

# Leveraging Large Language Models in Conversational Recommender Systems

Luke Friedman\*, Sameer Ahuja, David Allen, Zhenning Tan, Hakim Sidahmed, Changbo Long, Jun Xie, Gabriel Schubiner, Ajay Patel, Harsh Lara, Brian Chu, Zexi Chen and Manoj Tiwari  
Google Research

## ABSTRACT

A Conversational Recommender System (CRS) offers increased transparency and control to users by enabling them to engage with the system through a real-time multi-turn dialogue. Recently, Large Language Models (LLMs) have exhibited an unprecedented ability to converse naturally and incorporate world knowledge and common-sense reasoning into language understanding, unlocking the potential of this paradigm. However, effectively leveraging LLMs within a CRS introduces new technical challenges, including properly understanding and controlling a complex conversation and retrieving from external sources of information. These issues are exacerbated by a large, evolving item corpus and a lack of conversational data for training. In this paper, we provide a roadmap for building an end-to-end large-scale CRS using LLMs. In particular, we propose new implementations for user preference understanding, flexible dialogue management and explainable recommendations as part of an integrated architecture powered by LLMs. For improved personalization, we describe how an LLM can consume interpretable natural language user profiles and use them to modulate session-level context. To overcome conversational data limitations in the absence of an existing production CRS, we propose techniques for building a controllable LLM-based user simulator to generate synthetic conversations. As a proof of concept we introduce RecLLM, a large-scale CRS for YouTube videos built on LaMDA, and demonstrate its fluency and diverse functionality through some illustrative example conversations.

## 1 INTRODUCTION

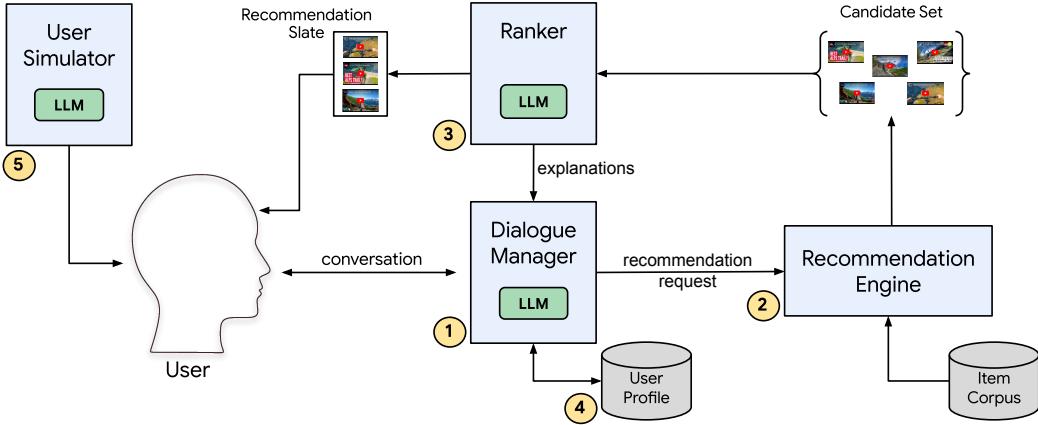
Recommender systems are one of the most prominent success stories of machine learning in industry, delivering personalized content to billions of users over a wide range of domains such as Search, Videos, News, and Shopping. Machine learning algorithms have transformed the way these systems are built within industry; in particular, over the last decade deep learning based systems that thrive in the large data regime have capitalized on the abundance of user interaction data available through products to learn sophisticated statistical correlations and better optimize for key engagement metrics [17]. However, despite the success of ML in this setting, this increasing reliance on implicit interaction signals like clicks as a proxy for user preference has its downsides as well. It is well documented that many modern large-scale recommender systems encounter problems like surfacing clickbait, propagating societal biases and polarization of the user base [25, 55, 104]. Recommender systems based on point-and-click interfaces also afford the user only a limited channel to communicate with the system and little opportunity to engage in any type of interactive exploration.

A Conversational Recommender System (CRS) gives users more control over their recommendations through the ability to engage in a real-time multi-turn dialogue [22, 37]. This enables the system to actively query the user instead of relying solely on prior behavior to infer preferences, and in response the user can provide feedback and refine suggestions over a series of turns. This new paradigm is a generalization of both recommender and classical search systems, in which typically users direct the system through a single-shot query. Now the system must explore cooperatively with the user and, to maintain a natural flow, sometimes even veer off into modes tangential to the core recommendation task such as undirected chitchat and question answering (see e.g. [9]). Oftentimes the CRS must also be multimodal, for instance displaying recommendation slates within a visual UI while simultaneously carrying on a conversation through a natural language interface (see e.g. [112]).

Although conversational recommender systems have existed in some form for decades [51, 85], the recent explosion of Large Language Models (LLMs) [6, 15, 86] unlocks new opportunities. LLMs have made a huge leap in the ability of machines to converse in a human-like way and can power the natural language interface between the user and the system. LLMs have also shown an unprecedented ability to draw on general world knowledge and utilize some level of common sense reasoning [34], which can be exploited in various ways within a CRS. For instance, we can try to use an LLM to directly reason about how well an item matches the context of a conversation within a ranking module and generate an intuitive natural language explanation as a byproduct. Other possible use cases for LLMs behind the scenes include dialogue management, incorporating natural language user profiles for better personalization and building realistic user simulators to generate synthetic data at scale for evaluation and tuning of system components.

Tantalizing as LLMs are as a tool for a CRS, new technical challenges must be overcome to leverage them effectively. For instance, LLMs are prone to hallucinations and grounding them remains a largely unsolved problem [40]. Also, one of the appeals of LLMs is their sense of naturalness and unpredictability, but when operating in a task-oriented setting this means that controlling an LLM can be more difficult than with a template based system. Particularly challenging in the recommendation setting is how to interface between the LLM and the underlying recommendation engine. One approach is to have the LLM *be* the recommendation engine in addition to its role as a dialogue agent (see e.g. [42]). However, for large-scale recommender applications the item corpus can contain millions or billions of always-changing items, making it challenging for an LLM to memorize the corpus within its parameters. Alternatively the LLM must somehow connect to an external recommendation engine or database, passing on relevant preference information.

\*Corresponding author: lbfried@google.com.



**Figure 1: Overview of key contributions from RecLLM.** (1) A dialogue management module uses an LLM to converse with the user, track context and make system calls such as submitting a request to a recommendation engine all as a unified language modeling task. (2) Various solutions are presented for tractable retrieval over a large item corpus within an LLM-based CRS. (3) A ranker module uses an LLM to match preferences extracted from the context of the conversation to item metadata and generate a slate of recommendations that is displayed to the user. The LLM also jointly generates explanations for its decisions that can be surfaced to the user. (4) Interpretable natural language user profiles are consumed by system LLMs to modulate session-level context and increase personalization. (5) A controllable LLM-based user simulator can be plugged into the CRS to generate synthetic conversations for tuning system modules.

While studied recently [8, 73], this approach is yet to be solved in the general large-scale recommendation setting.

In this paper we provide a roadmap for leveraging LLMs in a variety of ways to build a controllable and explainable large-scale CRS. Key contributions of the proposal are:

- A dialogue management module that reframes natural language generation, preference understanding, context tracking, and calls to a recommendation engine as a unified language modeling task performed by a single LLM.
- A general conceptual framework for performing retrieval with an LLM over a huge corpus of items. Various solutions are presented depending on efficiency requirements and what data and external APIs are available.
- A joint ranking / explanation module that uses an LLM to extract user preferences from an ongoing conversation and match them to textual artifacts synthesized from item metadata. As a byproduct of intermediate chain-of-thought reasoning [95], the LLM generates natural language justifications for each item shown to the user, increasing the transparency of the system.
- Incorporation of persistent, interpretable natural language user profiles as additional input to system LLMs, which supplements session-level context and improves the personalized experience.
- Techniques for building controllable LLM-based user simulators that can be used to generate synthetic conversations for tuning system modules.

As a proof of concept we introduce RecLLM, an LLM-based CRS for YouTube videos powered by LaMDA [86], and share some example conversations showing the fluency and diverse functionality

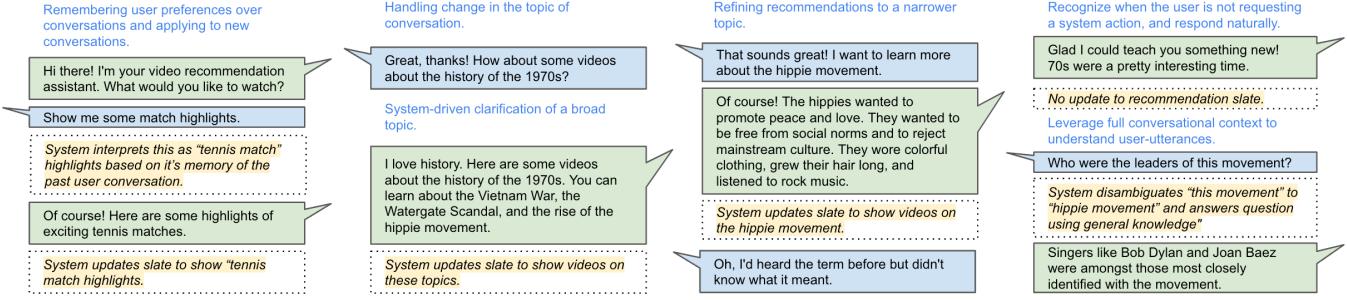
of the system. Our goal is to make a compelling argument for the promise and viability of LLM-based conversational recommender systems and to take a first step towards realizing this vision in practice.

## 2 PROBLEM SCOPE



**Figure 2: Screenshot of an LLM-based user simulator talking with RecLLM.**

In RecLLM, we represent the CRS in a multi-modal setup comprised of two components: A slate of recommendations and an



**Figure 3: RecLLM possesses many conversational capabilities such as the ability to retain context throughout a session, handle topic shifts and reference items from recommendation slates.**

ongoing conversation between the user and the conversational agent (see Figure 2). The user outputs a natural language message on their turn, and the agent responds with a natural language message, optionally updating the slate of recommendations based on the conversation. By separating dialogue from the recommendation slate we hope to more accurately reflect how a large-scale CRS would eventually look in a production setting.

Traditionally, users have interacted with recommender systems via user interface interactions such as viewing the recommended items, or marking recommendations as good or bad via interface widgets [20, 76]. Although currently in RecLLM we exclude these types of interactions we do not intend to replace them; our eventual goal is to augment them with the more expressive channel of natural language that allows users to better express nuance about their interests.

In terms of the item corpus, RecLLM recommends from the corpus of all public YouTube videos. We make this choice due to two characteristics that increase the applicability of the system to other real-world problems: One, unlike corpora of items that occur frequently in the LLM’s training data (e.g., movies and popular music), an LLM cannot feasibly be used to directly recommend YouTube videos and must interface with the corpus. Secondly, it’s a large-scale corpus, requiring a scalable approach to recommendations. A natural consequence of building such a system from scratch is that there are no logs of users interacting with this system to jumpstart training of the model(s). Although RecLLM focuses on YouTube videos, our intention in this paper is to outline a general approach that can be easily extended to many other domains.

While evaluating with initial testers, we found that users expect a CRS that pairs slate recommendations with natural language conversation to possess a wide range of conversational capabilities, such as retaining context, handling topic shifts and referencing slate items. RecLLM focuses on leveraging techniques that can scale over a broad number of these use-cases. In Figure 3 a few of the core conversational capabilities currently supported are demonstrated via a mock conversation.

