say that an implementation σ is stable with respect to aggregation function q if for any $r \in \mathcal{R}$ there are two tokens $u_r \in T_+$ and $o_r \in T_-$ and a threshold $\theta_r \in \mathbb{Q}_+ \cup \{\infty\}$ such that:

$$\sigma(b_i, r) = o_r, \text{ if } b_i < \theta_r; \text{ and } \sigma(b_i, r) = u_r, \text{ if } b_i \geq \theta_r.$$

Here is an example of stable sampling for a single undersampled token u and a single overampled token o. Sample r uniformly in $[0, 1]$. If $q_o(b_i) < r$ assign o; assign u otherwise. Note that this implementation is: (i) deterministic for fixed r; (ii) monotone in the bid; (iii) matches the probabilities of q in expectation. The following theorem generalizes this to any number of tokens (proof in Appendix F).

Theorem B.10. *Given a monotone distribution aggregation function q then for any agent i and fixed $\boldsymbol{b}_{-i}$ and $\boldsymbol{p}$ there always exists a stable implementation σ of q.*

B.3.2. Second Price via Stable Sampling

A stable implementation of a monotone aggregation rule suggests a natural pricing rule: For any given randomness r, if the oversampled token o_r is sampled, the agent pays zero as it is the same token that would have been sampled if their bid was zero. If the agent's bid was high enough to move from the oversampled to the undersampled token u_r, then the agent pays the critical bid θ_r.

It has the property that the payment is proportional to the extent to which it shifts the distribution $q(b_i)$ towards the desired distribution p_i. Interestingly, the expected payment does not depend on the actual implementation chosen. Moreover, the expected payment corresponds to the standard Myersonian payment formula where the distribution is replaced by the total variation distance between the agent's preferred distribution p_i and the implemented distribution $q(b_i)$. We again omit the terms $\boldsymbol{b}_{-i}, \boldsymbol{p}$ since they are fixed in each context. If a token $t \in T_-$ is sampled, the payment is naturally $\zeta_i(b_i, t) = 0$. For a token $t \in T_+$ we have:

$$\begin{aligned}
\zeta_i(b_i, t) q_t(b_i) &= \mathbb{E}_r[\theta_r \cdot \mathbf{1}\{\sigma(b_i, r) = t\}] \\
&= \mathbb{E}_r \int_0^{b_i} \left[\mathbf{1}\{\sigma(b_i, r) = t\} - \mathbf{1}\{\sigma(b', r) = t\}\right] \mathrm{d}b' \\
&= \int_0^{b_i} [q_t(b_i) - q_t(b')] \mathrm{d}b'.
\end{aligned}$$

Hence:

$$\begin{aligned}
z_i(b_i) = \sum_t \zeta_i(b_i, t) q_t(b_i) &= \int_0^{b_i} \left[\sum_{t \in T_+} q_t(b_i) - \sum_{t \in T_+} q_t(b')\right] \mathrm{d}b' \\
&= \frac{1}{2} \int_0^{b_i} \left[\|q(b_i) - p_i\|_1 - \|q(b') - p_i\|_1\right] \mathrm{d}b'.
\end{aligned}$$

Counterfactuals. The practical advantage of a stable sampling implementation coupled with a second price rule is to offer advertisers a description where it is clear that they only pay if they can improve the outcome. Moreover, advertisers can more easily evaluate counterfactuals with a fixed r, where they can only produce one of two outcomes on each token that can easily be compared.

Universally Stable Sampling. We chose to define stable sampling as a single-agent algorithm with fixed $\boldsymbol{b}_{-i}$. One may define a universal stable implementation as $\sigma^{\mathsf{univ}} : \mathbb{Q}^n \times \mathcal{R} \to T$ such that

$$\forall \boldsymbol{b} \in \mathbb{Q}_+^n, t \in T, \ \Pr_{r \sim R}[\sigma^{\mathsf{univ}}(\boldsymbol{b}, r) = t] = q_t,$$

and for any i and $\boldsymbol{b}_{-i}$, $\sigma^{\mathsf{univ}}(\cdot, \boldsymbol{b}_{-i}, r)$ is stable. In Appendix G, we provide counter-examples where such σ^{univ} do not always exist.

C. Design of Aggregation Functions

In the previous section, we discussed payment schemes and incentive properties assuming we have an aggregation function. Here we investigate the design of aggregation functions. We adopt two guiding principles: (1) We first define a welfare function to model the overall satisfaction of the agents with the final distribution q, weighted by their bids b_i. The welfare function has the form:

$$\textsc{Wel}(\boldsymbol{p}, \boldsymbol{b}, q) = \sum_i b_i \rho(p_i, q),$$

where $\rho : \Delta(T) \times \Delta(T) \to \mathbb{R}$ indicates how close the distribution q is to the preferred p_i. (2) The second is to define the closeness function ρ based on the typical loss functions in LLM training.

C.1. Review of LLM Training

So far we assume that an LLM $f : T^* \to \Delta(T)$ is already trained. In order to discuss the training process, it is useful to recall that an LLM is a neural network parameterized by a vector of weights $W \in \mathbb{R}^N$ in a very high dimensional space. To discuss training, it is useful to think of f as a function of both input and weights:

$$f : T^* \times \mathbb{R}^N \to \Delta(T).$$

We represent the second argument by a superscript W. Training is to optimize W such that $f^W(\cdot)$ minimizes a certain loss function over a dataset. A dataset is a sequence of pairs (x_i, y_i) with $x_i \in T^*$ (input sequence) and $y_i \in T$ (label), and a loss function is a function $\ell : T \times \Delta(T) \to \mathbb{R}$. A network typically seeks to minimize:

$$\min_W \sum_i \ell(y_i, f^W(x_i)).$$

A widely used loss function in the first stage is the KL-divergence:

$$\ell_{\mathsf{KL}}(y, x) = -\ln[f^W(y|x)],$$

where we use the notation $f(y|x)$ to represent the probability that a token $y \in T$ is sampled from $f(x) \in \Delta(T)$.

