Controlling the Conversation[*]

Martino Banchio — Bocconi, IGIER, Google Research
Bing Liu — Stanford
Andres Perlroth — Reddit

February 10, 2026

Abstract

We study how AI assistants design their conversations. An assistant, with a private objective, guides a user through her search via sequential queries, modeled as partitions of the search space. The user selects the subset with the highest average value. Using a prior-free framework, we minimize the platform's regret relative to an omniscient benchmark. We provide an optimal randomized policy that achieves bounded regret of $\frac{1}{n-1}$, where n is the number of items. This policy is implemented as a sequence of "recommendations", that is, the AI sequentially proposes candidates from the item set until one is accepted.

1 Introduction

The rise of Large Language Models (LLMs) and conversational assistants represents a paradigm shift in how users discover content and make decisions. Unlike traditional search engines that present a static list of results, a conversational assistant interacts with the user sequentially. To help a user find their desired outcome, the assistant asks questions to clarify the user's intent. For example, when a user asks for a restaurant recommendation, the assistant effectively narrows down the search space by asking, "Are you looking for a casual dinner or a fine dining experience?"

This process of eliciting intent has an algorithmic flavor. In this paper, we model each question by the assistant as a partition of the set of possible items. The user, who has some intrinsic value in mind, chooses the element of the partition with the largest

[*]A substantial portion of the work was developed while all authors were at Google Research.

average value. Through this selection, the assistant gleans some insight into the user's preferences, allowing it to ask a follow-up question and zooming in further until a final item is selected.

A nuanced dynamic emerges when the platform considers objectives that extend beyond the user's immediate selection. While the user naturally seeks to maximize their idiosyncratic private value v, the platform may operate under an objective function π that captures broader, community-oriented goals. For instance, π may represent a commitment to safety, fairness, or the long-term health of the information ecosystem. By prioritizing diversity, the platform can ensure users are exposed to a balanced perspective and high-quality sources. In this paper, we model this interaction abstractly as a platform optimizing a utility π distinct from the user's utility v, under uncertainty about v. We focus on the theoretical properties of such conversational guidance.[1]

We model the optimal steering of a conversation within this theoretical framework. The AI perfectly observes its own payoffs $\pi = (\pi_1, \ldots, \pi_n)$ for the n items, but does not observe the corresponding private values $v = (v_1, \ldots, v_n)$ of the user. The interaction between the AI and the user proceeds in stages. The user, who is initially agnostic about all items, is presented sequentially with partitions of the item set, i.e. questions. Upon observing the average value of each element of the partition, the user selects the partition element with largest average value. The conversation continues until the user selects a single-item partition, which corresponds to a final decision.

By choosing a fully contingent plan of questions (i.e., partitions of the item set), the platform aims to minimize the gap between the payoff π_i of the item eventually chosen by the user and the payoff π^* of the best item the user could have chosen under some sequence of questions. That is, the platform maximizes the competitive ratio against an "omniscient" benchmark, where the AI can partition items with full knowledge of the value profile v. We construct this optimal policy for the AI platform.

While the space of policies grows super-exponentially in the number of items, we first show that it is sufficient to consider a much smaller set of policies, which we refer to as *recommenders*. A question is a recommendation if it induces a binary partition where one of the items of the partition is a singleton. In words, a recommendation asks the user: "Do you want this item, or something else?" Recommenders are policies where each question is a recommendation. That is, the platform proposes an item in every turn, and the user

[1]Of course, allowing platforms to steer user conversations should be done with the utmost care, ensuring a high quality experience for the user. This paper should be viewed as a theoretical exercise, and we are not making any claims as to the practicality of the policies we derive in this paper from a regulatory or policy perspective.

“peels” at the set of items, sequentially.

We then characterize the optimal deterministic policy, which boils down to the order in which the items are recommended to the user. Unfortunately, the competitive ratio of this policy can be arbitrarily low. That is, we can construct examples where the ratio between the value delivered by this policy and the value of the omniscient policy is arbitrarily close to zero, for any number of items. It is then natural to turn to randomization. We construct an optimal randomized policy, which is simply a sequence of randomized recommendations. The policy is fully described by a linear program, and it achieves a bounded competitive ratio of $\frac{1}{n-1}$, where n is the number of items. This bound is tight among all randomized policies.

Finally, we propose two further improvements to the optimal policy. First, we extend this optimal policy to a sequentially optimal one: the algorithm randomizes in every turn, minimizing the conditional competitive ratio at every stage of the conversation. Second, we accommodate outside options for the user. This requires more variety in the recommendations of the platform, which in turn reduces user churn.

1.1 Related Work

Our work contributes to the active literature on the design of conversational search systems and the emerging economics of advertising in Large Language Models (LLMs). The early work of Radlinski and Craswell (2017) conceptualized conversational search as a multi-turn process where the system must model user intent dynamically. For a survey of the literature we refer the reader to Jannach et al. (2021). Unlike this literature, we focus on the role of incentives in optimal conversation design. We abstract away from many engineering challenges to capture the essence of a multi-turn conversation, and the underlying tradeoffs stemming from the misalignment in incentives between platform and user.

Within this literature we contribute to recent studies on the topic of "influenced" conversation steering. Much of the literature considers advertising as their primary motive for steering, for example Bhawalkar et al. (2025). Along similar threads Banchio et al. (2025) studies the optimal timing of advertising along the conversation, whereas this work studies the platform’s direct influence on the conversation flow. Unlike our work, Bergemann et al. (2024) assumes the user has private information about its preferences. We assume that the conversation allows the user to discover its own preferences, making the role of the conversation as a search tool explicit. Many works are then concerned with the engineering aspects of conversational steering, starting with Duetting et al. (2024), Dubey et al. (2024), and Soumalias et al. (2024).

Our work contributes to the rapidly evolving literature on algorithmic principal-agent problems (Dütting et al. (2024)). Specifically, our framework is closely related to the delegation problem, where a principal seeks to influence an agent to take a specific action on their behalf (Holmstrom (1980); Kleinberg and Kleinberg (2018)). While the standard delegation model focuses on a principal who restricts the agent's search space, our framework introduces a dynamic decision process between the principal's suggestions and the agent's subsequent decisions over those suggestions. Interestingly, a significant distinction arises regarding deterministic guarantees for the worst case scenario. While Kleinberg and Kleinberg (2018) proves that for the delegation model a deterministic mechanism exists achieving 1/2 of the optimal policy, in our dynamic model we show that no protocol exists achieving a bounded guarantee over the optimal policy.