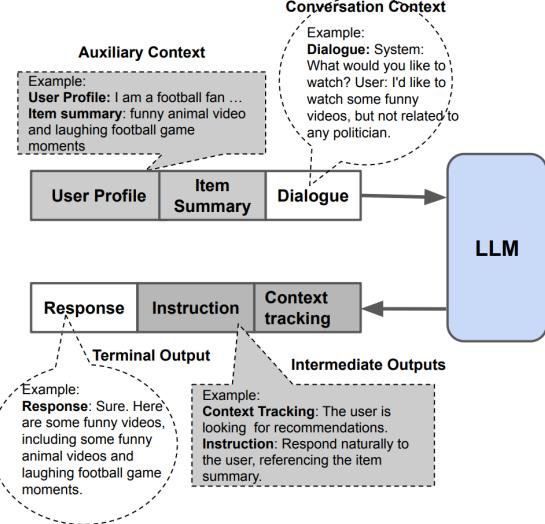
Finally, there are several problems that need to be addressed for conversational agents to become mainstream. These include safety of dialogue, debiasing, consistent personality of agents, etc. In this

work we do not attempt to tackle these problems directly, rather focusing on problems that are unique to the setting of conversational recommenders.

### 3 SYSTEM OVERVIEW

In this section we take a closer look at key system components of RecLLM (see Figure 1). In particular we focus on dialogue management, retrieval, ranking and explanations, and incorporation of natural language user profiles.

#### 3.1 Dialogue Management



**Figure 4: A unified LLM dialogue management module. An LLM takes as input the full session context and outputs a sequence of messages ending in a terminal output that triggers a system action, such as a response to the user.**

Dialogue management is the central module of a CRS, acting as the interface between the user and the rest of the system. It is responsible for guiding the user through a multi-turn exploration of the recommendation corpus and generating sensible, useful, and

grounded responses at each turn. In the process it must either implicitly or explicitly perform context tracking to extract useful representations of user preferences and intents. This information can be used to inform the dialogue policy and also as the basis for outputting API calls to initiate system actions (e.g. by sending a search query to a recommendation engine backend, see Section 3.2.1). From an end-to-end point of view, given context information (dialogue history, a user profile, item summaries, etc.), the goal of the dialogue manager is to generate system actions to take, as well as an appropriate system utterance.

There are extra challenges and requirements to dialogue management in the context of conversational recommenders:

- **Control:** In contrast to open-ended dialogue, a CRS dialogue manager must actively work with the user to explore the recommendation corpus. This entails a mixed-initiative setup where the system must respond to user requests and also at times actively steer the conversation in a specific direction. For instance, preference elicitation—in which the system must figure out when and how to best query the user in order to extract maximal information about their preferences—is an entire subfield of CRS dialogue management [11, 74, 83, 112].
- **Ambiguity:** Compared to task-oriented dialogue there is no clear cut measure of success for a CRS dialogue manager. Although the system should try to ensure that the conversation does not get too far off track the core recommendation task, the goal is not necessarily to minimize the number of turns that it takes the user to find an acceptable item, but rather to provide an overall satisfactory exploratory experience (see, for instance [78]). This means that there is rarely a single objectively "correct" thing for a conversational system to say at any given time, nor an easily defined metric for whether the dialogue manager is doing a good job.
- **Grounding:** One of the main challenges of a CRS dialogue manager is to faithfully ground its responses to the user in the recommendation corpus. After returning a slate of recommendations, the system should be able to refer to the items in a relevant and factually correct way. Other sources of external information, such as long term preferences coming from a user profile, may also be injected and the dialogue manager should be able to incorporate them appropriately in the ongoing conversation.

Traditionally CRSs take a modular approach to dialogue management, where a hardcoded policy graph maps dialogue states (e.g. intent) to different system actions, such as whether to get a recommendation, ask a question, or chat casually. Natural language understanding models extract preferences and determine the dialogue states, and separate natural language generation models generate system responses. Alternatively, in some recent CRSs language models are tuned end-to-end directly to imitate dialogue collected from crowdsource workers, discarding any notion of dialogue states or internal structure.

In RecLLM we employ a single unified LLM to execute dialogue management purely in terms of language modeling. At each turn the LLM takes as input the prior conversation context along with

additional information like textual representations of recommendation slates and user profiles that are potentially injected from external sources. Like the end-to-end approach mentioned above, one of the distinguishing features of this architecture is that there no longer exists a hardcoded policy graph with fixed dialogue states. Instead, on a given system turn the LLM generates a sequence of natural language outputs that encapsulate all context tracking, intermediate reasoning, natural language generation, and API calls to the rest of the system. It is hardcoded that certain string patterns in outputs from the dialogue manager trigger system actions. For instance an output "Response: <message>" will cause *message* to be shown as a user facing response, and "Request: <query>" will cause *query* to be sent to the recommendation engine backend to retrieve a slate of recommendations. Other outputs of the LLM can function as chain-of-reasoning steps, instructions to itself to follow, or dialogue state tracking inferences. Unlike the system calls, there are no ingrained rules about the functionality of these intermediate outputs, and conventions about their use must be taught to the LLM either through in-context few-shot learning or tuning.

The advantage of this architecture over the modular approach is its simplicity and flexibility. In the modular approach, any new functionality such as the addition of a new user intent or dialogue state has to be engineered into the system, which is a serious impediment to scalability. The unified LLM architecture shifts the emphasis from engineering-driven to data-driven quality iteration. To fix a problem or introduce new capabilities, instead of engineering a new component a designer must now create examples that enable the LLM to learn the desired behavior. This also creates the potential for the dialogue manager to learn new policy states and useful dialogue state tracking artifacts through the generalization abilities of the LLM.

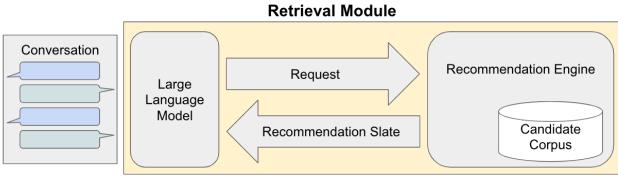
The main challenge to the unified LLM approach is how to effectively control the dialogue manager and guide it towards a reasonable dialogue policy without explicitly constraining it via hard rules. In our initial implementation we tune our unified LLM on a moderate number of manually generated examples. In this way we are able to establish some direction about the type of behavior and internal states we would like to see while still relying heavily on the ability of LLMs pretrained on dialogue data to converse naturally with only minimal supervision. Although we are able to build a functional dialogue manager this way, with only a limited amount of training examples it is difficult to teach the dialogue manager a sophisticated policy tailored to the conversational recommender domain. In Section 4.2 we discuss ideas for overcoming this limitation by tuning our dialogue manager and recommendation modules with larger amounts of synthetically generated data.

## 3.2 Recommendations and Refinement

Once triggered by the dialogue management module, it is the responsibility of the recommendation module to return a slate of high quality, relevant, and diverse recommendations that will be shown to the user. This can either be an initial recommendation slate or a refinement of an earlier slate from the session based on feedback from the user. A traditional recommender system chooses items by inferring preferences of the user from some type of user profile or dense representation built from historical data, possibly taking into

account other contextual factors (e.g. the location or time of day). In a search system, the user can supplement these implicit signals with explicit intents, usually through a simple static query. A primary challenge of a CRS is that now the user can express these explicit intents over the course of a full multi-turn conversation, which the recommendation module must understand and connect to the item corpus. Many traditional recommender systems employ a two stage pipeline, first retrieving candidate items and then ranking them [21, 54]. RecLLM follows this strategy, with the added twist that the ranker also jointly generates natural language explanations for why each item is being selected.

**3.2.1 Retrieval.** The purpose of the retrieval phase is to take the full corpus, which for some domains such as videos or urls may contain hundreds of millions of items, and based on the context select a small number of candidate items (e.g. 100) that will be fed to a downstream ranker. A key challenge of retrieval is to make this process tractable, as it is not computationally feasible to process each item independently at inference time. In Figure 5 we illustrate a general conceptual framework for retrieval in our problem setting. An LLM processes the session context and generates a request, either implicitly through a model activation layer or explicitly through its language output interface. A recommendation engine then uses a tractable search algorithm to retrieve candidates from the item corpus. In Table 1 we give a few illustrative examples of possible retrieval algorithms that fit into this framework, which we describe in more detail below.



**Figure 5: Overview of large-scale retrieval in an LLM-based CRS.**

Approach	Request Type	Tractable Search Algorithm
Generalized Dual Encoder Model	Internal LLM embeddings	KNN or ScaNN [30]
Direct LLM Search	Title or id	Fuzzy lookup
Concept Based Search	List of concepts	Concept Activation Vector [43]
Search API Lookup	Search query	Search API

**Table 1: Various possible solutions to large-scale retrieval in a CRS.**

**Generalized Dual Encoder Model.** A popular solution to retrieval in traditional deep learning based recommenders is to use a dual encoder model consisting of two neural net towers, one to encode the context and one to encode the items (see e.g [102] and Figure 10a). Item embeddings can be generated offline using the item tower and stored in an efficient data structure. An approximate nearest

neighbor lookup can then use the generated context embedding to perform a sub-linear time retrieval of item embeddings at inference time [98]. We can extend this approach for conversational recommenders by using an LLM as a context encoder that processes the full ongoing conversation between the user and system along with any other additional context information. In this case the request sent to the recommendation engine is an embedding, which can be generated by extracting and then projecting a suitable activation layer from the model.

One downside to this approach of pulling embeddings from the internals of an LLM is that it severely hampers our ability to learn a retrieval model in a sample efficient way. Dual encoder models trained from scratch require large amounts of training data to constrain the context tower embeddings to occupy the same subspace as the item tower embeddings. Sometimes it is possible to use pretrained embeddings on the item side (for instance by taking them from an existing production search or recommender system), but still the context embeddings must be tuned to align with the item embeddings to get good results. LLMs operate via a text-in / text-out interface and much of their power comes from the transfer learning afforded by knowledge gained through extensive pretraining. By leaving the level of language abstraction we are sacrificing much of this ability to generalize from a small amount of data.

**Direct LLM Search.** In this method the LLM directly outputs ids or titles of items to recommend as text. The tractable search algorithm is an exact or fuzzy match against items in the corpus and the recommendation engine plays no role beyond this simple matching. The LLM must learn to output these ids/titles through some combination of its pretraining and a corpus-specific fine tuning phase (see e.g [84]). Given the assumption that our system must be able to return slates from a fixed item corpus, this is the closest thing to having an LLM-based chatbot function directly as a CRS. The downside to this approach is that because only negligible work is being offloaded to the recommendation engine, the LLM must memorize information about the entire item corpus within its model parameters. For a large corpus this can be prohibitively expensive in terms of the model size and training data needed, and also makes it difficult to refresh the item corpus without retraining the LLM.

**Concept Based Search.** In this method the LLM outputs a list of concepts, which are then embedded and aggregated by the recommendation engine into a single context embedding. This is used to lookup items through approximate k-nearest neighbor search similar to the generalized dual encoder method. A technique like Concept Activation Vectors [43] can be used to perform this transformation from concepts to embeddings in the item space. The appeal of this approach is that extracting relevant concepts from a conversation is a natural task that can be taught to an LLM through in-context learning or tuning with a small number of examples. Also, because only item embeddings are needed (the concept embeddings are derived from these) if pretrained item embeddings can be borrowed from an existing source then no additional tuning of embeddings is required. However, one limitation is that lists of concepts are often a coarse representation of a conversation and similar to continuous bag-of-words methods [60] are lossy with

respect to word order and other nuances of language, which can negatively affect retrieval quality.