It is useful to think of this problem in the limit where the size of the dataset grows to infinity and it can be effectively thought of as a full-support distribution over $T^* \times T$. In that setting, we can represent the dataset as a distribution $\mu \in \Delta(T^* \times T)$ over input-label pairs $(x, y) \in T^* \times T$. We will write $\mu(x) = \sum_y \mu(x, y)$ to denote the marginal distribution on x and $\mu(\cdot|x)$ to denote the conditional distribution of labels given an input x. Then

$$\min_W \mathcal{L}_{\mathsf{KL}}^{\mu}(f^W), \quad \mathcal{L}_{\mathsf{KL}}^{\mu}(f) := \sum_{x \in T^*} \mu(x) \cdot D_{\mathsf{KL}}(\mu(\cdot|x) \| f(x)), \tag{KL}$$

where $D_{\mathsf{KL}}(p \| q) = \sum_t p(t) \ln \frac{p(t)}{q(t)}$ is the KL-divergence.

LLMs are typically trained through a successive refinement of weights: $W^{\mathsf{pre}} \to W^{\mathsf{SFT}} \to W^{\mathsf{RL}}$. In pre-training we compute W^{pre} by solving problem (KL) on a generic dataset μ^{pre} via stochastic gradient descent. In the second stage,

we initialize the weights as $W = W^{\mathsf{pre}}$ and run stochastic gradient descent to solve the same problem (KL) on a more specialized dataset μ^{SFT}. In other words, we solve the same problem on two different datasets.

The problem in the RLHF stage is different. The dataset only contains x (still represented by μ) and for any y, we have a function $r(x, y)$ giving the reward of mapping x to y. Then W^{RL} is obtained by maximizing reward while minimizing the distance to the function f^{SFT} from previous stages (the PPO algorithm (Schulman et al., 2017; Ouyang et al., 2022)):

$$\max_W \mathcal{L}^{\mu,r}_{\mathsf{RL}}(f^W), \tag{RL}$$
$$\mathcal{L}^{\mu,r}_{\mathsf{RL}}(f) := \sum_{x \in T^*} \mu(x) \Big[\sum_y r(x,y) f(y|x) - \beta D_{\mathsf{KL}}(f(x) \| f^{\mathsf{SFT}}(x)) \Big].$$

KL-divergence and Entropy. For the next propositions it is useful to recall that the entropy of a distribution $p \in \Delta(T)$ is $H(p) = -\sum_{t \in T} p(t) \ln p(t)$. Given two distributions $p, q \in \Delta(T)$, the cross entropy of q relative to p is $H(p, q) = -\sum_{t \in T} p(t) \ln q(t)$. Hence we can write $D_{\mathsf{KL}}(p \| q) = H(p, q) - H(p)$. We will also use Gibbs' inequality $H(p) \leq H(p, q), \forall p, q \in \Delta(T)$.

C.2. KL-inspired Aggregation

The first aggregation method will be based on the (KL) program. When trying to aggregate LLMs f_i according to bids b_i, we will design a function q that mimics the outcome of the following thought experiment. We will imagine that each LLM was obtained by solving the (KL) on a dataset represented by μ_i, where the marginal over inputs are the same $\mu_i(x) = \mu(x), \forall i$ but potentially differ on the marginals on the labels $\mu_i(y|x)$. In this thought experiment, we will combine their LLMs by re-training an LLM on the combined labels weighted by the bids. In other words, we will solve the problem:

$$\min_W \ \mathcal{L}^{\bar{\mu}}_{\mathsf{KL}}(f^W), \qquad \text{where } \bar{\mu} = \frac{\sum_i b_i \mu_i}{\sum_i b_i}. \tag{3}$$

The next proposition morally says that we can obtain a solution to the (KL) problem on the aggregated dataset (3) by combining its solutions on individual datasets. The proof appears in Appendix H.

Proposition C.1. *Consider datasets μ_i such that $\mu_i(x) = \mu(x), \forall i, x$ and let $\bar{\mu}$ be their weighted average. Let f_i be the minimizer of $\mathcal{L}^{\mu_i}_{\mathsf{KL}}$ and f^* be the minimizer of $\mathcal{L}^{\bar{\mu}}_{\mathsf{KL}}$, the loss on the aggregated dataset. Then f^* is the solution to:*

$$\min_f \sum_x \mu(x) \sum_i b_i D_{\mathsf{KL}}(f_i(x) \| f(x)). \tag{4}$$

Proposition C.1 motivates the following welfare function:

$$\textsc{Wel}_{\mathsf{KL}} = \sum_{i \in [n]} b_i \cdot \rho_{\mathsf{KL}}(p_i, q) = -\sum_{i \in [n]} \sum_{t \in T} b_i \cdot p_i(t) \cdot \ln \frac{p_i(t)}{q(t)}.$$

Now, we characterize the aggregation function that optimizes $\textsc{Wel}_{\mathsf{KL}}$. The proof in Appendix H uses Gibb's inequality.

Lemma C.2. *The efficient aggregation function that maximizes $\textsc{Wel}_{\mathsf{KL}}$ is the linear combination of p_i:*

$$\forall t \in T, \ q_{\mathsf{KL}}(t) = \frac{\sum_i b_i \cdot p_i(t)}{\sum_i b_i}.$$

Besides being monotone and aligned with how LLMs are trained, this aggregation function has the advantage that we only need to call a single LLM to sample each token. We can choose an index i proportionally to the bids and then sample a token from the i-th LLM. To compute second price payments, however, we still need to query the LLMs for all agents.