Finally, some recent work studies the dynamics of users interacting with LLM chatbots from an empirical perspective, see for example Jahani et al. (2024) and Lin (2025). In particular, Werner et al. (2024) shows that LLM are indeed able to steer users in conversations. In this paper we take a normative perspective on the shape of the dynamics of such interactions for a particular objective, and we provide a microfoundation for the use of recommendations as a primary tool to direct search.

Our methodological approach contributes to the literature on online learning algorithms. We model the conversation as a sequential discovery of the input, not unlike the Pandora's box search problem, but with a twist: the search order is determined by an intermediary rather than the searcher herself.

2 Model

We consider a dynamic interaction between a platform (AI) and a user, centered on a set of n distinct items, denoted by $I = \{1,\ldots,n\}$. Each item $i \in I$ is characterized by two attributes: a value to the platform, $\pi_i \in \mathbb{R}_+$, and a private value to the user, $v_i \in \mathbb{R}_+$. We assume the platform's value is deterministic and known to the platform. Without loss of generality, we index the items such that $\pi_1 \geq \pi_2 \geq \cdots \geq \pi_n$. In contrast, the user's valuation profile $v = (v_1,\ldots,v_n)$ is unknown to the platform (and to the user). We adopt a prior-free approach, where the platform seeks to guarantee performance across all possible realizations of v.

2.1 The Conversation Protocol

The interaction proceeds in discrete rounds, representing a conversation where the platform guides the user toward a final choice. At the beginning of the process (round $t = 0$), the active set of items is the full set I. In each round, the platform presents a "question" to the user, modeled formally as a partition of the currently active set. The user then selects one element from this partition, which becomes the active set for the subsequent round. This process of refinement continues until the user selects a singleton set $\{i\}$, at which point the process terminates, the user receives item i, and the platform accrues payoff π_i.

We model the user as ex-ante uninformed about the available choices and their values. A user, when presented with a subset of items $S \subseteq I$, inspects it to costlessly learn the average value of its objects, denoted by

$$v(S) := \frac{1}{|S|} \sum_{i \in S} v_i.$$

She cannot instead inspect each individual object s within the set, unless presented with the singleton $\{s\}$. Faced with a partition $\mathcal{P}_t$ of the current active set, the user acts myopically to maximize her immediate utility, selecting the subset $S \in \mathcal{P}_t$ with the highest average value.[2] If multiple subsets share the maximum average value, ties are broken in favor of the platform.

Formally, the timeline is as follows: The platform initiates the conversation by offering a partition $\mathcal{P}_0$ of I. The user selects $S_1 \in \arg\max_{S \in \mathcal{P}_0} v(S)$. In the next round, the platform presents a partition $\mathcal{P}_1$ of S_1, and the user selects $S_2 \in \arg\max_{S \in \mathcal{P}_1} v(S)$. This continues until a singleton is chosen. With this mechanics we capture one aspect of conversational search, that is, vague intent (large sets) being refined into specific choices (singletons) via sequential queries.[3]

Remark. This conversation protocol has a desirable user-no-regret property: the user never wishes she had chosen a different element of previous rounds' partitions. To see this formally, fix an algorithm P. In round t, the platform presents a partition P^t, and the

[2] We use the average as a natural signal of the user's value for a particular set of items. An alternative way to interpret this model takes as primitive some taste parameters for the user and allows the conversational assistant to partition the set of items according to individual (or joint) taste profiles, as for example with questions such as "Do you like blue or red shoes?"

[3] Naturally, we instead abstract away from many other interesting aspects of conversational search in this theoretical model.

user chooses

$$S^t \in \arg\max_{S \in P^t} v(S).$$

In round $t+1$, the platform presents a partition P^{t+1} of S^t. We say the user regrets her round-t choice if there exists some $S \in P^t$ such that every option available in round $t+1$ is weakly worse than S, with at least one strict loss; i.e.,

$$\exists\, S \in P^t \text{ s.t. } \forall\, S' \in P^{t+1}, \quad v(S) \geq v(S') \text{ and } \exists\, S' \in P^{t+1} \text{ with } v(S) > v(S').$$

Such regret cannot occur under our protocol. First, observe that by optimality of S^t, we have $v(S^t) \geq v(S)$ for all $S \in P^t$. Second, the total value in S^t is $v(S^t) \cdot |S^t|$, whereas the total value in each subset S' is $v(S') \cdot |S'|$, and

$$\sum_{S' \in P^{t+1}} v(S') \cdot |S'| = v(S^t) \cdot |S^t|.$$

This implies that either $V(S') = v(S^t)$ for all $S' \in P^{t+1}$, or there exists a $S' \in P^{t+1}$ such that $v(S') > v(S^t)$. In the former case, the first observation implies that the user has no regret. In the latter, it is simple to see that the user has no regret, because $v(S') > v(S^t) \geq v(S)$ for all $S \in P^t$.

2.2 Algorithms and Payoffs

A deterministic algorithm (or policy) P is a mapping that specifies the sequence of partitions the platform presents, contingent on the user's history of selections. Since the user's behavior is deterministic conditional on v, the realized path of the conversation and the final item selected are fully determined by the algorithm P and the valuation profile v. We denote the profit generated by algorithm P under profile v as $\pi(P, v)$.

We also consider randomized algorithms, denoted by p, which are probability distributions over the set of deterministic algorithms $\mathcal{P}$. For a randomized algorithm defined by a distribution $p \in \Delta(\mathcal{P})$, the expected profit under valuation profile v is given by $\mathbb{E}_{P\sim p}[\pi(P, v)]$. Equivalently, a randomized algorithm induces a distribution over each stage's partitions conditional on the current active set.

2.3 The Omniscient Benchmark

To evaluate the performance of the platform's policy, we benchmark it against an *omniscient algorithm* that knows the user's valuation profile v ex-ante. The omniscient planner seeks to partition the set I to maximize the platform's value of the finally selected item, subject to the user's average-value selection rule.

For a fixed profile v, let $\overline{S}(v) = \{i \in I : v_i \geq v(I)\}$ be the set of items with value at least the global average. It is straightforward to see that any item $i \notin \overline{S}(v)$ cannot be the outcome of a conversation starting with the full set.

Proposition 1. For any algorithm P, $\pi(P,v) \in \{\pi_i : i \in \overline{S}(v)\}$.

Proof. Suppose by contradiction that there exists an algorithm P such that $\pi(P,v) \notin \{\pi_i : i \in \overline{S}(v)\}$. Let $j \notin \overline{S}$ be the item selected by the algorithm P. Let T be the termination round of the algorithm where j is selected and let $(S^t)_{t\leq T}$ be the active set of items in each round. Then it must be that

$$v_j = v(S^T) \geq v(S^{T-1}) \geq \ldots v(S^0) = v(I),$$

which implies that $j \in \overline{S}(v)$, a contradiction. □

Conversely, an omniscient planner can structure a partition that isolates a specific target item $i^* \in \overline{S}(v)$ against the rest of the items. Specifically, the partition $\mathcal{P}_v = \{\{i^*\}, I \setminus \{i^*\}\}$ ensures the user selects $\{i^*\}$ if and only if $v_{i^*} \geq v(I \setminus \{i^*\})$, which is equivalent to $v_{i^*} \geq v(I)$.