*Search API Lookup.* In this method, the LLM directly outputs a search query, which gets fed into a black-box search API to retrieve items. Unlike Concept Based Search, which is generic as long as item embeddings can be trained or reused, Search API Lookup is only applicable when such a search API already exists for the domain in question. However, when available, this type of API is often backed by a sophisticated search stack and can yield higher quality results. Analogous to Concept Based Search, in Search API Lookup the LLM can be taught to output relevant search queries using a small number of examples (see Section 3.1), but the quality of retrieval is limited by the extent to which a search query can properly represent the full context of a conversation.

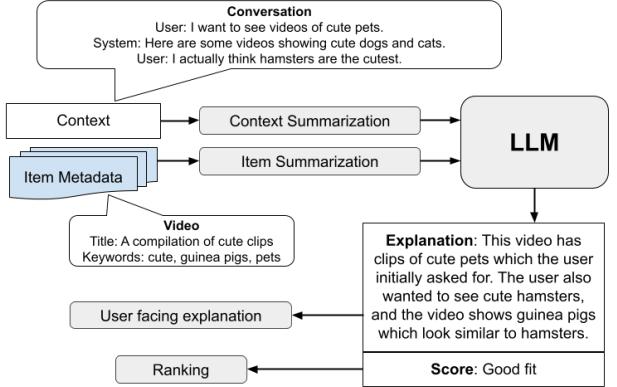
In Section 4.2 we build upon these methods by discussing options for tuning a retrieval model using large-scale synthetic data.

**3.2.2 Ranking / Explanations.** After candidate items have been retrieved, a ranker decides which of them will be included in the recommendation slate and in what order. Unlike the retrieval module, the ranking module does not need to perform tractable search over a large corpus and is therefore less constrained in the types of computation that are possible. In a traditional recommender system, this usually manifests in the ranker crossing context and item features (instead of processing them in separate towers as is done in a dual encoder) and potentially using custom ranking losses during training that directly compare candidate items [10]. In the case of RecLLM, we take advantage of this extra room for computation to use an LLM that reasons sequentially about how well an item matches the context and generates a rationalization for its decision as a byproduct.

Figure 6 gives a schematic for the LLM ranker. For each candidate item, the LLM jointly generates a score and a natural language explanation for the score<sup>1</sup>. These scores implicitly induce a ranking of the items. The first step is to create a text summarization of the item that fits into the context window of the LLM based on metadata associated with the item. In the case of a YouTube video recommender, this metadata consists of information such as the title, knowledge graph entities associated with the video, developer description of the video, transcript of the video, and user comments. In the future we would also expect a large multimodal model to directly process the raw video instead of relying only on textual artifacts. This item summarization can be done offline and is necessary in the case where the metadata is high volume (e.g. if we have thousands of user comments). We can view this summarization as a special case of the multi-document summarization problem [53]; it is also related to a main challenge of the user profile module (see Section 3.3), which must summarize large amounts of prior user data into a text format that can be passed into an LLM (or alternatively augment the LLM with the ability to access this information efficiently at inference time). There also can be a similar preprocessing step for summarizing the context information, although this must be done at inference time since

<sup>1</sup>There are many proposed solutions for enabling text in / text out LLMs to solve regression problems (i.e. output a score) [52]; within RecLLM we use the simple approach of bucketing the range of possible scores and having the LLM output a semantically meaningful phrase (e.g. "excellent fit") corresponding to a bucket id.

unlike for items we cannot enumerate all possible contexts and process them offline.



**Figure 6: A joint LLM ranking / explanation module.** The conversation is used as context for the user’s preferences and the video metadata is used as context for the item. The LLM takes in summaries of the item side and context side to produce a score for the item and an explanation for the score.

Given these item and context summaries as input, the LLM ranker then scores the item using chain-of-thought reasoning, which has been shown to improve the performance of LLMs on these types of classification / regression tasks [95]. The intermediate chain-of-thought reasoning steps generated by the LLM function as explanations for why certain items are eventually included or left out of the recommendation slate. These explanations can be viewed internally for debugging purposes and also shown to the user, either by including them as input to the dialogue manager that produces utterances within the conversational interface or by postprocessing and including them within pop-up boxes in the visual UI where the recommendation slates are displayed.

### 3.3 User Profile

One of the key advantages to a CRS is the ability of the user to articulate their preferences over the course of a session, so that the system can assist them without necessarily needing any prior background information. Despite this, the personalized experience can be improved if the system has built up a profile of the user beforehand so that there is a mutual starting base to build the conversation on top of. For instance, if a user dislikes jazz music and has shared this previously, they should not have to reiterate this point every new session when searching for music videos.

In traditional deep learning based recommender systems, non-verbal interaction signals such as clicks or ratings are often used to train embedding representations of a user that can be fed into a neural net. In RecLLM we instead represent users with natural language profiles (see e.g. [70]), which can be consumed by an LLM. These are more transparent compared to embeddings and specific pieces of information can usually be attributed to an original source, which aids in explainability. Also, users can manually edit

these natural language profiles, which gives them greater control to monitor and update their preferences. In RecLLM we build user profiles based on a user’s repeated interaction with the system over multiple sessions, although it would be possible to incorporate other data sources as well.

An important open research question is how to structure a user profile in terms of natural language. Currently in RecLLM we represent a user by a set of salient facts we have extracted from prior sessions (e.g. “I do not like listening to jazz while in the car”) similar to [109], although many other more sophisticated schemes are possible. Another extreme possibility is to avoid any lossiness by defining a user profile degenerately as the raw conversational history of all sessions the user has had with the system in the past. In this case we would need to implement an efficient mechanism for an LLM to retrieve relevant facts from this raw history at inference time.

There are three main components to the User Profile module, which we now describe.

*Memory Extraction.* The purpose of the memory extraction component is to identify when a particular utterance contains a meaningful and enduring fact about the user that can be extracted and added to the user profile. In RecLLM, this is currently implemented by an LLM using in-context few-shot learning as part of the dialogue management module.

*Triggering and Retrieval.* The triggering and retrieval component decides at what instances during a session it is likely beneficial to query the user profile for supplementary information and to then retrieve the most relevant facts related to the current context. Currently at each turn RecLLM retrieves a single fact from the user profile by embedding the last user utterance and doing a cosine distance comparison between this embedding and precomputed embeddings of each fact in the user profile. Triggering is implemented post hoc by thresholding on this minimal cosine distance. Better performance is likely possible by using a separate LLM classifier for triggering, retrieving multiple facts from the user profile, and basing retrieval on the entire conversation context of the session as opposed to just the last utterance.

*System Integration.* Once the user profile information is retrieved, it must be integrated into the rest of the system so that it can influence behavior such as the system’s dialogue and API calls to the recommendation engine. How to properly integrate facts coming from a user profile is a difficult open question, as it is highly context dependent how they should modulate short term preferences expressed by the user in the current session. For instance, the system may know that the user is allergic to seafood, but if the user explicitly says they want to see some videos about fish recipes to pass along to a friend it’s important that the system overrides this preference from the user profile and gives the user what they are asking for. In RecLLM we use a simple strategy of injecting facts from the user profile into the text input of the dialogue manager (see Section 3.1). By doing so we allow LLMs powering the dialogue manager to make nuanced decisions about how to utilize this auxiliary information in the context of the ongoing session without having to engineer any hard rules into the system.

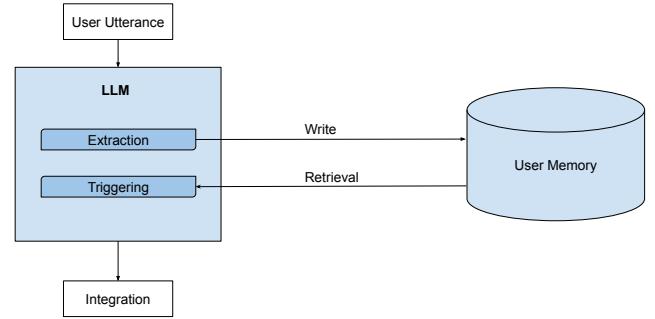


Figure 7: Overview of the architecture incorporating the User Profile module

## 4 SIMULATION AND LARGE-SCALE TUNING

A major impediment to building a high-quality industrial CRS is a lack of data available for training and evaluation. Typically, large-scale recommender systems are trained on user interaction data mined from the logs of existing products; however, conversational recommenders are a nascent technology and for the most part products using this paradigm do not exist yet. An initial high quality system must be built to make such a product viable, after which a bootstrapping cycle can begin in which real data is generated from the system and then increasingly better versions of the system are trained using that data. RecLLM deals with the data sparsity problem by exploiting the transfer learning ability of large language models using in-context few-shot learning or fine-tuning on a small number of manually generated examples. However, we hypothesize that ultimately there is a ceiling to the quality that can be achieved through these approaches, given the long-tail of different scenarios that can arise within a mixed-initiative CRS. In this section we discuss the use of LLM-powered user simulators to generate realistic data at scale and techniques for tuning system components using larger amounts of data.

### 4.1 User Simulation

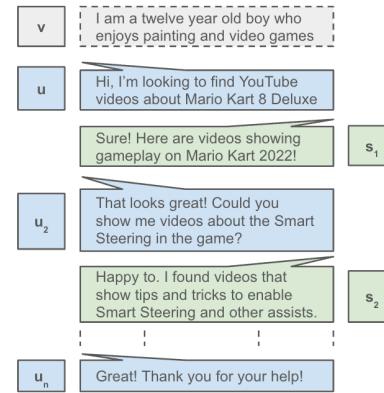
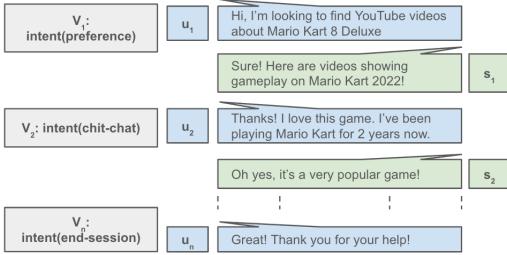


Figure 8: An example of session based control: A single variable (a user profile) is used to condition the user simulator.



**Figure 9: An example of turn level control: A series of variables (user intents) are used to condition the user simulator at each turn.**

According to the conversational recommender setup considered in this paper (see Section 2), a session consists of a sequence  $S = \{s_1, u_1, s_2, u_2, \dots, s_n, u_n\}$ , where each  $u_i$  is a natural language utterance by the user and each  $s_i$  is a combination of a natural language utterance and possibly a slate of recommendations by the CRS. Therefore, a user simulator is defined by a function  $f(S') = U_i$ , where  $S' = \{s_1, u_1, s_2, u_2, \dots, s_i\}$  is a partial session and  $U_i$  is a distribution over possible user utterances  $u_i$  continuing the session. Given a fixed CRS and such a user simulator  $f$ , we can generate a new sample session by having the CRS and  $f$  interact for a given number of turns (i.e. the CRS generates each  $s_i$  and  $f$  generates each  $u_i$ ).