C.3. RL-inspired Aggregation

Now consider a different thought experiment where all agents use the same pre-trained and fine-tuned model of weights W^{SFT} but each one uses a different reward function $r_i(x, y)$ for RLHF. We combine their LLMs by solving the (RL)

problem on the combined reward functions weighted by the bids. We will solve the problem:

$$\max_W \mathcal{L}_{\mathsf{RL}}^{\mu,\bar{r}}(f^W), \quad \bar{r} = \frac{\sum_i b_i r_i}{\sum_i b_i}.$$

Analogously to Proposition C.1, we show that we can obtain the solution to the aggregated problem by combining the solutions on each dataset. We defer the proof of Proposition C.3 to Appendix H.

Proposition C.3. *Consider datasets μ, r_i and let f^{SFT} be the solution of program (KL) with data μ. If f_i is the maximizer of $\mathcal{L}_{\mathsf{RL}}^{\mu,r_i}$, let f^* be the maximizer of $\mathcal{L}_{\mathsf{RL}}^{\mu,\bar{r}}$ where $\bar{r}$ is the weighted average of the reward functions, then f^* is the function minimizing:*

$$\min_f \sum_x \mu(x) \sum_i b_i D_{\mathsf{KL}}(f(x) \| f_i(x)). \tag{5}$$

Similar to what we did in the previous subsection, we can also define a welfare function inspired by Proposition C.3. Namely:

$$\overline{\mathrm{WEL}}_{\mathsf{KL}} = \sum_{i\in[n]} b_i \cdot \rho_{\mathsf{KL}}(q, p_i) = -\sum_{i\in[n]} \sum_{t\in T} b_i \cdot q(t) \cdot \ln \frac{q(t)}{p_i(t)}.$$

Lemma C.4. *The efficient aggregation function that maximizes $\overline{\mathrm{WEL}}_{\mathsf{KL}}$ is the log-linear combination of p_i:*

$$\forall t \in T, \ \ln \bar{q}_{\mathsf{KL}}(t) = C + \frac{\sum_i b_i \ln \cdot p_i(t)}{\sum_i b_i},$$

where C is a normalization constant such that $\sum_t \bar{q}_{\mathsf{KL}}(t) = 1$.

The proof follows directly from the proof of Proposition C.3. While not monotone (Example B.2), the log-linear aggregation function is a reasonable choice assuming that the agents' preferences are aligned with the KL-divergence loss for the RL-stage training.

D. Additional Related Work

To the best of our knowledge, the exact research question and our approaches in this work have not been previously studied. However, our work is indeed connected to a few lines of research. The most relevant to us is perhaps the growing literature on fine-tuning LLM, with reinforcement learning from human feedback (RLHF) as a representative approach (Wei et al., 2021; Bakker et al., 2022; Ouyang et al., 2022; Bai et al., 2022). At a high level, fine-tuning and RLHF seek to align a generally pre-trained LLM with certain desirable behaviors. This is in spirit analogous to our goal of designing LLMs to better align with a group of agents' overall preferences. However, our research challenges and methods are both different from fine-tuning. Specifically, fine-tuning refines the underlying model's parameters whereas our approach is one-layer up and directly aggregates the output distributions from multiple models. The main challenge we address is the potential incentive misalignment while eliciting LLM agents' preferences, whereas human labelers or other models that generate reward feedback for RLHF are assumed to be genuine and do not misrepresent their own preferences. In-context learning (Brown et al., 2020; Wei et al., 2022; 2023) are similar to us in the sense that they also do not change the model parameters. They influence token distributions by conditioning on better-generated prefix contexts, whereas our approach directly aggregates distributions from multiple LLM agents.

Another related line of research is the celebrated field of mechanism design (MD), particularly for the choice of a "public project" (Papadimitriou et al., 2008; Dughmi, 2011) (which is the output of the designed LLM in our situation) that maximizes a certain welfare function. Similar to these type of design problem, a core challenge in our problem is to elicit truthful preferences from unknown agents. However, the design problem in our case is fundamentally different — we choose a high-dimensional distribution from an $\mathbb{R}^T$ space with only partial knowledge about agents' preferences, whereas previous MD for public project typically pick a choice from a discrete (often exponentially large) set with clear agent valuation functions (Papadimitriou et al., 2008; Dughmi, 2011). Freeman et al. (2019); Goel et al. (2019) consider truthful aggregation of budget proposals. Their mechanisms do output a distribution over budgets that best serves the population. However, the objectives and techniques between our work and theirs are both different. First, their problem is mechanism design without money, whereas our problem has monetary transfers involved. A direct consequence of the difference is that their

mechanism will treat every participant with equal weight, whereas the weights of our participants are determined by their bids. Second, research on truthful budget proposal aggregations often assume explicit valuation functions (e.g., l_1 distance between preferred and output distributions), under which the VCG mechanism is truthful. Their main research question hence is to study additional properties of the mechanisms such as Pareto-efficiency and certain fairness properties (Freeman et al., 2019). Assuming such an explicit valuation function does not appear realistic in our problem, so our core research question is to design robust mechanisms that enjoy good incentive properties simultaneously for a broad range of valuation functions. From this perspective, our work also bears some similarity to the rich literature on robust mechanism design. Most of robust MD literature still assume existence of value functions with uncertainty modeled by Bayesian beliefs or in a max-min sense (Bergemann and Morris, 2005; 2012; Roughgarden and Talgam-Cohen, 2016; Carroll, 2015; Dütting et al., 2019). However, assuming such a valuation function over tokens or their distributions does not appear realistic in creatives generation, thus our model is more similar to a worst-case style consideration during which we only assume partial yet "obvious" preferences.

E. Omitted Proofs from Section B.2

Our proofs make use of the following lemma, which shows that any complete preference relation on a countable set, can be expressed via a function that maps elements of the countable set into the non-negative rationals.