Thus, the maximum payoff achievable by an omniscient planner for a given realization v is:

$$\pi^*(v) = \max_{i \in \overline{S}(v)} \pi_i.$$

This benchmark represents the best the platform could achieve if it could perfectly "steer" the user by exploiting its full knowledge of v.

2.4 Objective

The platform's goal is to design a randomized algorithm that performs well against this benchmark across all possible valuation profiles. We seek to maximize the *competitive ratio* of the algorithm. We define the competitive ratio as the ratio between the performance of algorithm p and the omniscient benchmark's payoff. The competitive ratio then takes the form

$$\gamma(p) = \min_{v \in \mathbb{R}_+^n} \frac{\mathbb{E}_{P\sim p}[\pi(P,v)]}{\pi^*(v)}.$$

Our main theoretical contribution is the characterization of optimal randomized algorithms, that is, algorithms that maximize this guarantee.

Definition 1. An algorithm p is optimal if $p \in \operatorname{argmax}_p \min_{v \in \mathbb{R}_+^n} \frac{\mathbb{E}_{P \sim p}[\pi(P,v)]}{\pi^*(v)}$.

Additionally, we prove that the competitive ratio under such optimal algorithms is bounded by $\frac{1}{n-1}$, and the bound is tight.

3 The Limits of Deterministic Steering

We begin our analysis by characterizing the performance of deterministic algorithms. Recall that a deterministic algorithm is a tree of partitions, specifying the platform's next partition given any history of user's choices. The space of algorithms is therefore quite large. In any given recommendation step, the algorithm chooses one element of the set of partitions of n items, whose cardinality (known as the Bell number) grows faster than n^n.

Our first observation instead simplifies the space of strategies we need to consider. We focus on the following class of partitions that we call *recommendations*.

Definition 2. A recommendation is a partition of the form $\mathcal{P} = \{\{i\}, S \setminus \{i\}\}$, where S is the current active set.

In words, in a recommendation the platform isolates a single item i and asks the user, "Do you want this item, or would you like something else?" If the user selects $\{i\}$, the process ends. Otherwise, the conversation continues. The platform's problem of searching over recommendations is clearly much simpler, as the number of recommendations grows linearly in n. We have already observed in the previous section that the omniscient benchmark only requires a single recommendation. As it turns out, recommendations are entirely *sufficient* for implementing the optimal algorithm. In this sense, our paper provides a microfoundation for recommender systems as conversation-steering techniques. To show this result, in all our arguments we develop algorithms that rely solely on recommendations and we then prove that choosing any other partition at any stage of the algorithm yields no higher competitive ratio.

Naturally, the central weakness of deterministic algorithms is their predictability. Since v is unknown to the platform, any deterministic recommendation can be exploited by an adversarial realization of values. We formalize this in Theorem 1.

Theorem 1 (Deterministic Worst-Case)**.** *For any deterministic algorithm P that does not recommend item 1 in the first round there exists a valuation profile v such that the regret ratio is:*

$$\frac{\pi(P,v)}{\pi^*(v)} = \frac{\pi_n}{\pi_1}.$$

Proof. Consider the value profile v^1 that assigns value $\frac{1}{n-1}$ to the first item and value 1 to item n, and zero value to all other items. By Proposition 1, the only implementable items are item 1 and item n. Suppose algorithm P chooses some partition $\mathcal{P} = \{P_1, P_2, \dots, P_k\}$ different than $\mathcal{P}_1 = \left\{\{1\}, S \setminus \{1\}\right\}$, and WLOG assume $1 \in P_1$ and $n \in P_k$. If, by contradiction, algorithm P had a competitive ratio different than $\frac{\pi_n}{\pi_1}$, then it must be that $\pi(P, v^1) = \pi_1$. Hence, $v(P_1) \geq v(P_i)$ for all $i = 2, \dots, k$. In particular, it must be that $v(P_1) \geq v(P_k)$. But $v(P_1) \leq \frac{1}{n-1}$, with equality only if P_1 is a singleton, and $v(P_k) \geq \frac{1}{n-1}$, with equality only if $P_k = S \setminus \{1\}$, which is a contradiction. □

Any recommender that does not recommend item 1 in the first round incurs the worst possible competitive ratio. Suppose instead that the platform commits to algorithm P, which recommends item 1 (the most profitable item) in the first round. Then, consider a profile of values v such that the value of item n is 1 and item 2 has value $\frac{1}{n-1}$. Every other item has value 0 to the user. Then, by Proposition 1 only items 2 and n are implementable. When P recommends item 1, the user rejects it. With the smaller active set S, the set of implementable items has now shrunk. Item 2 no longer beats the average value, and hence any partition offered by P will eventually result in the user choosing item n, so $\pi(P, v) = \pi_n$. The omniscient benchmark could have implemented item 2 immediately, and would have received payoff π_2, hence the competitive ratio of this policy is $\frac{\pi_n}{\pi_2} > \frac{\pi_n}{\pi_1}$.

The best regret ratio we can hope to achieve with a deterministic algorithm is therefore $\frac{\pi_n}{\pi_2}$. In settings where the platform's value gap is large (i.e., $\pi_n \ll \pi_2 \leq \pi_1$), the competitive ratio of any deterministic policy can be arbitrarily close to zero. This motivates the need for randomization.

4 Optimal Randomized Steering

To overcome the fragility of deterministic policies, we allow the platform to randomize its questions. By mixing between different recommendations, the platform can "hedge" its bets, ensuring that it captures high-profit items with some probability while protecting against the worst-case scenarios where those items are rejected.

A randomized algorithm selects a partition distribution at each step. The *optimal policy* is the randomized algorithm p that maximizes the competitive ratio

$$\max_p \min_v \frac{\mathbb{E}_{P \sim p}[\pi(P, v)]}{\pi^*(v)}.$$

Denote by p_i be the probability that algorithm p recommends item i in the first round.