The ideal property we would like our user simulator to have when synthetically generating data for evaluation or training is **realism**: Conversations between the user simulator and CRS should be nearly indistinguishable from conversations between a representative group of real users and the CRS. Let  $R$  be a set of sessions generated by having real users interact with a particular CRS, and  $Q$  be a set of simulated sessions sampled from the CRS and a user simulator  $f$  according to the procedure outlined above. We offer three possible ways to measure the realism of  $f$ :

- Have crowdsource workers attempt to distinguish between simulated sessions coming from  $Q$  and real sessions coming from  $R$ .
- Train a discriminator model [28] on the same differentiation task.
- Let  $g(S) \rightarrow [1, k]$  be a function that classifies a session into  $k$  categories and let  $G = \{g_i\}$  be an ensemble of such classifiers. One way to define such an ensemble is by adapting dialogue state tracking artifacts used within the dialogue management module of a CRS (see Section 3.1). For instance, we can have a classifier that labels the user intent at a specific turn, or the topics that are covered within a session, or the primary sentiment of a session. Once defined, we can measure how close the distributions  $Q$  and  $R$  are by matching statistics according to the classifier ensemble  $G$ .

A necessary condition of realism is **diversity**: Simulated sessions from  $Q$  should have sufficient variation to invoke all the different functionality of a CRS users will encounter in practice when using the system. It may be that in certain situations measuring realism directly is difficult, for instance if collecting a representative set of real user sessions is infeasible. In this case we can at least attempt to

measure the diversity of the user simulator, for instance by defining a notion of entropy of  $Q$  with respect to the classifier ensemble  $G$ .

*Controlled Simulation.* Our starting point for building a user simulator is the observation that an unconstrained LLM built for dialogue such as LaMDA [86] can interact with a CRS in a similar way to real users. The LLM takes as input the full history of the ongoing conversation and outputs the next user utterance, analogous to how a CRS dialogue manager can use an LLM to generate system utterances. However, we would like to exhibit greater control over the simulator to increase its realism. In controlled simulation, we condition the user simulator on additional latent (to the CRS) variables that allow us to guide its behavior in a certain direction. We explore with two different variations:

- **Session-level control:** A single variable  $v$  is defined at the beginning of the session and is used to condition the user simulator throughout the session. For instance, we could define  $v$  as a user profile such as the ones discussed in Section 3.3.
- **Turn-level control:** A distinct variable  $v_i$  is defined at each turn of the session and is used to condition the simulator for that turn. For instance, we could define each  $v_i$  to be a user intent for the simulator to adopt at that turn.

In the case of an LLM user simulator, one way to execute the control is to translate the variable into text that can be included as part of the simulator’s input along with the rest of the conversation. For instance, for the user profile example we could append the statement "I am a twelve year old boy who enjoys painting and video games" to the beginning of the conversation to induce the LLM to imitate this personality. To increase realism, one possible strategy is to define session-level or turn-level variables in terms of the classifiers making up one of the ensembles  $G$  discussed above and then to sample the variables according to the empirical distribution of the collection of real user sessions  $R$ . Another possibility is to ground the conditioning in trajectories coming from real data from a related product. For instance, we could look at query sequences submitted by users in a non-conversational search application and sample turn-level variables as trajectories of topics that match these query sequences.

*Generating Synthetic Training Data.* To use a user simulator to generate data for supervised training of one of the CRS system modules an additional property is needed: ground truth labels that the system can learn from. As a toy example, suppose we are trying to learn a sentiment classifier as part of a traditional dialogue state tracking module. For this we need to generate a set of examples  $S_i, l_i$ , where  $S_i$  is a session  $s_1, u_1, s_2, u_2, \dots, s_n, u_n$  and  $l_i$  is a ground truth label for the primary user sentiment within  $S_i$  coming from a set of possible labels  $L$ , e.g. {angry, satisfied, confused, ...}. We can use controlled user simulation to solve this problem, by defining a session level variable  $v$  over this set of labels  $L$ . First we sample a variable  $v$  from  $L$  (e.g. "angry") and then condition the simulator based on this label, for instance in a priming implementation by appending the message "You are an angry user" to the beginning of the input of the simulator. If we are able to solve this LLM control problem effectively then we can attach a label  $l_i = \text{"angry"}$  to the session  $S_i$  and trust that with high probability it will be accurate.

A more ambitious use case is generating data for training the retrieval and ranking modules discussed in Sections 3.2.1 and 3.2.2. For this we can define a session level variable  $v$  as a tuple  $(x, j)$ , where  $x$  is an item from the corpus and  $j$  is an integer turn index. Once we sample a  $v = (x, j)$ , we condition the simulator to generate a session  $S = \{v, s_1, u_1, s_2, u_2, \dots, s_j, u_j, \dots\}$  such that after  $j$  turns the item  $x$  is a good match for the context  $S$  (i.e. the user would be satisfied if on turn  $s_{j+1}$  the system included  $x$  within a recommendation slate). This session can then be used as an input example for training a recommendation module, where the item  $x$  is a positive instance and other items from the corpus can be sampled as negatives. This is a far more complex conditioning problem, and a simple zero-shot priming instruction (e.g. "Generate a session such that after  $j$  turns item  $x$  is a good match for the context") will not work. How to solve this control problem effectively, either through more sophisticated turn level priming or by tuning the user simulator LLM, is an ongoing research effort.

## 4.2 Tuning System Modules

For the remainder of this section we focus on tuning LLMs within our system using large amounts of synthetically generated data. For concreteness we examine three modules discussed earlier in the paper: Retrieval (Section 3.2.1), Ranking / Explanation (Section 3.2.2), and Dialogue Management (Section 3.1).

*Retrieval.* In Section 4.1 we outlined a strategy for generating synthetic training data for tuning a recommendation module. For retrieval we assume our training examples are tuples of the form  $(S', x_{pos}, \{x_{neg}\})$ , where  $S'$  is a partial session  $s_1, u_1, s_2, u_2, \dots, s_i, u_i$ ,  $x_{pos}$  is an item that is a good match for the context  $S'$  (in the sense defined previously) and  $\{x_{neg}\}$  is a set of negative items generated by some negative sampling procedure. Given this data, we can tune a Generalized Dual Encoder Model (see Section 3.2.1), in which the initial context representation and item representations are each encoded by an LLM. Regardless of whether we choose to tune only the adapter layers of the two tower model or the LLM params as well, the loss is fully differentiable and normal supervised learning with gradient descent suffices.

In Search API Lookup (see Section 3.2.1), an LLM processes the session history and outputs a search query, which then gets passed into a black-box search algorithm. When this architecture is used, the loss is no longer differentiable and ordinary supervised learning is not possible. Instead, we can reframe the setup as a contextual bandit problem [5], where the LLM is a policy, the labels are rewards signals, and the black box search algorithm is treated as the environment (see Figure 10b). If the LLM encoder is shared with other modules we have the choice of tuning protected parameters of the LLM that influence only this task of outputting a search query, or instead tuning shared parameters of the LLM that also influence the behavior of these other modules.

*Ranking.* For this use case we assume our training examples are tuples of the form  $(S', Y)$ , where  $S'$  is a partial session  $s_1, u_1, s_2, u_2, \dots, s_i$  such that  $s_i$  contains a recommendation slate and  $Y$  is a list of relevancy scores for the items in that slate. In Section 3.2.2 we present an LLM based ranking module that jointly generates a score for each item and an explanation for that score. Using this data, we

can tune the ranking LLM to predict the ground truth labels as a regression problem. Using only this relevancy data we cannot directly tune the LLM to generate better explanations, although this is still possible using bootstrapping methods that depend only on labels for the end task (in this case the scoring task) [35, 107].

*Dialogue Management.* In Section 3.1 we present a dialogue management module based on a unified LLM that at each turn generates a sequence of natural language outputs, the last of which triggers a system action such as a response to the user or an API call to the recommendation engine. Our initial implementation involves tuning the LLM on a moderate (e.g.  $O(1000)$ ) number of example sequences meant to demonstrate desired behavior. Here we propose a Reinforcement Learning from Human Feedback (see e.g. [65]) strategy for building on this medium-scale tuning:

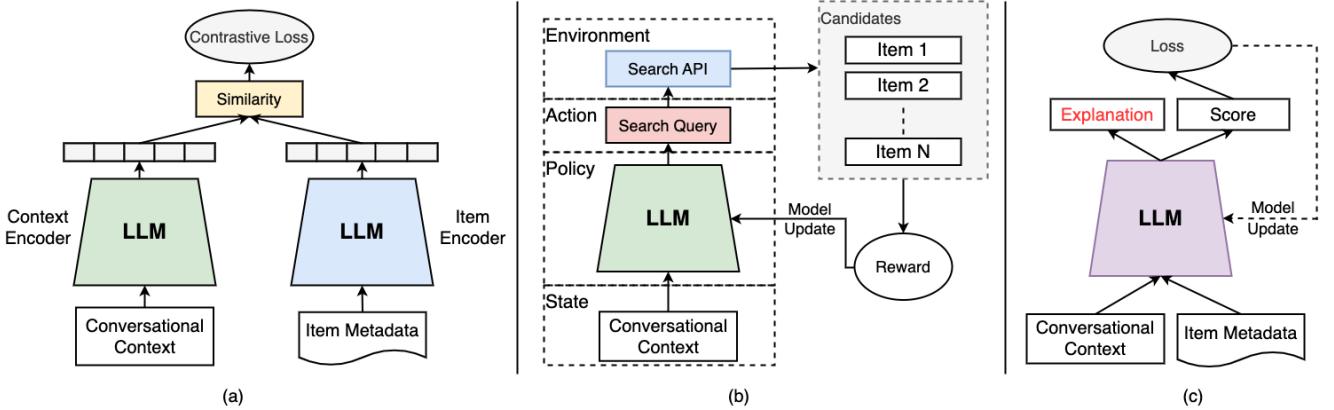
- (1) Generate a set of simulated sessions  $Q$  using a user simulator as outlined in Section 4.1
- (2) Have crowdsource workers evaluate our unified LLM by rating per turn responses within  $Q$  in terms of fluency, interestingness, groundedness etc, as well as giving session level ratings based on overall how effective the system was at helping the user explore the recommendations corpus
- (3) Train reward models on this rating data (likely also using LLMs with chain-of-thought reasoning).
- (4) Further tune the unified LLM on simulated sessions through reinforcement learning to optimize for proxy rewards generated by these reward models

## 5 RELATED WORK

In this section we briefly survey prior research related to the main topics covered in this paper; for a more comprehensive treatment covering CRSs and other conversational information seeking applications, see for instance [22, 24, 39, 106]

*Large Language Models.* Large Language Models based on the transformer architecture [88] have revolutionized the field of artificial intelligence in recent years, producing state of the art results across a variety of natural language understanding and dialogue tasks [6, 15, 71, 86]. In particular these models excel in the zero or few-shot learning setting, where through appropriately engineered prompts they can be adapted to novel tasks without modifying the model parameters [75]. When more training data is available, parameter efficient tuning methods such as prompt tuning can achieve even better performance while still enabling a single LLM to handle multiple sub tasks. [47, 89]. LLMs have also demonstrated an exciting ability to execute multi-step reasoning using chain of thought prompting [95], a faculty that appears to emerge only at certain model scales [94]. A number of recent papers have employed self-consistency and boosting techniques to amplify this reasoning potential further [50, 92, 107]. One challenge explored in this paper is how to effectively harness these new skills within the conversational recommender space, where scarcity of available training data places a high premium on these types of sample efficient learning methods.