Lemma E.1. *Let X be a countable set, and let $\succeq$ be a complete preference relation on X. Then there is a function $f : X \to \mathbb{Q}_+$ such that $x \preceq y$ if and only if $f(x) \leq f(y)$.*

Proof. Since X is countable, we can arrange it in a sequence $\{x_1, x_2, \ldots\}$. We describe a procedure to define $f(x_i)$ for each x_i in turn. Let $f(x_1) = 1$. Suppose we have defined $f(x_1)$, $f(x_2)$, ..., $f(x_n)$ in such a way that all order relations are preserved. That is, for all $i, j \leq n$, we have $x_i \preceq x_j$ if and only if $f(x_i) \leq f(x_j)$. We want to define $f(x_{n+1})$. If $x_{n+1} \succeq x_i$ and $x_{n+1} \preceq x_i$ (i.e., $x_{n+1} \sim x_i$) for some $i \leq n$, then we can let $f(x_{n+1}) = f(x_i)$ and we have extended the function to one more element in a way that preserves all order relations. Otherwise, we can partition the set $\{x_1, \ldots, x_n\}$ into two sets

$$L = \{x_i : \ i \leq n, \ x_i \prec x_{n+1}\} \quad \text{and} \quad H = \{x_i : \ i \leq n, \ x_i \succ x_{n+1}\}.$$

In $\mathbb{Q}_+$ every element of $f(L)$ is strictly smaller than every element of $f(H)$. Choose $q \in \mathbb{Q}_+$ strictly larger than all the elements of $f(L)$ and strictly smaller than all the elements of $f(H)$. For each x_i with $i \leq n$ the relationship between x_i and x_{n+1} is the same as the relationship between $f(x_i)$ and q. Therefore, letting $f(x_{n+1}) = q$ extends the function to one more element, preserving all order relations, as required. The resulting function defined on all of X is thus an isomorphism from X to the image of f. □

Proof of Lemma B.6

Proof. Since the distribution aggregation function q is consistent, we can define a preference relation $\succeq_{i,\boldsymbol{p}}$ on $\mathbb{Q}_+$ according to the total orders over $Q(\boldsymbol{b}_{-i}, \boldsymbol{p})$ for all possible $\boldsymbol{b}_{-i}$ (assumed by the lemma) such that, $\forall b_i, b'_i \in \mathbb{Q}_+$:

- If there exists $\boldsymbol{b}_{-i}$ such that $q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) \succ_i q(b'_i, \boldsymbol{b}_{-i}, \boldsymbol{p})$, then b_i is preferred to b'_i, i.e., $b_i \succ_{i,\boldsymbol{p}} b'_i$;
- If $\forall \boldsymbol{b}'_{-i}$, $q(b_i, \boldsymbol{b}'_{-i}, \boldsymbol{p}) = q(b'_i, \boldsymbol{b}'_{-i}, \boldsymbol{p})$, then b_i and b'_i are indifferent, i.e., $b_i \sim_{i,\boldsymbol{p}} b'_i$.

Note that the validity of this construction is guaranteed by the definition of consistency (Definition B.2):

- For any b_i, b'_i assigned $b_i \succ_{i,\boldsymbol{p}} b'_i$, there exists some $\boldsymbol{b}_{-i}$ such that $q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) \succ_i q(b'_i, \boldsymbol{b}_{-i}, \boldsymbol{p})$. By the definition of consistency, $\forall \boldsymbol{b}'_{-i}$, $q(b_i, \boldsymbol{b}'_{-i}, \boldsymbol{p}) \succeq_i q(b'_i, \boldsymbol{b}'_{-i}, \boldsymbol{p})$. So we never assign the contradicting order ($b'_i \succ_{i,\boldsymbol{p}} b_i$).
- Meanwhile, for any b_i, b'_i assigned $b_i \sim_{i,\boldsymbol{p}} b'_i$, we have that $\forall \boldsymbol{b}'_{-i}$, $q(b_i, \boldsymbol{b}'_{-i}, \boldsymbol{p}) = q(b'_i, \boldsymbol{b}'_{-i}, \boldsymbol{p})$, which implies that for all $\boldsymbol{b}'_{-i}$, $q(b_i, \boldsymbol{b}'_{-i}, \boldsymbol{p}) \not\succ_i q(b'_i, \boldsymbol{b}'_{-i}, \boldsymbol{p})$. Therefore, we never assign $b_i \succ_{i,\boldsymbol{p}} b'_i$.

By the assumption of the lemma, $\succeq_i$ establishes a total order over $Q(\boldsymbol{b}_{-i}, \boldsymbol{p})$. We argue that this implies that $\succeq_{i,\boldsymbol{p}}$ is a complete preference relation on $\mathbb{Q}_+$. Concretely, for every pair $b_i, b'_i \in \mathbb{Q}_+$ we have $q(b_i), q(b'_i) \in Q(\boldsymbol{b}_{-i}, \boldsymbol{p})$. So either for some $\boldsymbol{b}_{-i}$, $q(b_i) \succ_i q(b'_i)$ (or $q(b'_i) \succ_i q(b_i)$), or for all $\boldsymbol{b}_{-i}$, $q(b_i) = q(b'_i)$, due to the lemma's assumption of total order

over $Q(\boldsymbol{b}_{-i}, \boldsymbol{p})$. Hence every pair $b_i, b_i' \in \mathbb{Q}_+$ has an order under $\succeq_{i,\boldsymbol{p}}$ (i.e., $\succ_{i,\boldsymbol{p}}$ or $\sim_{i,\boldsymbol{p}}$), so the preference relation $\succeq_{i,\boldsymbol{p}}$ is complete.

By Lemma E.1, since $\mathbb{Q}_+$ is countable, there exists a function $f_{i,\boldsymbol{p}} : \mathbb{Q}_+ \to \mathbb{Q}_+$ that represents the preference relation $\succeq_{i,\boldsymbol{p}}$, i.e.,

$$\forall b_i, b_i' \in \mathbb{Q}_+,\ b_i \succeq_{i,\boldsymbol{p}} b_i' \iff f_{i,\boldsymbol{p}}(b_i) \geq f_{i,\boldsymbol{p}}(b_i').$$

Hence we can construct $\tilde{q}$ as:

$$\tilde{q}(f_{i,\boldsymbol{p}}(b_i), \boldsymbol{b}_{-i}, \boldsymbol{p}) = q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}).$$

Note that for any b_i, b_i' such that $f_{i,\boldsymbol{p}}(b_i) = f_{i,\boldsymbol{p}}(b_i')$, we have $b_i \sim_{i,\boldsymbol{p}} b_i' \implies q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) = q(b_i', \boldsymbol{b}_{-i}, \boldsymbol{p})$. Hence, the construction of $\tilde{q}$ never assigns different values to the same input.