To characterize the optimal policy, we first define a set of adversarial profiles that the algorithm needs to hedge against. Consider for each i the profile $v^{(i)}$, such that

$$\begin{cases} v_i^{(i)} = \frac{1}{n-1} \\ v_n^{(i)} = 1 \\ v_j^{(i)} = 0 \text{ for all } j \notin \{i,n\}. \end{cases}$$

Under profile $v^{(i)}$ only items i and n are implementable. The omniscient benchmark always selects item i for a payoff $\pi^*(v^{(i)}) = \pi_i$, hence the optimal algorithm's performance is $\frac{p_i\pi_i+(1-p_i)\pi_n}{\pi_i} = p_i + (1-p_i)\frac{\pi_n}{\pi_i}$. We claim that any optimal algorithm's first round recommendation must be the solution to the following linear program:

$$\begin{aligned} \text{Maximize} \quad & \gamma \\ \text{Subject to} \quad & p_i + (1-p_i)\frac{\pi_n}{\pi_i} \geq \gamma \quad \forall i \in \{1,\dots,n\} \\ & \sum_{i=1}^{n} p_i = 1 \\ & p_i \geq 0 \quad \forall i \in \{1,\dots,n\} \end{aligned}$$

In particular, the first constraint ensures that the competitive ratio is at least γ for each worst-case profile $v^{(i)}$.[4] To maximize γ, the solution will satisfy these constraints with equality for items in the support, as described by the result below. In particular, this implies that $p_1 \geq p_2 \geq \cdots \geq p_k > 0$ for some $k \leq n$, and $p_i = 0$ for all $i > k$.

Theorem 2 (Optimal Randomized Algorithm)**.** *The optimal randomized algorithm p recommends an item $i \in \{1,\dots,k\}$ in the first round with probability p_i, where the probabilities satisfy the following equalizer condition for a constant γ:*

$$p_i + (1-p_i)\frac{\pi_n}{\pi_i} = \gamma \quad \textit{for all } i \in \{1,\dots,k\}. \tag{1}$$

Additionally, $p_i = 0$ and $\frac{\pi_n}{\pi_i} \geq \gamma$ for all $i > k$. The value γ is the maximum competitive ratio achievable.

Proof. First, we prove that this algorithm indeed guarantees a competitive ratio of at least γ.

[4]The reader familiar with linear programming will recognize that the worst-case profiles $v^{(i)}$ are exactly the dual variables for these constraints.

Let v be an arbitrary valuation profile, and let i be the item selected by the omniscient benchmark, i.e. $\pi^*(v) = \pi_i$. By Proposition 1 we know that $v_i \geq v(I)$. Our algorithm p recommends item i with probability p_i, hence the payoff from the algorithm is lowerbounded as

$$\mathbb{E}_{P\sim p}[\pi(P,v)] \geq p_i\pi_i + (1-p_i)\pi_n.$$

This is because in the worst case, once the first recommendation is made, the only implementable item is item n. Then the competitive ratio is lowerbounded by

$$\gamma(p) = \frac{\mathbb{E}_{P\sim p}[\pi(P,v)]}{\pi^*(v)} \geq \frac{p_i\pi_i + (1-p_i)\pi_n}{\pi_i} = p_i + (1-p_i)\frac{\pi_n}{\pi_i} \geq \gamma,$$

where the last equality is by construction of the optimal algorithm in the theorem's statement.

Second, we prove that no other randomized algorithm q can achieve a better competitive ratio. Let q_i be the recommendation probabilities in the first round of algorithm q. Suppose by contradiction that $\gamma(q) > \gamma$, that is, that q has a better competitive ratio than the algorithm in the theorem's statement. In particular, q's performance on each profile $v^{(i)}$ must be strictly better than p's performance on each, hence

$$q_i + (1-q_i)\frac{\pi_n}{\pi_i} > \gamma$$

in particular for all $i \leq k$. Rearranging, we obtain

$$q_i\left(1 - \frac{\pi_n}{\pi_i}\right) > \gamma - \frac{\pi_n}{\pi_i} \iff$$
$$\iff q_i > \frac{\gamma\pi_i - \pi_n}{\pi_i - \pi_n} = p_i$$

where the last equality comes from the theorem's statement. Hence, for all $i \leq k$ we have $q_i > p_i$.

Its expected payoff ratio against $v^{(i)}$ is clearly

$$\frac{\mathbb{E}_{Q\sim q}[\pi(Q, v^{(i)})]}{\pi^*(v^{(i)}} = \frac{q_i\pi_i + (1-q_i)\pi_n}{\pi_i} = q_i + (1-q_i)\frac{\pi_n}{\pi_i}.$$

The competitive ratio of q is then no better than the performance on the hardest of these instances, i.e.

$$\gamma(q) \leq \min_{i\in I}\left(q_i + (1-q_i)\frac{\pi_n}{\pi_i}\right).$$

Suppose by contradiction that $\gamma(q) > \gamma$, that is, that q has a better competitive ratio than the algorithm in the theorem's statement. But then,

$$\sum_{i=0}^{n} q_i \geq \sum_{i=0}^{k} q_i > \sum_{i=0}^{k} p_i = 1,$$

so q does not induce a well-defined distribution over first-round recommendations, which is a contradiction. □

Interestingly, the optimal algorithm's performance is fully determined by the first round distribution over items. That is because the worst-case profiles $v^{(i)}$ are such that the first round recommendation is all that is necessary.

To close the loop we prove that any algorithm that in the first round presents the user with a non-recommendation partition suffers a worse competitive ratio.

Lemma 1. Any algorithm q placing positive probability on a deterministic, non-recommender algorithm in the first round has competitive ratio larger than γ.

Proof. Take any deterministic algorithm Q such that, in the first round, the partition proposed to the user is $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_j\}$ and it is not a recommendation. Suppose that $n \in Q_j$. Because it is not a recommendation, it must be that $|Q_j| < n-1$, hence $v(Q_j) > \frac{1}{n-1}$ for all worst-case profiles $v^{(i)}$. This means that the algorithm never achieves a better payoff than π_n on any worst-case profile whenever the first round partition is not a recommendation.

Let $1 - \tilde{q}$ be the total probability assigned by algorithm q to all non-recommender algorithm. On the worst-case profiles $v^{(i)}$ the competitive ratio of q must be at most

$$(1-\tilde{q})\frac{\pi_n}{\pi_i} + \tilde{q}\tilde{\gamma}$$

where $\tilde{\gamma}$ is the competitive ratio achieved by the recommenders conditional on being selected by the randomization device.