*CRS Datasets.* Evaluation of CRSs is difficult in part due to the generative and open-ended nature of the mixed-initiative dialogue (see for example [39] for a more detailed discussion). Many popular



**Figure 10: Tuning Recommendation Modules:** (a) Tuning a General Dual Encoder retrieval model. (b) Tuning a Search API Lookup retrieval model, framed as a contextual bandits problem. (c) Tuning a joint ranking / explanation model. The only learning signal comes from ground truth scores, but through self-consistency / bootstrapping tricks it is possible to indirectly tune the explanations as well.

public CRS datasets are based around relatively small domains such as movies and are conversation-only, i.e. recommendations are offered by the system directly within the dialogue without a notion of recommendation slates or other visual elements (see e.g. [31, 42, 49, 114]). These datasets rely on crowdsource workers to provide examples of good system utterances and recommendations that are treated as ground truth. A few other CRS datasets are adapted from non-conversational datasets involving large-scale domains such as e-commerce or Yelp reviews [45, 112, 114]. However in this case the conversations are generated either synthetically or through substitution from other sources, and are overly rigid compared to actual human dialogue. In this paper we focus on recommendations over a large-scale corpus with recommendation slates distinct from the dialogue, a setup that doesn't fit cleanly into any existing offline benchmarks. As future work we are planning to release human evaluations and a public dataset to quantitatively evaluate design alternatives within RecLLM.

*Dialogue Management.* Early CRSs did not rely on natural language but instead on simple forms of preference elicitation by the system and "critiquing" by the user [7, 51, 57]. When conversational recommenders with natural language interfaces first emerged, they were highly rule-based and limited in their ability to handle mixed-initiative interactions [3, 85]. Later, CRSs with model based language generation and understanding appeared, although they tended to still be narrowly focused on issues such as when to ask a question of the user versus showing recommendations, and what questions to ask [16, 83, 110, 112]. Other works have explored learning more flexible dialogue management modules end-to-end, usually by fine-tuning language models on dialogues collected from crowdsource workers [11, 42, 49, 67], although a recent study has indicated that more progress is needed to make these systems practically useful [38]. In some cases the end-to-end approach has been extended to jointly train a separate item recommendation module along with the dialogue [19, 45, 46, 90]. The unified LLM dialog management architecture from Section 3.1 builds on this prior work by:

- Integrating with a recommendation module that can handle a large scale corpus.
- Learning internal natural language representations such as dialogue state tracking artifacts and self-instructions along with the final dialogue utterances.
- Incorporating natural language inputs such as user profiles and textual representations of recommendation slates from external sources.

*Recommendations / Explanations.* In [105] the authors define a conceptual framework for Retrieval Enhanced Machine Learning; our framework defined in Section 3.2.1 is similar in nature but is simplified and focused on capturing existing approaches to retrieval in the recommendation domain. An overall theme of this paper is how to properly integrate LLMs with external resources, particularly recommendation engines and user profiles, in order to build a better CRS. Some prior research [8, 61, 73, 86] explores with tuning conversational systems through human demonstrations to make calls to an external search API, but not for recommendations over a large corpus. More generally, it is a fundamental research area in machine learning to augment deep learning models with external memory [29, 97, 99], and it has been demonstrated that giving LLMs the ability to retrieve from external corpora can improve performance on tasks like question answering and reduce hallucinations [4, 48, 79].

Explainability has been a longstanding concern in recommender systems [111] and a number of works have previously explored jointly generating explanations and recommendations in more traditional recommender systems [11–13, 23, 58, 91, 113]. Recently, LLMs have been used to explain classifiers and also boost their performance [44, 62, 72]. LLMs have also been used for document ranking [41, 64, 68]; however, we are not aware of previous attempts to apply them to ranking problems in the CRS setting or over large-scale corpora where items are represented by heterogeneous metadata, as we do within RecLLM. What type of explanations a recommender system should share is a difficult question (see e.g. [26, 66]); in RecLLM we currently have the system give post hoc

natural language justifications for item slates, although this still leaves open the question of how to verify their correctness.

*User Profile.* A number of recent works explore extracting transparent natural language user profiles in order to personalize open-ended chat bots [56, 101, 109], and recommender systems [2, 70, 87]. Our proposal from Section 3.3 is perhaps most closely related to BlenderBot [80], which also breaks the problem down into separate extraction, triggering, retrieval and generation phases.

*Simulation / Large scale training.* Various user simulators have been built for training and evaluating recommender systems, often to support experimentation with reinforcement learning algorithms [36, 77, 108, 115]. Recently there has also been a surge in research using LLMs to generate synthetic data for training dialogue systems and text classifiers [18, 59, 69, 103]. Particularly relevant is Unsupervised Data Generation [93], in which an LLM takes a description of a desired label and then generates an input that fits the label. This input / label pair then becomes a synthetic example that can be used for training. Controlled simulation from section 4.1 employs a similar principle where we condition on a latent variable to generate a simulated session and then use the latent variable as a label for tuning. However, we are attempting to generate entire conversations (partially generated by a system outside the simulator’s control) and more sophisticated techniques than basic few-shot prompting are likely required.

In [33, 63, 100] a pretrained language model is tuned to process documents as part of a dual encoder retrieval model, and in [32] this is extended to full conversations as in the Generalized Dual Encoder proposal from Section 4.2. When the ground truth labels do not enable a fully differentiable loss function (such as in Search API Lookup), [65, 82] show it is still effective to tune LLMs for language generation tasks using techniques derived from reinforcement learning. Other works [14, 81] also use reinforcement learning to tune LLMs for open ended or task based dialogue using reward signals inferred from the conversations (e.g. through sentiment analysis or a notion of task completion). The proposal for tuning a dialogue manager LLM in Section 4.2 is an example of Reinforcement Learning from Human Feedback [1, 27, 65], a technique that is often used for teaching LLMs to follow instructions and align better with human values.

## 6 RECLLM PROTOTYPE

We have built an initial RecLLM prototype based on the outline shared within this paper. Retrieval is currently implemented via Search API Lookup (see Section 3.2.1) using in-context few-shot learning and a public YouTube search API. LaMDA [86] is currently used as the underlying LLM powering dialogue management, recommendations and explanations, user profile integration and user simulation within the system. In Appendix A we share sample sessions from RecLLM demonstrating some of its core competencies.

## 7 ETHICAL CONSIDERATIONS

It is our belief that by leveraging large language models within CRSs we can mitigate some challenging ethical problems that have been noted in many recommender systems. RecLLM has the following desirable properties:

- A recommendation module that reasons over the attributes of items and is less reliant on learning from interaction data such as clicks that are noisy and can promote unintentional biases.
- The ability to give natural language justifications for why certain recommendations are being shown, which the user can then validate.
- Opportunity for the user to control their recommendations in a nuanced way through language.
- Transparent personalization through human interpretable and editable user profiles.

On the other hand, our proposed system relies heavily on large language models and therefore inherits all of their well-known problems centered around societal biases learned through pretraining, hallucinations, and expensive use of resources [96]. Various controls are included to constrain the LLMs to the conversational recommender task, but these are unlikely to fully wash away their inherent issues. Significant further progress needs to be made in areas like debiasing, grounding in factuality and efficient serving before we can safely deploy this type of system in a production setting.

## 8 CONCLUSIONS AND FUTURE WORK

In this paper we examine the system architecture of a conversational recommender system and identify areas where large language models can unlock new capabilities, along with the technical challenges that emerge through their use. In particular we reimagine how LLMs can transform dialogue management, retrieval, ranking and user profiles to improve system quality, give the user greater control and increase transparency throughout the system. We focus on how to build a large-scale end-to-end CRS without assuming access to logs data coming from an existing product, by utilizing the generalization abilities of LLMs and generating synthetic training data using LLM-powered user simulators. As a proof of concept we introduce RecLLM and share example conversations highlighting its diverse functionality. Our hope is that this roadmap can accelerate progress towards a world where controllable and explainable CRSs allow users to explore content within a healthier recommender system ecosystem.

Some important items for future work include:

- We are planning the release of human evaluations and a public dataset based on our system to quantitatively evaluate design alternatives for RecLLM and help the community better study CRSs in the multimodal, large-scale setting.
- In this paper we assume a simplified setting where users interact with the system only through conversation. We would like to generalize our system to handle more realistic scenarios where users give feedback through other channels as well such as clicking on items or like buttons. We would also like to consider more complicated recommender system UIs containing hierarchical structures such as item shelves as opposed to just flat slates.
- We have proposed ideas for large-scale tuning of the main system modules based on synthetically generated data, but currently RecLLM relies exclusively on in-context few-shot learning or tuning on small amounts of data collected through

crowdsourcing. Successfully proving out these ideas will be critical to properly handle huge item corpora and the full space of possible conversations.

- We would like to support new use cases that naturally arise in a mixed-initiative conversational recommender dialogue, such as question answering over corpus items.

## 9 ACKNOWLEDGEMENTS

We would like to thank Filip Radlinski, Karan Singhal, Abhinav Rastogi, Raghav Gupta and Yinlam Chow for useful feedback on drafts of this paper.