To complete the construction of $\tilde{q}(\cdot, \boldsymbol{b}_{-i}, \boldsymbol{p})$, we expand its definition from the image set of $f_{i,\boldsymbol{p}}$ to the entire bid space while preserving monotonicity. The existence of the expansion is guaranteed by the monotonicity of $\tilde{q}(\cdot, \boldsymbol{b}_{-i}, \boldsymbol{p})$ on the image set of $f_{i,\boldsymbol{p}}$ and the fact that the space of aggregations is compact.

Letting $\pi_i(b) = f_{i,\boldsymbol{p}}(b)$, we have $q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) = \tilde{q}(\pi_i(b_i), \boldsymbol{b}_{-i}, \boldsymbol{p})$ and $\tilde{q}$ is a monotone distribution aggregation function for agent i. Applying the same argument and the relabeling procedure for every i from 1 to n, completes the proof. □

Proof of Lemma B.7

Proof. We first prove the "only if" ("$\implies$") direction. That is, suppose $\mathcal{M} = \langle q, z \rangle$ is payment-monotone, then it must imply a total order over $Q(\boldsymbol{b}_{-i}, \boldsymbol{p}) = \{q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) : b_i \in \mathbb{Q}_+\}$ for any fixed $\boldsymbol{b}_{-i}$ and $\boldsymbol{p}$.

Fix any $\boldsymbol{b}_{-i}$ and $\boldsymbol{p}$. For any $q, q' \in Q(\boldsymbol{b}_{-i}, \boldsymbol{p}) = \{q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) : b_i \in \mathbb{Q}_+\}$ such that $q = q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p})$, $q' = q(b_i', \boldsymbol{b}_{-i}, \boldsymbol{p})$. Let $z_i = z_i(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p})$ and $z_i' = z_i(b_i', \boldsymbol{b}_{-i}, \boldsymbol{p})$ be the corresponding payment given by $\mathcal{M}$. Without loss of generality, suppose that $z_i \geq z_i'$. Then by payment-monotonicity of the mechanism $\mathcal{M}$, we have $q \succeq_i q'$. In other words, $\succeq_i$ establishes a total order over Q.

Next we show the "if" ("$\impliedby$") direction. That is, given a total order over $Q(\boldsymbol{b}_{-i}, \boldsymbol{p}) = \{q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) : b_i \in \mathbb{Q}_+\}$ for any fixed $\boldsymbol{b}_{-i}$ and $\boldsymbol{p}$, we can construct a payment rule z such that $\mathcal{M} = \langle q, z \rangle$ is payment-monotone.

Fix any $\boldsymbol{b}_{-i}$ and $\boldsymbol{p}$. Since $\succeq_i$ establishes a total order over $Q(\boldsymbol{b}_{-i}, \boldsymbol{p})$ and $Q(\boldsymbol{b}_{-i}, \boldsymbol{p})$ is countable, by Lemma E.1 there exists a function $f_{i,\boldsymbol{b}_{-i},\boldsymbol{p}} : Q(\boldsymbol{b}_{-i}, \boldsymbol{p}) \to \mathbb{Q}_+$ such that $\forall q, q' \in Q(\boldsymbol{b}_{-i}, \boldsymbol{p})$,

$$q \succeq_i q' \iff f_{i,\boldsymbol{b}_{-i},\boldsymbol{p}}(q) \geq f_{i,\boldsymbol{b}_{-i},\boldsymbol{p}}(q').$$

Letting $z_i(\boldsymbol{b}, \boldsymbol{p}) = f_{i,\boldsymbol{b}_{-i},\boldsymbol{p}}(q(\boldsymbol{b}, \boldsymbol{p}))$, we obtain a payment-monotone mechanism $\langle q, z \rangle$. □

Proof of Theorem B.5

Proof. Applying Lemma B.7, we know that the payment monotonicity of $\mathcal{M}$ implies a total order over $Q(\boldsymbol{b}_{-i}, \boldsymbol{p})$ for any fixed $\boldsymbol{b}_{-i}$ and $\boldsymbol{p}$. Then applying Lemma B.6, we know that for any $\boldsymbol{p}$ there exist strategy mappings $\pi_i : \mathbb{Q}_+ \to \mathbb{Q}_+$ for each i such that $q(\boldsymbol{b}, \boldsymbol{p}) = \tilde{q}(\pi(\boldsymbol{b}), \boldsymbol{p})$ and $\tilde{q}(\cdot, \boldsymbol{p})$ is a monotone aggregation function.

Now, following the same argument in the proof of Lemma B.6, let us further define $\tilde{z}(\pi(\boldsymbol{b}), \boldsymbol{p}) = z(\boldsymbol{b}, \boldsymbol{p})$ for each $\boldsymbol{b} \in \mathbb{Q}_+^n$ and $\boldsymbol{p} \in \Delta(T)^n$. So $\mathcal{M} = \langle q, z \rangle$ is strategically equivalent to $\tilde{\mathcal{M}} = \langle \tilde{q}, \tilde{z} \rangle$ by definition.