If by contradiction q had better competitive ratio than p, then

$$p_i + (1-p_i)\frac{\pi_n}{\pi_i} = \gamma < \gamma(q) \leq \tilde{q}\tilde{\gamma} + (1-\tilde{q})\frac{\pi_n}{\pi_i}.$$

Let q_i' be the probability that the recommender portion of algorithm q recommends item

i in the first round. We can rewrite the above inequality simply as

$$\begin{aligned} p_i + (1-p_i)\frac{\pi_n}{\pi_i} < \tilde{q}\left(q_i' + (1-q_i')\frac{\pi_n}{\pi_i}\right) + (1-\tilde{q})\frac{\pi_n}{\pi_i} &= \\ = \tilde{q}q_i' + \tilde{q}(1-q_i')\frac{\pi_n}{\pi_i} + \frac{\pi_n}{\pi_i} - \tilde{q}\frac{\pi_n}{\pi_i} &= \\ = \tilde{q}q_i' + (1-\tilde{q}q_i')\frac{\pi_n}{\pi_i}. \end{aligned}$$

Clearly, since $\pi_i > \pi_n$, this inequality is equivalent to $\tilde{q}q_i' > p_i$, which must hold for all i in the support of the optimal policy. In particular then, summing over i we get $\sum_i \tilde{q}q_i' = \tilde{q}\sum_i q_i' > \sum_i p_i = 1$. Since q_i' are probabilities, they sum to 1, so $\tilde{q} > 1$. But $\tilde{q}$ was a probability mass itself, hence $1 < \tilde{q} \leq 1$ which is a contradiction. □

4.1 Performance Guarantee

The optimal policy's randomization yields a significant improvement over the deterministic case. We have shown that the deterministic policy has arbitrarily low competitive ratio. Instead the randomized policy achieves a uniform guarantee that depends only on the number of items.

Theorem 3 (Bounded Competitive Ratio)**.** *The optimal randomized policy achieves a competitive ratio of at least $\frac{1}{n-1}$. That is, for any valuation profile v,*

$$\frac{\mathbb{E}_{P\sim p}[\pi(P,v)]}{\pi^*(v)} > \frac{1}{n-1}.$$

The bound is tight.

Proof. The proof relies on the optimality conditions from the LP formulation of the optimal policy. If we denote $\frac{\pi_n}{\pi_i}$ by x_i, the equalizer condition implies we can write the probabilities p_i for $i = 1,\dots,k$ as $p_i = \frac{\gamma - x_i}{1-x_i}$. The well-definiteness condition then requires these probabilities to sum up to one:

$$\sum_{i=0}^{k} \frac{\gamma - x_i}{1-x_i} = 1.$$

Now, isolating γ, we get

$$1 = \sum_{i=0}^{k} \frac{\gamma - x_i}{1-x_i} = \sum_{i=0}^{k}\left(\frac{\gamma-1}{1-x_i} + 1\right) = (\gamma-1)\sum_{i=0}^{k}\frac{1}{1-x_i} + k$$

and hence

$$1-\gamma = \frac{k-1}{\sum_{i=0}^{k} \frac{1}{1-x_i}}.$$

The minimal value of γ that satisfies this equation is achieved by minimizing the denominator of the right hand side, $\sum_{i=0}^{k} \frac{1}{1-x_i}$. Note that each x_i is between 0 and 1, so $\frac{1}{1-x_i} > 1$. This sum can then never be smaller than the number of individual terms, k. Substituting this back, we get

$$1-\gamma = \frac{k-1}{\sum_{i=0}^{k} \frac{1}{1-x_i}} < \frac{k-1}{k}$$

so $\gamma > \frac{1}{k}$. The support size of the randomization is at most $n-1$, because the last item can always be achieved in later stages, hence $\gamma > \frac{1}{n-1}$.

To prove tightness, fix π_n and consider the limit where $\pi_1 = \cdots = \pi_{n-1} = M$ and $M \to \infty$. Then, $x_i \to 0$ for all $i < n$. The denominator approaches $k = n-1$ and γ approaches $\frac{1}{n-1}$. □

Intuitively, the worst possible case for the platform is when all items have similarly high profitability π except for the last one. Then, the best the platform can do is randomize uniformly among the $n-1$ items, and settle for item n if she doesn't get lucky in the first period.

This result demonstrates that by strategically pooling items into a "something else" option and randomizing the order of recommendations, the platform can secure a constant fraction of the optimal value, regardless of the user's private preferences. In other words, the optimal randomized steering takes the form of a *recommender*.

4.2 Sequential Optimality

The optimal randomized algorithm p described in Theorem 2 is not unique: its regret guarantee depends only on the algorithm's first-round partition and is agnostic about what it does thereafter. This is an artifact of the worst-case profiles $v^{(i)}$, where only the first recommendation has any hope of steering the user away from item n. A designer who aims to maximize the algorithm's competitive ratio is hence indifferent to the algorithm's behavior after the first round.

To select among these optimal algorithms, let us consider an adaptive notion of competitive ratio. To understand the idea, assume that the user rejects the recommended item i in the first round. If we were in the ex-ante worst-case scenario, nothing the platform does will steer the user away from item n. Yet, the platform may worry about its competitive ratio at this decision node. That is, the platform may worry that the true profile v is not one of the worst-case profiles $v^{(j)}$ for $j \neq i$. The true profile v may allow an optimal algorithm to implement an item $j < n$ at this stage, and the platform should try to capture that possibility.[5]

Formally, let us restrict our attention to recommendation policies, where at all rounds the platform recommends a singleton against the remaining set of items. A history $h_t = (r_1, r_2, \ldots, r_t)$ is an ordered sequence of distinct items $r_k \in I$ that have been recommended by the platform and rejected by the user, up to round t. The set of all histories is $\mathcal{H}_{\text{all}}$, and the empty history is denoted by $h_0 = \emptyset$. Denote by $h|_i$ the restriction of history h to the first i rounds, where $i < |h|$. Histories can then be identified with the set of rejected items, and the set of active items at history h is denoted by $S(h) = I \setminus h$. Histories constrain the possible valuations. A valuation profile v is consistent with history h if $v_{r_i} < v(S(h|_{i-1}))$ for any $i \in h$. The set $V(h)$ is the set of consistent valuation profiles, and the set of reachable histories is $\mathcal{H} = \{h \in \mathcal{H}_{\text{all}} \mid V(h) \neq \emptyset\}$.

Using this formalism, a randomized algorithm is a mapping $p \colon \mathcal{H} \to \Delta(I)$ such that for any history h the support of $p(h)$ is contained in the active set $S(h)$. We denote the probability that this algorithm recommends item i at history h as $p(h)_i$. Given an algorithm p and a valuation profile v, the conversation induces a probability distribution over terminal outcomes, i.e. over items eventually chosen by the user. We denote this random variable by $\mathbf{p}$. The payoff of the algorithm can then be written as

$$\mathbb{E}[\pi(\mathbf{p}, v)].$$

In particular, the conditional expected payoff of the algorithm is simply

$$\mathbb{E}[\pi(\mathbf{p}, v) \mid h].$$

We are now ready to state our definition of sequential optimality.