## REFERENCES

- [1] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862* (2022).
- [2] Krisztian Balog, Filip Radlinski, and Shushan Arakelyan. 2019. Transparent, scrubtable and explainable user models for personalized recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. 265–274.
- [3] Nicholas J Belkin, Colleen Cool, Adelheit Stein, and Ulrich Thiel. 1995. Cases, scripts, and information-seeking strategies: On the design of interactive information retrieval systems. *Expert systems with applications* 9, 3 (1995), 379–395.
- [4] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2021. Improving language models by retrieving from trillions of tokens. *arXiv preprint arXiv:2112.04426* (2021).
- [5] Djallel Bouneffouf, Irina Rish, and Charu Aggarwal. 2020. Survey on applications of multi-armed and contextual bandits. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.
- [6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.
- [7] Robin D Burke, Kristian J Hammond, and BC Yound. 1997. The FindMe approach to assisted browsing. *IEEE Expert* 12, 4 (1997), 32–40.
- [8] Bill Byrne, Karthik Krishnamoorthi, Saravanan Ganesh, and Mihir Sanjay Kale. 2020. TicketTalk: Toward human-level performance with end-to-end, transaction-based dialog systems. *arXiv preprint arXiv:2012.12458* (2020).
- [9] Wanling Cai and Li Chen. 2020. Predicting user intents and satisfaction with dialogue-based conversational recommendations. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*. 33–42.
- [10] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*. 129–136.
- [11] Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang, and Jie Tang. 2019. Towards knowledge-based recommender dialog system. *arXiv preprint arXiv:1908.05391* (2019).
- [12] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 108–116.
- [13] Xu Chen, Yongfeng Zhang, and Zheng Qin. 2019. Dynamic explainable recommendation based on neural attentive models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 53–60.
- [14] Yinlam Chow, Aza Tulepbergenov, Ofir Nachum, MoonKyung Ryu, Mohammad Ghavamzadeh, and Craig Boutilier. 2022. A Mixture-of-Expert Approach to RL-based Dialogue Management. *arXiv preprint arXiv:2206.00059* (2022).
- [15] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311* (2022).
- [16] Konstantina Christakopoulou, Alex Beutel, Rui Li, Sagar Jain, and Ed H Chi. 2018. Q&R: A two-stage approach toward interactive recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 139–148.
- [17] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. New York, NY, USA.
- [18] Zhuyun Dai, Arun Tejasvi Chaganty, Vincent Y Zhao, Aida Amini, Qazi Mamunur Rashid, Mike Green, and Kelvin Guu. 2022. Dialog inpainting: Turning documents into dialogs. In *International Conference on Machine Learning*. PMLR, 4558–4586.
- [19] Yang Deng, Yaliang Li, Fei Sun, Bolin Ding, and Wai Lam. 2021. Unified conversational recommendation policy learning via graph-based reinforcement learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1431–1441.
- [20] Linus W Dietz, Saadi Myftija, and Wolfgang Wörndl. 2019. Designing a conversational travel recommender system based on data-driven destination characterization. In *ACM RecSys workshop on recommenders in tourism*. 17–21.
- [21] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. 2018. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *Proceedings of the 2018 world wide web conference*. 1775–1784.
- [22] Chongming Gao, Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. 2021. Advances and challenges in conversational recommender systems: A survey. *AI Open* 2 (2021), 100–126.
- [23] Jingyue Gao, Xiting Wang, Yasha Wang, and Xing Xie. 2019. Explainable recommendation through attentive multi-view learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3622–3629.
- [24] Jianfeng Gao, Chenyan Xiong, Paul Bennett, and Nick Craswell. 2022. Neural approaches to conversational information retrieval. *arXiv preprint arXiv:2201.05176* (2022).
- [25] Venkata Rama Kiran Garimella and Ingmar Weber. 2017. A long-term analysis of polarization on Twitter. In *Eleventh international AAAI conference on web and social media*.
- [26] Fatih Gedikli, Dietmar Jannach, and Mouzhi Ge. 2014. How should I explain? A comparison of different explanation types for recommender systems. *International Journal of Human-Computer Studies* 72, 4 (2014), 367–382.
- [27] Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, et al. 2022. Improving alignment of dialogue agents via targeted human judgments. *arXiv preprint arXiv:2209.14375* (2022).
- [28] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* 63, 11 (2020), 139–144.
- [29] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* (2014).
- [30] Ruiqi Guo, Quan Geng, David Simcha, Felix Chern, Sanjiv Kumar, and Xiang Wu. 2019. New Loss Functions for Fast Maximum Inner Product Search. *CoRR* abs/1908.10396 (2019). arXiv:1908.10396 <http://arxiv.org/abs/1908.10396>
- [31] Shirley Anugrah Hayati, Dongyeop Kang, Qingxiao Yang Zhu, Weiyi Shi, and Zhou Yu. 2020. INSPIRED: Toward sociable recommendation dialog systems. *arXiv preprint arXiv:2009.14306* (2020).
- [32] Matthew Henderson, Iñigo Casanueva, Nikola Mrkšić, Pei-Hao Su, Tsung-Hsien Wen, and Ivan Vilic. 2019. ConveRT: Efficient and accurate conversational representations from transformers. *arXiv preprint arXiv:1911.03688* (2019).
- [33] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 113–122.
- [34] Jie Huang and Kevin Chen-Chuan Chang. 2022. Towards Reasoning in Large Language Models: A Survey. *arXiv preprint arXiv:2212.10403* (2022).
- [35] Jiaxin Huang, Shixiang Gu, Le Hou, Yuxin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *arXiv preprint arXiv:2210.11610* (2022).
- [36] Eugene Ie, Chih-wei Hsu, Martin Mladenov, Vihan Jain, Sammit Narvekar, Jing Wang, Rui Wu, and Craig Boutilier. 2019. Recsim: A configurable simulation platform for recommender systems. *arXiv preprint arXiv:1909.04847* (2019).
- [37] Dietmar Jannach and Li Chen. 2022. Conversational Recommendation: A Grand AI Challenge. *arXiv preprint arXiv:2203.09126* (2022).
- [38] Dietmar Jannach and Ahtsham Manzoor. 2020. End-to-End Learning for Conversational Recommendation: A Long Way to Go?. In *IntRS@RecSys*. 72–76.
- [39] 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.
- [40] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2022. Survey of hallucination in natural language generation. *Comput. Surveys* (2022).
- [41] Jia-Huei Ju, Jheng-Hong Yang, and Chuan-Ju Wang. 2021. Text-to-text Multi-view Learning for Passage Re-ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1803–1807.
- [42] Dongyeop Kang, Anusha Balakrishnan, Pararth Shah, Paul Crook, Y-Lan Boureau, and Jason Weston. 2019. Recommendation as a communication game: Self-supervised bot-play for goal-oriented dialogue. *arXiv preprint arXiv:1909.03922* (2019).
- [43] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. 2017. Interpretability Beyond Feature Attribution:

- Quantitative Testing with Concept Activation Vectors (TCAV). (2017). <https://doi.org/10.48550/ARXIV.1711.11279>
- [44] Andrew K Lampinen, Ishita Dasgupta, Stephanie CY Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L McClelland, Jane X Wang, and Felix Hill. 2022. Can language models learn from explanations in context? *arXiv preprint arXiv:2204.02329* (2022).
- [45] Wenqiang Lei, Xiangnan He, Yisong Mao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020. Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 304–312.
- [46] Wenqiang Lei, Gangyi Zhang, Xiangnan He, Yisong Mao, Xiang Wang, Liang Chen, and Tat-Seng Chua. 2020. Interactive path reasoning on graph for conversational recommendation. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2073–2083.
- [47] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691* (2021).
- [48] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.
- [49] Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards deep conversational recommendations. *Advances in neural information processing systems* 31 (2018).
- [50] Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2022. On the Advance of Making Language Models Better Reasoners. *arXiv preprint arXiv:2206.02336* (2022).
- [51] Greg Linden, Steve Hanks, and Neal Lesh. 1997. Interactive assessment of user preference models: The automated travel assistant. In *User Modeling*. Springer, 67–78.
- [52] Frederick Liu, Siamak Shakeri, Hongkun Yu, and Jing Li. 2021. EncT5: Fine-tuning T5 Encoder for Non-autoregressive Tasks. *arXiv preprint arXiv:2110.08426* (2021).
- [53] Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. *arXiv preprint arXiv:1905.13164* (2019).
- [54] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Ji Yang, Minmin Chen, Jiaxi Tang, Lichan Hong, and Ed H Chi. 2020. Off-policy learning in two-stage recommender systems. In *Proceedings of The Web Conference 2020*. 463–473.
- [55] Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. 2020. Feedback loop and bias amplification in recommender systems. In *Proceedings of the 29th ACM international conference on information & knowledge management*. 2145–2148.
- [56] Pierre-Emmanuel Mazaré, Samuel Humeau, Martin Raison, and Antoine Borde. 2018. Training millions of personalized dialogue agents. *arXiv preprint arXiv:1809.01984* (2018).
- [57] Kevin McCarthy, Yasser Salem, and Barry Smyth. 2010. Experience-based critiquing: Reusing critiquing experiences to improve conversational recommendation. In *International Conference on Case-Based Reasoning*. Springer, 480–494.
- [58] James McInerney, Benjamin Lacker, Samantha Hansen, Karl Higley, Hugues Bouchard, Alois Gruson, and Rishabh Mehrotra. 2018. Explore, exploit, and explain: personalizing explainable recommendations with bandits. In *Proceedings of the 12th ACM conference on recommender systems*. 31–39.
- [59] Shikib Mehri, Yasemin Altun, and Maxine Eskenazi. 2022. LAD: Language Models as Data for Zero-Shot Dialog. *arXiv preprint arXiv:2207.14393* (2022).
- [60] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
- [61] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332* (2021).
- [62] Sharan Narang, Colin Raffel, Katherine Lee, Adam Roberts, Noah Fiedel, and Karishma Malkan. 2020. WT5?! training text-to-text models to explain their predictions. *arXiv preprint arXiv:2004.14546* (2020).
- [63] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y Zhao, Yi Luan, Keith B Hall, Ming-Wei Chang, et al. 2021. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899* (2021).
- [64] Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. *arXiv preprint arXiv:2003.06713* (2020).
- [65] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155* (2022).
- [66] Sunjeong Park and Lim Youn-kyung. 2019. Design considerations for explanations made by a recommender chatbot. In *IASDR Conference 2019*. IASDR.
- [67] Gustavo Penha and Claudia Hauff. 2020. What does bert know about books, movies and music? probing bert for conversational recommendation. In *Fourteenth ACM Conference on Recommender Systems*. 388–397.
- [68] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2021. The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models. *arXiv preprint arXiv:2101.05667* (2021).
- [69] Raul Puri, Ryan Spring, Mostafa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2020. Training question answering models from synthetic data. *arXiv preprint arXiv:2002.09599* (2020).
- [70] Filip Radlinski, Krisztian Balog, Fernando Diaz, Lucas Dixon, and Ben Wedin. 2022. On Natural Language User Profiles for Transparent and Scrutable Recommendation. *arXiv preprint arXiv:2205.09403* (2022).
- [71] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 21, 140 (2020), 1–67.
- [72] Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. *arXiv preprint arXiv:1906.02361* (2019).
- [73] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 8689–8696.
- [74] Xuhui Ren, Hongzhi Yin, Tong Chen, Hao Wang, Zi Huang, and Kai Zheng. 2021. Learning to ask appropriate questions in conversational recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 808–817.
- [75] Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–7.
- [76] Francesco Ricci and Quang-Nhat Nguyen. 2007. Acquiring and revising preferences in a critique-based mobile recommender system. *IEEE Intelligent systems* 22, 3 (2007), 22–29.
- [77] David Rohde, Stephen Bonner, Travis Dunlop, Flavian Vasile, and Alexandros Karatzoglou. 2018. Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising. *arXiv preprint arXiv:1808.00720* (2018).
- [78] Tobias Schnabel, Paul N Bennett, Susan T Dumais, and Thorsten Joachims. 2018. Short-term satisfaction and long-term coverage: Understanding how users tolerate algorithmic exploration. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 513–521.
- [79] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567* (2021).
- [80] Kurt Shuster, Jing Xu, Mojtaba Komeili, Da Ju, Eric Michael Smith, Stephen Roller, Megan Ung, Moya Chen, Kushal Arora, Joshua Lane, et al. 2022. BlenderBot 3: a deployed conversational agent that continually learns to responsibly engage. *arXiv preprint arXiv:2208.03188* (2022).
- [81] Charlie Snell, Sherry Yang, Justin Fu, Yi Su, and Sergey Levine. 2022. Context-Aware Language Modeling for Goal-Oriented Dialogue Systems. *arXiv preprint arXiv:2204.10198* (2022).
- [82] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems* 33 (2020), 3008–3021.
- [83] Yueming Sun and Yi Zhang. 2018. Conversational recommender system. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 235–244.
- [84] Yi Tay, Vinh Q Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. 2022. Transformer memory as a differentiable search index. *arXiv preprint arXiv:2202.06991* (2022).
- [85] Cynthia A Thompson, Mehmet H Goker, and Pat Langley. 2004. A personalized system for conversational recommendations. *Journal of Artificial Intelligence Research* 21 (2004), 393–428.
- [86] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239* (2022).
- [87] Ghazaleh H Torbati, Andrew Yates, and Gerhard Weikum. 2021. You get what you chat: Using conversations to personalize search-based recommendations. In *European Conference on Information Retrieval*. Springer, 207–223.
- [88] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [89] Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2021. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904* (2021).