It remains to show that the mechanism $\tilde{\mathcal{M}} = \langle \tilde{q}, \tilde{z} \rangle$ is payment-monotone. We know that the original mechanism $\mathcal{M} = \langle q, z \rangle$ satisfies payment monotonicity, meaning that for each $\boldsymbol{p}$, $\boldsymbol{b}_{-i}$, b_i, b_i',

$$z_i(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) \geq z_i(b_i', \boldsymbol{b}_{-i}, \boldsymbol{p}) \iff q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) \succeq_i q(b_i', \boldsymbol{b}_{-i}, \boldsymbol{p}).$$

But then, with $\boldsymbol{b} = (b_i, \boldsymbol{b}_{-i})$ and $\boldsymbol{b}' = (b_i', \boldsymbol{b}_{-i})$, we also have

$$\begin{aligned} & \tilde{z}_i(\pi(\boldsymbol{b}), \boldsymbol{p}) = z_i(\boldsymbol{b}, \boldsymbol{p}) \geq z_i(\boldsymbol{b}', \boldsymbol{p}) = \tilde{z}_i(\pi(\boldsymbol{b}'), \boldsymbol{p}) \\ \iff \quad & \tilde{q}(\pi(\boldsymbol{b}), \boldsymbol{p}) = q(\boldsymbol{b}, \boldsymbol{p}) \succeq_i q(\boldsymbol{b}', \boldsymbol{p}) = \tilde{q}(\pi(\boldsymbol{b}'), \boldsymbol{p}), \end{aligned}$$

so the pair $\tilde{q}$, $\tilde{z}$ satisfies payment monotonicity as needed. □

F. Omitted Proofs from Section B.3

Proof of Lemma B.8

Proof. We first prove the "only if" ("$\Longrightarrow$") direction. Suppose q is a monotone distribution aggregation function. By Definition B.3, for any agent i and $b_i' \geq b_i \geq 0$, we have

$$q(b_i', \boldsymbol{b}_{-i}, \boldsymbol{p}) \succeq_i q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) \succeq_i q(0, \boldsymbol{b}_{-i}, \boldsymbol{p}).$$

For any undersampled token $t \in T_+$, because $q_t(0, \boldsymbol{b}_{-i}, \boldsymbol{p}) \leq (p_i)_t$, then by Definition A.1, we have

$$q_t(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}), q_t(b_i', \boldsymbol{b}_{-i}, \boldsymbol{p}) \in [q_t(0, \boldsymbol{b}_{-i}, \boldsymbol{p}), (p_i)_t].$$

Hence by $q(b_i', \boldsymbol{b}_{-i}, \boldsymbol{p}) \succeq_i q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p})$, we have

$$q_t(b_i', \boldsymbol{b}_{-i}, \boldsymbol{p}) \geq q_t(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}),$$

namely, $q_t(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p})$ weakly increases with b_i and never goes above $(p_i)_t$.

Similarly, we can prove that for any oversampled token $t \in T_-$, $q_t(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p})$ weakly decreases with b_i and never goes below $(p_i)_t$.

Then we prove the "if" ("$\Longleftarrow$") direction. Consider any $b_i' \geq b_i$. For any undersampled token $t \in T_+$, as $q_t(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) \leq (p_i)_t$ weakly increases with b_i, we have

$$q_t(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) \leq q_t(b_i', \boldsymbol{b}_{-i}, \boldsymbol{p}) \leq (p_i)_t.$$

Similarly, we have for any oversampled token $t \in T_-$,

$$q_t(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}) \geq q_t(b_i', \boldsymbol{b}_{-i}, \boldsymbol{p}) \geq (p_i)_t.$$

Then by Definition A.1,

$$q(b_i', \boldsymbol{b}_{-i}, \boldsymbol{p}) \succeq_i q(b_i, \boldsymbol{b}_{-i}, \boldsymbol{p}),$$

which then implies the monotonicity of q. □

Proof of Theorem B.10

Proof. Given an aggregation $q(b_i)$ we construct a stable sampling procedure σ. Let $Q_+(b_i)$ and $Q_-(b_i)$ be the probability of sampling tokens from T_+ and T_-:

$$Q_+(b_i) = \sum_{t \in T_+} q_t(b_i), \qquad Q_-(b_i) = \sum_{t \in T_-} q_t(b_i).$$

Both are monotone as q is monotone (by Lemma B.8). Reparameterize functions $q_t(b_i)$ in the range $I = [Q_+(0), Q_+(\infty)]$ by defining:[7]

$$\hat{q}_t(x) = q_t(Q_+^{-1}(x)), \forall t \in T, x \in I.$$

Since $\hat{q}_t(x)$ is monotone, by Lebesgue's Differentiation Theorem, it is differentiable almost everywhere on I. We observe that:

$$\sum_{t \in T_+} \hat{q}_t(x) = Q_+(Q_+^{-1}(x)) = x, \quad \sum_{t \in T_-} \hat{q}_t(x) = Q_-(Q_+^{-1}(x)) = 1 - x.$$

Then $\hat{q}_t'(x)$ for $t \in T_+$ forms a probability distribution over T_+. Similarly $-\hat{q}_t'(x)$ for $t \in T_-$ forms a probability distribution over T_-. Define $\kappa^+(x), \kappa^-(x) \in \Delta(T)$ as $\kappa_t^+(x) = \hat{q}_t'(x)$ for $t \in T_+$, and zero otherwise and $\kappa_t^-(x) = -\hat{q}_t'(x)$ for $t \in T_-$ and zero otherwise.

[7] Here $Q_+^{-1}(x)$ refers to a generalized inverse (or the quantile function (Bulow and Roberts, 1989)) so that it is properly defined even when $Q_+(x)$ is discontinuous.

We also define the vector $q^+, q^- \in \Delta(T)$ such that $q_t^+ = q_t(0)/Q_+(0)$ for $t \in T_+$ and zero for $t \in T_-$. Similarly: $q_t^- = q_t(\infty)/Q_+(\infty)$ for $t \in T^-$ and zero otherwise.

Finally, we define a deterministic function

$$\mathsf{Sampler} : \Delta(T) \times [0,1] \to T$$

that takes a probability vector $p \in \Delta(T)$ and $r \in [0,1]$ and outputs an index $t \in T$ such that $\sum_{j<t} p_j < r \leq \sum_{j \leq t} p_j$.