Definition 3. An algorithm p is sequentially optimal if for every history $h \in \mathcal{H}$ it maxi-

[5]The idea of hedging against such a moving target comes from viewing the decision maker as a pessimist. The decision maker worries about the worst-case scenario, even though the true state of the world may be a completely benign profile v. As decisions are made, the omniscient benchmark evolves and hence the "best" move for the algorithm moves as well. This reflects exactly the notion of subgame perfect equilibrium in a sequential-move zero-sum game. For a similar decision problem, see Malladi (2022) and Banchio and Malladi (2024)

mizes the worst-case conditional competitive ratio

$$p \in \underset{p}{\operatorname{argmax}} \min_{v \in V(h)} \frac{\mathbb{E}[\pi(\mathbf{p}, v) \mid h]}{\pi^*(v, h)}$$

where $\pi^*(v,h)$ is the highest implementable payoff π_j such that $j \in S(h)$.

That is, a sequentially optimal randomized algorithm is an optimal randomized algorithm that, after every user decision, maximizes the designer's competitive ratio going forward. By the same argument as in Theorem 2, an optimal randomized algorithm when the user rejects item i will recommend one of the k' most profitable items in $I \setminus \{i\}$, and it will assign recommendation probabilities according to a similar linear program as the one above. Repeating this step after each rejection yields the sequentially optimal randomized algorithm.

Theorem 4 (Sequentially Optimal Randomized Algorithm)**.** *The sequentially optimal randomized algorithm p*

1. *recommends an item $i \in \{1, \ldots, k\}$ in the first round with the same probability p_i as any optimal randomized alogrithm.*

2. *If the user rejects the recommendation, the algorithm proceeds recursively on the remaining set.*

3. *After each user rejection, the algorithm expands to recommend less profitable items (the threshold index k increases with the round number), the recommendation probability for each remaining item increases, and the contigent competitive ratio improves.*

Proof. It is straightforward to see that p with properties 1 and 2 is sequentially optimal. We show 3. Assume that in round t, the active set is I^t, the threhold index is k^t. In round t, the optimal recommendation probabilities $(p_i^t)_{i \in I^t}$ satisfy that

$$(p_i^t)_{i \in I^t} \in \arg \max_{p, \gamma:\, \sum_{i \in I^t} p_i = 1, \text{ and for all } i \in I^t,\, p_i + (1-p_i)\frac{\pi_n}{\pi_i} \geq \gamma} \gamma. \tag{2}$$

and the contingent regret guarantee

$$\gamma^t = p_i^t + (1 - p_i^t)\frac{\pi_n}{\pi_i}, \text{ for any } i \in I^t \text{ and } i \leq k^t. \tag{3}$$

For each $\gamma > 0$, denote the feasibility set in round t $\mathcal{F}^t(\gamma) := \{p : \sum_{i \in I^t} p_i = 1, \text{ and for all } i \in I^t,\ p_i + (1-p_i)\frac{\pi_n}{\pi_i} \geq \gamma\}$.

Assume that an item $j \in I^t$ is recommended and the user rejects it, the platform chooses recommendation probabilities $(p_i^{t+1})_{i \in I^t}$ such that

$$(p_i^{t+1})_{i \in I^t/\{j\}} \in \arg \max_{p,\gamma:\, \sum_{i \in I^t/\{j\}} p_i = 1, \text{ and for all } i \in I^t/\{j\},\, p_i + (1-p_i)\frac{\pi_n}{\pi_i} \geq \gamma} \gamma. \tag{4}$$

For each γ, denote the feasibility set in round $t+1$ by $\mathcal{F}^{t+1}(\gamma)$. For each $p \in \mathcal{F}^t(\gamma)$, let

$$\hat{p}_i = \begin{cases} p_i + \frac{p_j}{|I_t|-1}, & p_i \geq 0, i \neq j \\ 0, & p_i = 0, \text{ or } i = j \end{cases} \tag{5}$$

Then for each $i \in I^t/\{j\}$, $\hat{p}_i + (1-\hat{p}_i)\frac{\pi_n}{\pi_i} \geq \gamma$, so $\hat{p} \in \mathcal{F}^{t+1}(\gamma)$. Hence the contingent regret guarantee in round $t+1$, $\gamma^{t+1} \geq \gamma^t$. By the equalizer condition (Equation (1)), the threshold index k weakly increases and recommendation probabilities for each item i in the active set $I^t/\{j\}$ weakly increases with a strictly positive increase for some items. □

5 Extension

The model relies on two simplifying assumptions. First, we normalize the user's outside option (e.g., exiting the platform and buying elsewhere) to zero. This implies full participation: regardless of how aggressively the platform steers, the user never prefers to leave, so steering affects only which item is chosen rather than whether a choice is made on-platform. Second, we assume the platform cannot hide items and can only design information through partitions. In the first round the user observes a partition of the full item set I (not a strict subset), and in each subsequent round observes a partition of the subset selected in the previous round. In this section we relax both assumptions by allowing (i) a strictly positive outside option that can induce exit, and (ii) platform policies that can restrict the displayed set in addition to coarsening information.

5.1 Outside Option

Assume that the user has a strictly positive outside option $u > 0$, which is private information to the user. After observing any partition P, the user exits the platform (and the platform obtains payoff 0) if for all $S \in P$, $v(S) < u$.

With positive user outside option, there is an additional set of adversarial profiles that the algorithm needs to hedge against in addition to $(v^{(i)})_{i \in I}$. Consider for each i the profile

$v_0^{(i)}$, such that

$$\begin{cases} v_i^{(i)} = u \\ v_j^{(i)} = 0 \text{ for all } j \neq i. \end{cases}$$

Under profile $v_0^{(i)}$ only items i is implementable. If the user is presented with a partition P such that $\{i\} \in P$, then the user stays on the platform and chooses i; otherwise, the user leaves. In order to hedge against $(v_0^{(i)})_{i \in I}$, the optimal algorithms need another class of partitions that we call *lists*.

Definition 4. A list is a partition $\mathcal{P}$ such that $|\mathcal{P}| \geq 3$ and $\mathcal{P}$ that contains at least one singleton, i.e., there exists i such that $\{i\} \in \mathcal{P}$.

In words, in a list the platform shows the user at least 3 options with one of them being a specific item. Note that by our definition, a recommendation is not a list since a recommendation has size equal to 2. We say an algorithm lists item i if there is positive probability the algorithm will show a list containing $\{i\}$ to the user.