- [90] Lingzhi Wang, Huang Hu, Lei Sha, Can Xu, Kam-Fai Wong, and Dixin Jiang. 2021. Finetuning Large-Scale Pre-trained Language Models for Conversational Recommendation with Knowledge Graph. *CoRR* abs/2110.07477 (2021). arXiv:2110.07477 <https://arxiv.org/abs/2110.07477>
- [91] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. 2018. A reinforcement learning framework for explainable recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 587–596.
- [92] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171* (2022).
- [93] Zirui Wang, Adams Wei Yu, Orhan Firat, and Yuan Cao. 2021. Towards zero-label language learning. *arXiv preprint arXiv:2109.09193* (2021).
- [94] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682* (2022).
- [95] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903* (2022).
- [96] Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glæse, Borja Balle, Atoosa Kasirzadeh, et al. 2021. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359* (2021).
- [97] Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916* (2014).
- [98] Jason Weston, Emily Dinan, and Alexander H. Miller. 2018. Retrieve and Refine: Improved Sequence Generation Models For Dialogue. *CoRR* abs/1808.04776 (2018). arXiv:1808.04776 <http://arxiv.org/abs/1808.04776>
- [99] Yuhuai Wu, Markus N. Rabe, DeLesley S. Hutchins, and Christian Szegedy. 2022. Memorizing Transformers. *ArXiv* abs/2203.08913 (2022).
- [100] Lee Xiong, Chenyan Xiong, Yi Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808* (2020).
- [101] Jing Xu, Arthur Szlam, and Jason Weston. 2021. Beyond goldfish memory: Long-term open-domain conversation. *arXiv preprint arXiv:2107.07567* (2021).
- [102] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. 2019. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 269–277.
- [103] Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyeong Park. 2021. GPT3Mix: Leveraging large-scale language models for text augmentation. *arXiv preprint arXiv:2104.08826* (2021).
- [104] Yisong Yue, Rajan Patel, and Hein Roehrig. 2010. Beyond Position Bias: Examining Result Attractiveness as a Source of Presentation Bias in Clickthrough Data. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. Association for Computing Machinery, New York, NY, USA, 1011–1018. <https://doi.org/10.1145/1772690.1772793>
- [105] Hamed Zamani, Fernando Diaz, Mostafa Dehghani, Donald Metzler, and Michael Bendersky. 2022. Retrieval-Enhanced Machine Learning. *arXiv preprint arXiv:2205.01230* (2022).
- [106] Hamed Zamani, Johanne R Trippas, Jeff Dalton, and Filip Radlinski. 2022. Conversational information seeking. *arXiv preprint arXiv:2201.08808* (2022).
- [107] Eric Zelikman, Yuhuai Wu, and Noah D Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465* (2022).
- [108] Shuo Zhang and Krisztian Balog. 2020. Evaluating conversational recommender systems via user simulation. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 1512–1520.
- [109] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243* (2018).
- [110] Xiaoying Zhang, Hong Xie, Hang Li, and John CS Lui. 2020. Conversational contextual bandit: Algorithm and application. In *Proceedings of the web conference 2020*. 662–672.
- [111] Yongfeng Zhang, Xu Chen, et al. 2020. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval* 14, 1 (2020), 1–101.
- [112] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W Bruce Croft. 2018. Towards conversational search and recommendation: System ask, user respond. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 177–186.
- [113] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. 83–92.
- [114] Kun Zhou, Yuanhang Zhou, Wayne Xin Zhao, Xiaoke Wang, and Ji-Rong Wen. 2020. Towards topic-guided conversational recommender system. *arXiv preprint arXiv:2010.04125* (2020).
- [115] Lixin Zou, Long Xia, Pan Du, Zhuo Zhang, Ting Bai, Weidong Liu, Jian-Yun Nie, and Dawei Yin. 2020. Pseudo Dyna-Q: A reinforcement learning framework for interactive recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 816–824.

## A RECLLM SAMPLES

In this appendix we share a small number of samples meant to demonstrate some of the core competencies of RecLLM. They are grouped into the following categories: General System Overview, Refinement and Explanation, Simulated Conversations, Topic Exploration and Incorporation of User Profiles. Note that in these samples we only visualize the final recommendation slate, although in reality the recommendation slate evolves throughout the conversation within the RecLLM user interface.

Please note that all conversations in this prototype are stored anonymously for evaluation and analysis.

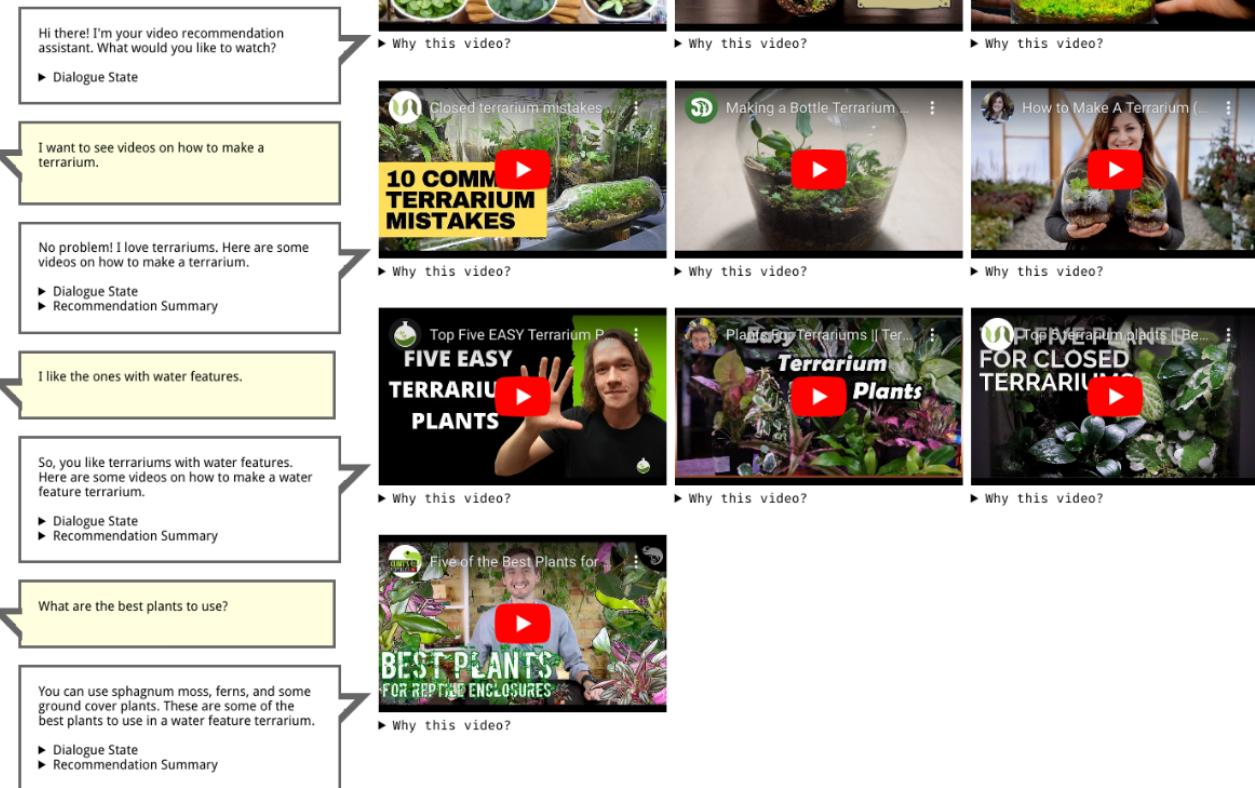
New session initialized



Figure 11: General System Overview Sample 1

Please note that all conversations in this prototype are stored anonymously for evaluation and analysis.

New session initialized



Enter free-form text ▼

Figure 12: General System Overview Sample 2

Please note that all conversations in this prototype are stored anonymously for evaluation and analysis.

New session initialized

Hi there! I'm your video recommendation assistant. What would you like to watch?

► Dialogue State

I'd like to see some animal videos

I love animal videos! Here are some popular animal videos that are cute and funny.

► Dialogue State  
► Recommendation Summary

I'm interested in the African ones.

The animal videos of Africa are fascinating. Here are some popular videos of African animals.

► Dialogue State  
► Recommendation Summary

► Why this video?  
Relaxation video of African animals

► Why this video?  
Top 5 African animals videos

► Why this video?  
Top 10 videos of African animals fighting back against predators

► Why this video?  
Scenic wildlife film of African animals in Meru National Park

► Why this video?  
Top African animals ranked by strength

► Why this video?  
Video of African wildlife in Lake Manyara National Park

► Why this video?  
Compilation of videos of African animal attacks

► Why this video?  
African animal videos of lions, elephants, and hippos

► Why this video?  
Video of a mother cheetah with her cubs and a pride of lions in the Masai Mara

► Why this video?  
Relaxing video of African wildlife

Enter free-form text

Figure 13: Refinement and Explanation Sample 1

Please note that all conversations in this prototype are stored anonymously for evaluation and analysis.

New session initialized

Hi there! I'm your video recommendation assistant. What would you like to watch?  
► Dialogue State

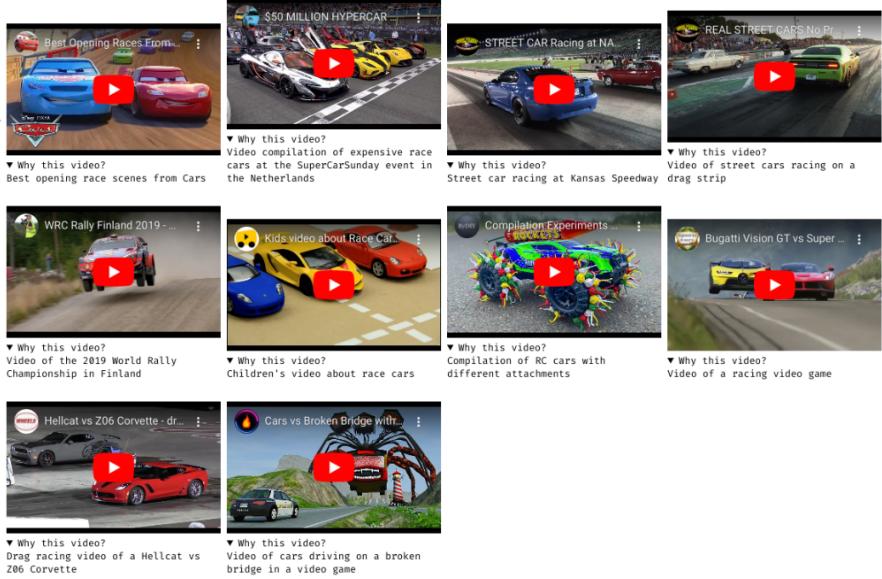


Figure 14: Refinement and Explanation Sample 2

Please note that all conversations in this prototype are stored anonymously for evaluation and analysis.

New session initialized

Hi there! I'm your video recommendation assistant. What would you like to watch?  
► Dialogue State

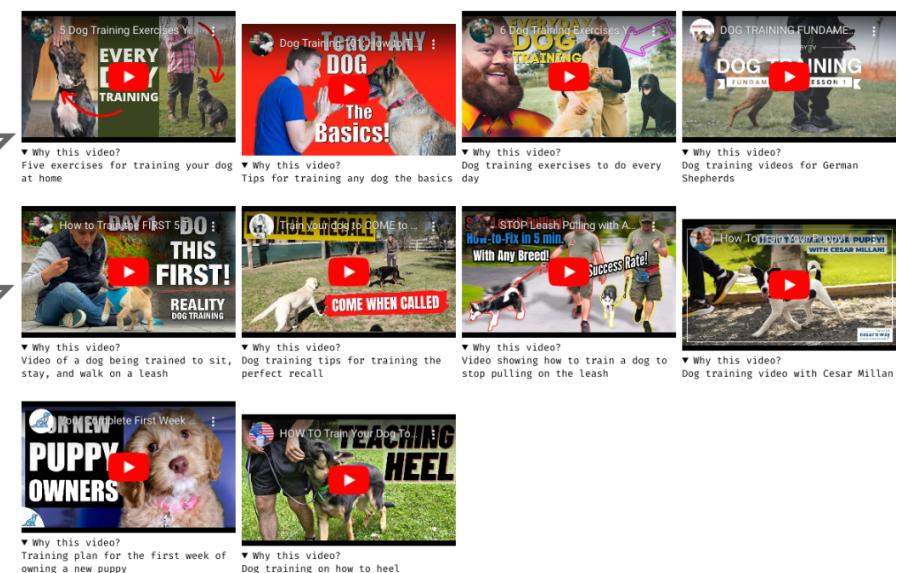


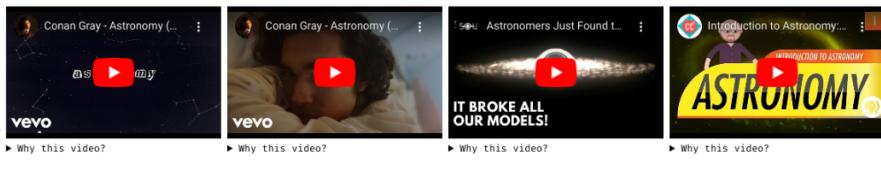
Figure 15: Refinement and Explanation Sample 3

## Simulated Conversation

Please note that all conversations in this prototype are stored anonymously for evaluation and analysis.