Now, we are ready to define the stable sampling procedure. Let $\mathcal{R}$ be the uniform distribution on $[0,1]^2$. Given $r = (r_A, r_B) \sim \mathcal{R}$ we define the output $t = \sigma(b_i, r)$ as follows:

1. if $r_A \leq Q_+(0)$, $t = \mathsf{Sampler}(q^+, r_B) \in T_+$;
2. if $Q_+(0) < r_A \leq Q_+(b_i)$, $t = \mathsf{Sampler}(\kappa^+(r_A), r_B) \in T_+$;
3. if $Q_+(b_i) < r_A \leq Q_+(\infty)$, $t = \mathsf{Sampler}(\kappa^-(r_A), r_B) \in T_-$;
4. if $Q_+(\infty) < r_A$, $t = \mathsf{Sampler}(q^-, r_B) \in T_-$.

Since $\sigma(b_i, r)$ is deterministic, for any fixed r, either the output can not be influenced by the bid ($r_A < Q_+(0)$ or $r_A > Q_+(\infty)$) or it can only cause the output to shift from an oversampled token $\mathsf{Sampler}(\kappa^-(r_A), r_B)$ to an undersampled token $\mathsf{Sampler}(\kappa^+(r_A), r_B)$.

We now argue that the tokens are sampled with the correct probabilities. For $t \in T_+$, the total probability of getting sampled is:

$$\begin{aligned}\int_0^{Q_+(0)} q_t^+ \mathrm{d}r_A + \int_{Q_+(0)}^{Q_+(b)} \hat{q}_t'(r_A)\mathrm{d}r_A &= q_t(0) + \hat{q}_t(Q_+(b)) - \hat{q}_t(Q_+(0)) \\ &= q_t(0) + q_t(b) - q_t(0) = q_t(b).\end{aligned}$$

Similarly for tokens in T_-, the probability of being sampled is:

$$\begin{aligned}&\int_{Q_+(b)}^{Q_+(\infty)} -\hat{q}_t'(r_A)\mathrm{d}r_A + \int_{Q_+(\infty)}^{1} q_t^- \mathrm{d}r_A \\ &= \hat{q}_t(Q_+(b)) - \hat{q}_t(Q_+(\infty)) + q_t(\infty) = q_t(b) - q_t(\infty) + q_t(\infty) = q_t(b).\end{aligned}$$

This completes the proof. □

G. Universally Stable Sampling

[Counterexample 4-token] Consider two agents $\{1, 2\}$ and 4 tokens $\{t_1, t_2, t_3, t_4\}$. Assume that both agents have the same preferred distribution $p_1 = p_2 = (0, 0, .5, .5)$ and the allocation function is such that if both agents bid zero the allocation is $(.5, .5, 0, 0)$. Hence both both agents have the same set of favored tokens $T_+ = \{t_3, t_4\}$ and less favored tokens $T_- = \{t_1, t_2\}$. The aggregation function $q(b_1, b_2, p_1, p_2)$ is given by the following table:

	$b_1 = 0$	$b_1 = 1$
$b_2 = 0$	$q_{00} = (.5, .5, 0, 0)$	$q_{10} = (0, .5, .5, 0)$
$b_2 = 1$	$q_{01} = (.5, 0, .5, 0)$	$q_{11} = (0, 0, .5, .5)$

One can verify that the aggregation function is monotone: When either of the agents increase the bid from 0 to 1, exactly $1/2$ of the probability mass moves from $T_- = \{t_1, t_2\}$ to $T_+ = \{t_3, t_4\}$.

Now we show that there does not exist a universally stable sampling algorithm that implements this aggregation function. Suppose there exist one, σ, let $r_A = \{r | \sigma(q_{00}, r) = t_1\}$ and $r_B = \{r | \sigma(q_{00}, r) = t_2\}$. Because σ is stable for bidder 1, when bidder 1 increases bid, the probability mass only transfers from T_- to T_+.

In this case, when the bid profile (b_1, b_2) moves from $(0,0)$ to $(1,0)$, we must have

$$\sigma(q_{10}, r) = \begin{cases} t_3, & r \in r_A \\ t_2, & r \in r_B \end{cases}.$$

Further apply the same argument when (b_1, b_2) moves from $(1,0)$ to $(1,1)$, we have

$$\sigma(q_{11}, r) = \begin{cases} t_3, & r \in r_A \\ t_4, & r \in r_B \end{cases}.$$

However, if we consider (b_1, b_2) moves along the path $(0,0) \to (0,1) \to (1,1)$, we should have

$$\sigma(q_{01}, r) = \begin{cases} t_1, & r \in r_A \\ t_3, & r \in r_B \end{cases}, \qquad \sigma(q_{11}, r) = \begin{cases} t_4, & r \in r_A \\ t_3, & r \in r_B \end{cases}.$$

We end up with a contradiction on the value of $\sigma(q_{11}, r)$ while moving from $(0,0)$ to $(1,1)$ along two different paths.

[Counterexample 3-token] Consider two agents $\{1,2\}$ and 3 tokens $\{t_1, t_2, t_3\}$, where the agents have different sets of favored (less favored) tokens. In particular, $T_1^+ = \{t_1, t_3\}, T_1^- = \{t_2\}$ and $T_2^+ = \{t_3\}, T_2^- = \{t_1, t_2\}$. The aggregation function is given by the following table:

	$b_1 = 0$	$b_1 = 1$
$b_2 = 0$	$q_{00} = (.5, .5, 0)$	$q_{10} = (.5, 0, .5)$
$b_2 = 1$	$q_{01} = (0, .5, .5)$	$q_{11} = (.5, 0, .5)$

One can verify that the aggregation function is monotone: When b_1 increases from 0 to 1, exactly $1/2$ of the probability mass moves from t_2 to t_3 (when $b_2 = 0$) or t_1 (when $b_2 = 1$). When b_2 increases from 0 to 1, either $1/2$ of the probability mass moves from t_1 to t_3 (when $b_1 = 0$) or no move (when $b_1 = 1$).