If an algorithm p recommends i with probability p_i and lists i with probability p_i', then the algorithm's performance for profile $v^{(i)}$ is $\frac{p_i \pi_i + (1-p_i)\pi_n}{\pi_i} = p_i + (1-p_i)\frac{\pi_n}{\pi_i}$ and its performance for profile $v_0^{(i)}$ is $\frac{(p_i + p_i')\pi_i + (1-p_i-p_i')0}{\pi_i} = p_i + p_i'$. We claim that any optimal algorithm's first round partition must be the solution to the following linear program:

$$\begin{aligned} \text{Maximize} \quad & \gamma \\ \text{Subject to} \quad & p_i + (1-p_i)\frac{\pi_n}{\pi_i} \geq \gamma \quad \forall i \in \{1, \ldots, n\} \\ & p_i + p_i' \geq \gamma \quad \forall i \in \{1, \ldots, n\} \\ & p_i' \leq p' \quad \forall i \in \{1, \ldots, n\} \\ & p' + \sum_{i=1}^{n} p_i = 1 \\ & p_i, p_i' \geq 0 \quad \forall i \in \{1, \ldots, n\} \end{aligned} \tag{6}$$

Again, the first two constraints ensure that the competitive ratio is at least γ for each worst-case profile $v^{(i)}$ and $v_0^{(i)}$. The third and fourth constraints are feasibility constraints: p' is total probability that a non-recommendation partition is presented in the first round; the third contraint requires each item i is listed with probabiliy lower than the total probabiliy of a non-recommendation partition is presented.

To maximize γ, the solution will satisfy these constraints with equality for items in the support, as described by the result below. In particular, this implies that $p_1 \geq p_2 \geq \cdots \geq$

$p_k > 0$ for some $k \leq n$, $p_i = 0$ for all $i > k$; and $p'_i = p' = 1 - \sum_i p_i$ for all $i > k$, $p'_i \in [p' - p_i, p']$ for all $i \leq k$.

Theorem 5 (Optimal Randomized Algorithm with Positive User Outside Option)**.** *The optimal randomized algorithm p recommends an item $i \in \{1, \dots, k\}$ in the first round with probability p_i, lists all items $j \in \{k+1, \dots, n\}$ in the first round with probability p', and lists an item $i \in \{1, \dots, k\}$ in the first round with probability $p'_i \in [p' - p_i, p']$. The probabilities $(p_i)_{i \leq k}$ and p' satisfy the following equalizer condition:*

$$p_i + (1 - p_i)\frac{\pi_n}{\pi_i} = p' \quad \text{for all } i \in \{1, \dots, k\}. \tag{7}$$

Additionally, $p_i = 0$ and $\frac{\pi_n}{\pi_i} \geq p'd$ for all $i > k$. The value p' is the maximum competitive ratio achievable and has the following expression

$$p' = \frac{1 + \sum_{i \leq k} \frac{\pi_n}{\pi_i - \pi_n}}{1 + \sum_{i \leq k} \frac{\pi_i}{\pi_i - \pi_n}}$$

where the optimal threshold index k

$$k \in \arg\max_{k' \leq n} \frac{1 + \sum_{i \leq k'} \frac{\pi_n}{\pi_i - \pi_n}}{1 + \sum_{i \leq k'} \frac{\pi_i}{\pi_i - \pi_n}}.$$

Proof. Let the value for the linear programm Equation (6) be γ^*. We first show that γ^* is the maximum competitive ratio. Then we show that the any algorithm p satisfying the conditions in the theorem solves the linear programm Equation (6) and achieves maximum competitive ratio γ^*.

Let q be any algorithm with recommendation probabilities $(q_i)_{i \in I}$ and listing probabilities $(q'_i)_{i \in I}$ such that $\gamma(q) > \gamma^*$. Then it has to be that for all $i \in I$, the competitive ratio of q against profile $v^{(i)}$ is strictly higher than γ^*, that is,

$$q_i + (1 - q_i)\frac{\pi_n}{\pi_i} > \gamma^*,$$

and the competitive ratio of q against profile $v_0^{(i)}$ is also strictly higher than γ^*, that is

$$q_i + q'_i > \gamma^*.$$

In addition, q is feasible such that $\sum_i q_i + \max_i q'_i \leq 1$. Then the probabilities $(q_i)_{i \in I}$, $(q'_i)_{i \in I}$ and $q' = \max_i q'_i$ are a feasible solution to the linear program (6) and hence γ^* cannot be the value of the linear program (6), a contradiction.

Instead, let us show that a p satisfying the condition (7) in the theorem solves the linear program (6). It is straightforward to see that the probabilities $(p_i)_{i\in I}$, p' and $(p'_i)_{i\in I}$ are a feasible solution to (6), and p' is the value achieved. We show that $p' = \gamma^*$.

Suppose by contradiction that $p' < \gamma^*$. Let $(q_i)_{i\in I}$, $(q'_i)_{i\in I}$ and $q' = \max_i q'_i$ be the probabilities that achieve $\gamma^* > p'$. Then it has to be that for each $i \in \{1,\ldots,k\}$

$$q_i + (1-q_i)\frac{\pi_n}{\pi_i} > p',$$

and for each $i \in \{k+1,\ldots,n\}$

$$q_i + q'_i > p'. \tag{8}$$

Combining with condition (7), the first set of inequalities imply that for each $i \in \{1,\ldots,k\}$, $q_i > p_i$. This also implies that

$$\sum_{i\geq k+1} q_i + q' = 1 - \sum_{i\leq k} q_i < 1 - \sum_{i\leq k} p_i = p'.$$

Hence for all $i \in \{k+1,\ldots,n\}$, it has to be that

$$q_i + q'_i \leq \sum_{i\geq k+1} q_i + q' < p'$$

which contradicts condition (8).

Finally, the expressions for p' and k come from Equation (7). □

5.2 Hiding items

Let us instead relax the implicit restriction that the platform cannot hide items. In the first round, the platform may present a partition of a strict subset of items $S \subset I$. In each subsequent round, it may present a partition of a strict subset of the set selected by the user in the previous round.

When the user has a exogeneous positive outside option that is private information, Theorem 5 implies that it is never optimal for the platform to hide any items. In the optimal policy, every item is presented to the user with a strictly positive probability. Intuitively, because the outside option is positive and unknown to the platform, withholding items increases the chance that the user exits; assigning positive probability to every item hedges against this exit risk (under which the platform earns zero payoff).

Assume now that the user has no exogenous positive outside option. In this case, the optimal algorithm depends on what the user knows about, and how she can access, hidden items. We begin with a trivial benchmark.

Theorem 6. *If the user knows nothing about hidden items (i.e. if the user simply takes the recommendations at face value), the optimal algorithm is to present only* 1 *in the first round.*

Proof. If the user remains on the platform regardless of the partitions presented, then by displaying only item 1 the platform guarantees that the user selects item 1. $\square$

This "no-churn" condition is satisfied, for example, when the user (i) has no outside option, and (ii) either has no information about hidden items (so she assigns them value 0), or has no access to hidden items outside of the platform.

Instead, suppose the user can observe the average value of the hidden items and will exit the platform whenever this value exceeds the value of every item (or bundle) displayed—for instance, because she can obtain the hidden items elsewhere. In this case it is again optimal for the platform to never hide items. Moreover, conditional on not hiding any items, any algorithm characterized by Theorem 2 remains optimal.

Theorem 7. *If the user can observe the average value of the hidden set and leaves whenever it exceeds the value of every displayed option, then the platform never hides any item, and the optimal algorithm coincides with that in Theorem 2.*

Proof. We only prove that in an optimal policy the platform does not hide items. Let A be any active set. Suppose the platform hides a set $\hat{S} \subset A$ and presents a partition P of $A \setminus \hat{S}$. Consider instead the partition $P' := P \cup \{\hat{S}\}$, i.e., the platform reveals $\hat{S}$ as an additional block. Fix any value profile v. If there exists $S \in P$ with $v(S) \geq v(\hat{S})$, then P and P' lead to the user choice and thus same outcome to the platform. If instead $v(S) < v(\hat{S})$ for all $S \in P$, then under P the user exits (since the hidden set dominates all displayed options), yielding payoff 0. Under P', the user can select $\hat{S}$ rather than exit, so the platform obtains a weakly higher payoff. Hence P' weakly dominates P for all v, implying that hiding items is never optimal. $\square$

6 Conclusions

In this paper, we formalized the problem of conversational steering as a sequential partition game between a user maximizing their own utility and a platform with a possibly misaligned objective function. Motivated by the rise of AI assistants that must balance user satisfaction with safety, fairness, or monetization goals, we sought to understand the theoretical limits of such guidance.

Our analysis yields three main insights. First, despite the double-exponential space of possible conversation trees (partitions), the optimal policy takes a surprisingly simple form: a randomized sequence of *recommendations*. The platform need not construct complex, coarse partitions of the search space; it is sufficient to sequentially propose specific items ("Do you want X, or something else?"). This theoretical finding offers a microfoundation for the ubiquity of "recommender systems" as the default mode of interaction in digital marketplaces.

Second, we characterized the limits of this steering. We proved that no deterministic policy can guarantee a non-zero competitive ratio against an adversarial user profile. However, by randomizing the order of recommendations, the platform can achieve a competitive ratio of $\frac{1}{n-1}$. This bound is tight and robust: it holds regardless of the user's private valuations. This result highlights a fundamental trade-off: to protect against worst-case alignment failures (where the user's best item is the platform's worst), the platform must concede a significant portion of potential surplus, specifically by randomizing among its own high-value items.

Finally, we extended our framework to sequentially optimal policies and settings with user outside options. We showed that the presence of an outside option forces the platform to diversify its "list" of recommendations, effectively curbing aggressive steering to prevent user churn.

As AI assistants evolve from passive search engines to active decision partners, understanding the economics of their guidance becomes paramount. Our work provides a baseline tractability result: effective steering does not require complex manipulation of the user's information set, but rather a carefully calibrated, randomized offering of specific choices. Future work might explore this dynamic under partial information, where the platform learns the user's preferences over time, bridging the gap between our prior-free benchmark and classical Bayesian learning.

References

Martino Banchio and Suraj Malladi. 2024. Search and Rediscovery. In *Proceedings of the 25th ACM Conference on Economics and Computation* (New Haven, CT, USA) *(EC

'24). Association for Computing Machinery, New York, NY, USA, 1194. doi:10.1145/3670865.3673452

Martino Banchio, Aranyak Mehta, and Andres Perlroth. 2025. Ads in Conversations. In *Proceedings of the 26th ACM Conference on Economics and Computation*. 350–350.

Dirk Bergemann, Marek Bojko, Paul Dütting, Renato Paes Leme, Haifeng Xu, and Song Zuo. 2024. Data-driven mechanism design: Jointly eliciting preferences and information. *arXiv preprint arXiv:2412.16132* (2024).

Kshipra Bhawalkar, Alexandros Psomas, and Di Wang. 2025. Sponsored Questions and How to Auction Them. *arXiv preprint arXiv:2512.03975* (2025).

Avinava Dubey, Zhe Feng, Rahul Kidambi, Aranyak Mehta, and Di Wang. 2024. Auctions with llm summaries. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 713–722.

Paul Duetting, Vahab Mirrokni, Renato Paes Leme, Haifeng Xu, and Song Zuo. 2024. Mechanism design for large language models. In *Proceedings of the ACM Web Conference 2024*. 144–155.

Paul Dütting, Michal Feldman, Inbal Talgam-Cohen, et al. 2024. Algorithmic contract theory: A survey. *Foundations and Trends® in Theoretical Computer Science* 16, 3-4 (2024), 211–412.

Bengt Holmstrom. 1980. *On The Theory of Delegation*. Technical Report. Northwestern University, Center for Mathematical Studies in Economics and ....

Eaman Jahani, Benjamin S Manning, Joe Zhang, Hong-Yi TuYe, Mohammed Alsobay, Christos Nicolaides, Siddharth Suri, and David Holtz. 2024. Prompt Adaptation as a Dynamic Complement in Generative AI Systems. *arXiv preprint arXiv:2407.14333* (2024).

Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, and Li Chen. 2021. A survey on conversational recommender systems. *ACM Computing Surveys (CSUR)* 54, 5 (2021), 1–36.

Jon Kleinberg and Robert Kleinberg. 2018. Delegated Search Approximates Efficient Search. In *Proceedings of the 2018 ACM Conference on Economics and Computation* (Ithaca, NY, USA) *(EC '18)*. Association for Computing Machinery, New York, NY, USA, 287–302. doi:10.1145/3219166.3219205

Sijie Lin. 2025. Learning to prompt: Human adaptation in production with generative ai. *Available at SSRN 5788402* (2025).

Suraj Malladi. 2022. Searching in the dark and learning where to look. *Available at SSRN 4084113* (2022).

Filip Radlinski and Nick Craswell. 2017. A Theoretical Framework for Conversational Search. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval* (Oslo, Norway) *(CHIIR '17)*. Association for Computing Machinery, New York, NY, USA, 117–126. doi:10.1145/3020165.3020183

Ermis Soumalias, Michael J Curry, and Sven Seuken. 2024. Truthful aggregation of llms with an application to online advertising. *arXiv preprint arXiv:2405.05905* (2024).

Tobias Werner, Ivan Soraperra, Emilio Calvano, David C Parkes, and Iyad Rahwan. 2024. Experimental evidence that conversational artificial intelligence can steer consumer behavior without detection. *arXiv preprint arXiv:2409.12143* (2024).