Similarly, suppose that there exists a universally stable sampling algorithm σ that implements q. Let $r_A = \{r | \sigma(q_{00}, r) = t_1\}$ and $r_B = \{r | \sigma(q_{00}, r) = t_2\}$.

Following the same argument in Example G, consider the bid profile (b_1, b_2) moves along the path $(0,0) \to (1,0) \to (1,1)$, we must have

$$\sigma(q_{10}, r) = \sigma(q_{11}, r) = \begin{cases} t_1, & r \in r_A \\ t_3, & r \in r_B \end{cases}.$$

However, consider the bid profile (b_1, b_2) moves along the path $(0,0) \to (1,0) \to (1,1)$, we must have

$$\sigma(q_{01}, r) = \begin{cases} t_3, & r \in r_A \\ t_2, & r \in r_B \end{cases}, \qquad \sigma(q_{11}, r) = \begin{cases} t_3, & r \in r_A \\ t_1, & r \in r_B \end{cases}.$$

We end up with a contradiction on the value of $\sigma(q_{11}, r)$.

H. Omitted Proofs from Section C

Proof of Proposition C.1

Proof of Proposition C.1. We prove the theorem by showing that the loss $\mathcal{L}_{\mathsf{KL}}^{\bar{\mu}}$ and the loss in equation (4) differ by a constant and hence have the same minimizer. For $B = \sum_i b_i$ and a fixed x we will show that:

$$B \cdot D_{\mathsf{KL}}(\sum_i \frac{b_i}{B} \mu_i(\cdot|x) \| f^W(x)) - \sum_i b_i D_{\mathsf{KL}}(f_i(x) \| f^W(x)) = \mathsf{const}.$$

For notation simplicity, we omit the parameters x and W when it is clear from the context and write $\sum_i b_i f_i(x)/B = \bar{f}(x)$. Below we treat any term that doesn't depend on f as a constant:

$$\begin{aligned}
\sum_i b_i D_{\mathsf{KL}}(f_i \| f) &= \sum_i b_i H(f_i) - \sum_y b_i f_i(y|x) \ln f(y|x) \\
&= -B \sum_y \frac{\sum_i b_i f_i(y|x)}{B} \cdot \ln f(y|x) + \sum_i b_i H(f_i) \\
&= -B \cdot H(\bar{f}, f) + \sum_i b_i H(f_i) \\
&= B \cdot D_{\mathsf{KL}}(\bar{f} \| f) - B \cdot H(\bar{f}) + \sum_i b_i H(f_i) \\
&= B \cdot D_{\mathsf{KL}}(\bar{f} \| f) - \mathsf{const}.
\end{aligned}$$

To complete the proof, observe that if f_i is the unconstrained minimizer of $\mathcal{L}_{\mathsf{KL}}^{\mu_i}(f)$ we must have $f_i(y|x) = \mu_i(y|x)$. □

Proof of Lemma C.2

Proof of Lemma C.2. Let $B = \sum_i b_i$ and consider W_{KL}:

$$\begin{aligned}
W_{\mathsf{KL}} &= -\sum_{i \in [n]} b_i \cdot (H(p_i, q) - H(p_i)) \\
&= \sum_{i \in [n]} b_i \cdot H(p_i) + \sum_{i \in [n]} b_i \cdot \sum_{t \in [T]} p_i(t) \ln q(t) \\
&= \sum_{i \in [n]} b_i \cdot H(p_i) + \sum_{t \in [T]} \ln q(t) \sum_{i \in [n]} b_i \cdot p_i(t) \\
&= \sum_{i \in [n]} b_i \cdot H(p_i) + B \cdot \sum_{t \in [T]} \frac{\sum_{i \in [n]} b_i \cdot p_i(t)}{B} \ln q(t) \\
&= \sum_{i \in [n]} b_i \cdot H(p_i) - B \cdot H(q_{\mathsf{KL}}, q)
\end{aligned}$$

By Gibbs' inequality, the cross entropy $H(q_{\mathsf{KL}}, q)$ is minimized if and only if $q = q_{\mathsf{KL}}$. Hence this is also the maximizer of W_{KL}. □

Proof of Proposition C.3

Proof. For a fixed x, $f^*(y|x)$ can be obtained by solving:

$$\begin{aligned}
\max_{f(\cdot|x)} \ & \sum_y f(y|x) \bar{r}(x, y) - \beta D_{\mathsf{KL}}(f(x) \| f^{\mathsf{SFT}}(x)) \\
\text{s.t.} \ & \sum_y f(y|x) = 1 \text{ and } f(y|x) \geq 0.
\end{aligned}$$

By the standard KKT conditions, the solution has the form:

$$f^*(y|x) = f^{\mathsf{SFT}}(y|x) e^{\bar{r}(x,y)/\beta} C_x.$$

where C_x is a normalization constant to ensure $\sum_y f^*(y|x) = 1$. For the same reason, we have that:

$$f_i(y|x) = f^{\mathsf{SFT}}(y|x) e^{r_i(x,y)/\beta} C_{i,x}.$$

If f° is the function minimizing problem (5), then we can apply KKT conditions to obtain:

$$f^\circ(y|x) = \exp\left(\frac{1}{B} \sum_i b_i \ln f_i(y|x)\right) \cdot C'_x$$

for normalization constants C'_x and $B = \sum_i b_i$. Replacing the formula for $f_i(x)$ from the previous line, we obtain that:

$$\begin{aligned} f^\circ(y|x) &= \exp\left(\frac{1}{B}\sum_i b_i \left(r_i(x,y)/\beta + \ln f^{\mathsf{SFT}}(y|x)\right)\right) \cdot C''_x \\ &= \exp\left(\bar{r}(x,y)/\beta + \ln f^{\mathsf{SFT}}(y|x)\right) \cdot C''_x = f^*(y|x). \end{aligned}$$

This completes the proof.

□