New session initialized

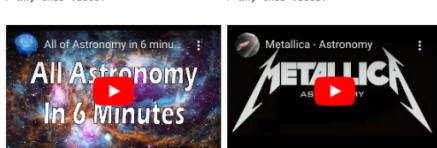
Hi there! I'm your video recommendation assistant. What would you like to watch?  
► Dialogue State



Oh, I am very interested in space exploration. What would you suggest?



Sure! Here are some videos about the history of space exploration. You will learn about the first humans in space, and the first moon landing.  
► Dialogue State  
► Recommendation Summary



That sounds fascinating! Are there any videos about the future of space exploration? Or about the other planets?



Of course! Here are some videos about the future of space exploration, and the other planets.  
► Dialogue State  
► Recommendation Summary

That's great. I am also interested in astronomy. Do you have anything about that?

► Why this video?

I have some astronomy videos. You can learn about the different planets, stars, and galaxies.  
► Dialogue State  
► Recommendation Summary



Please enter a word or phrase that represents the topic you would like the simulator to create a conversation about.

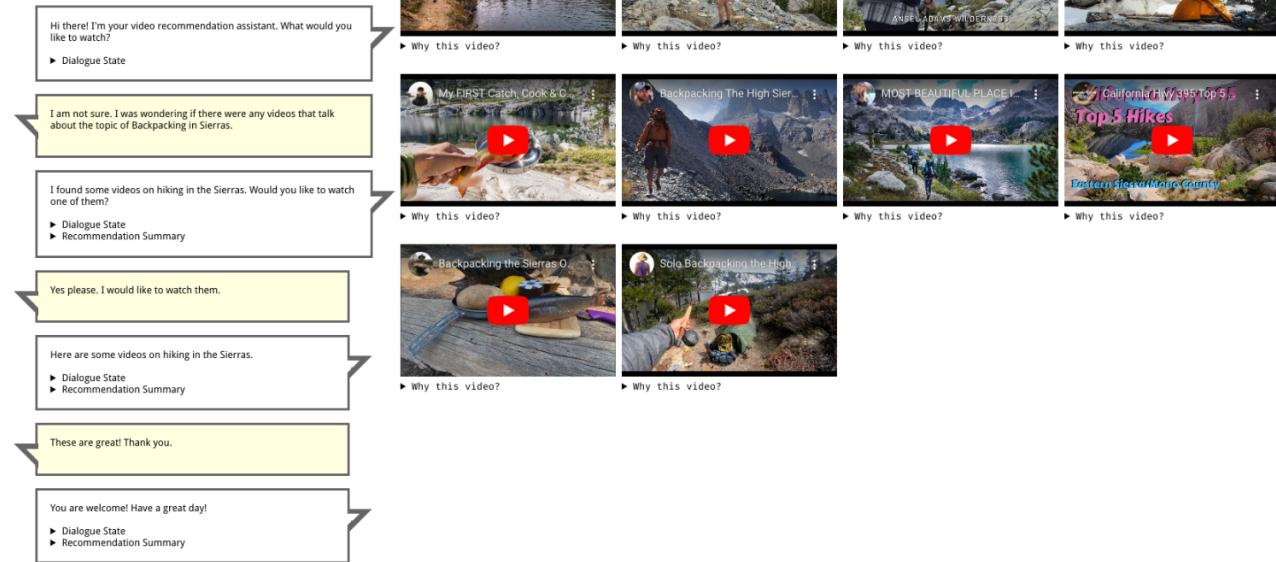
Science

**Figure 16: Simulated Conversations Sample 1**

### Simulated Conversation

Please note that all conversations in this prototype are stored anonymously for evaluation and analysis.

New session initialized



Please enter a word or phrase that represents the topic you would like the simulator to create a conversation about.

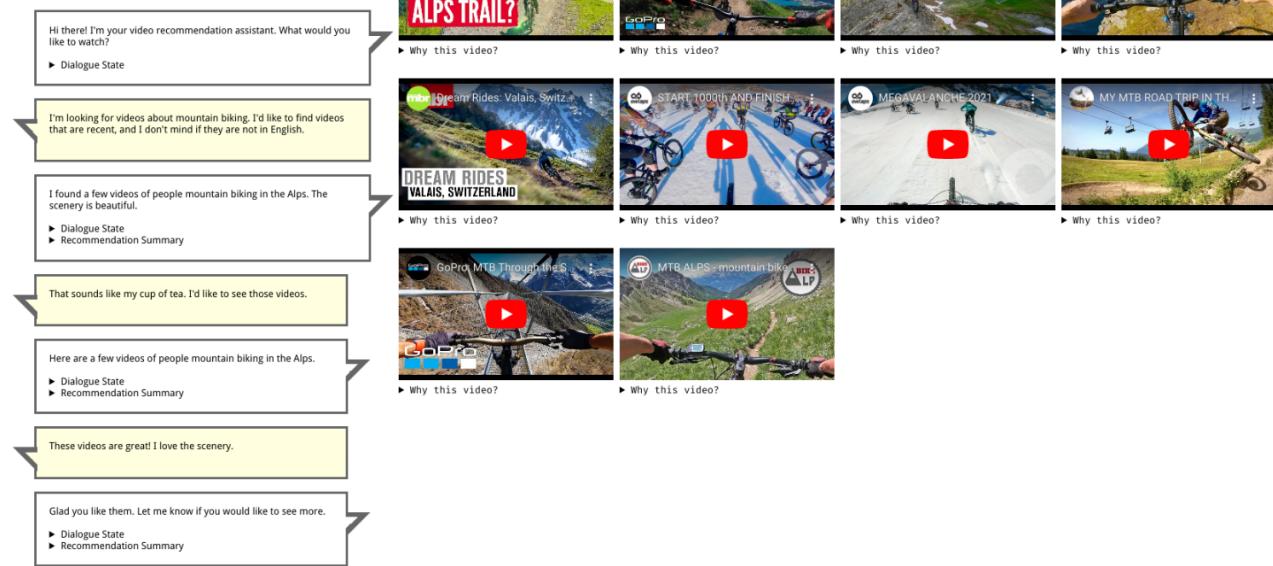
Backpacking in Sierras

**Figure 17: Simulated Conversations Sample 2**

### Simulated Conversation

Please note that all conversations in this prototype are stored anonymously for evaluation and analysis.

New session initialized



Please enter a word or phrase that represents the topic you would like the simulator to create a conversation about.

Mountain biking

Figure 18: Simulated Conversations Sample 3

Please note that all conversations in this prototype are stored anonymously for evaluation and analysis.

New session initialized

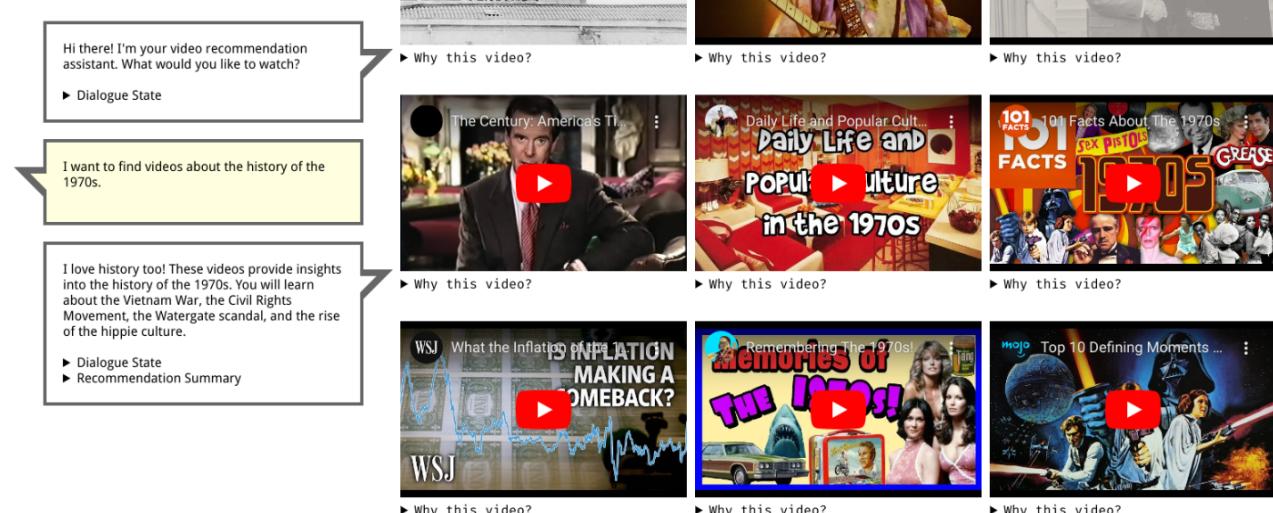


Figure 19: Exploration Sample 1

Please note that all conversations in this prototype are stored anonymously for evaluation and analysis.

New session initialized

Hi there! I'm your video recommendation assistant. What would you like to watch?

► Dialogue State

I'm vacationing in Switzerland, what should I see?

I love Switzerland. You should definitely visit the Swiss Alps. There are many beautiful views and activities like skiing, hiking, and sightseeing.

► Dialogue State

► Recommendation Summary

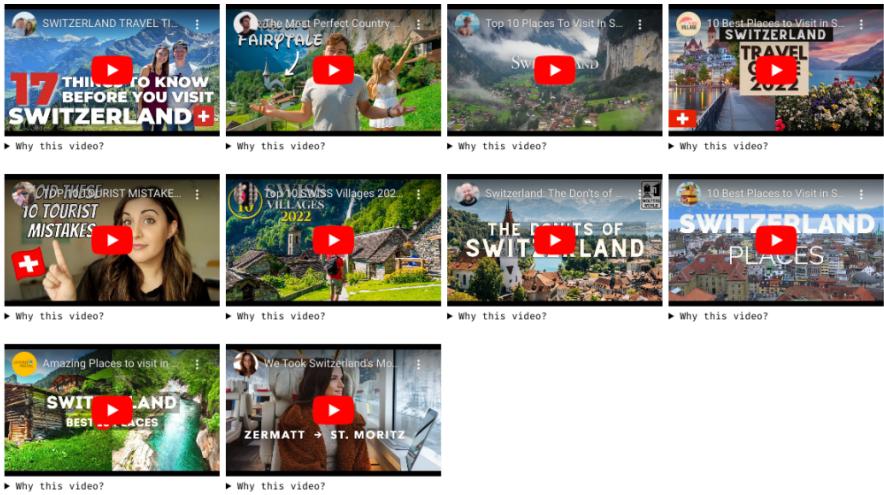


Figure 20: Exploration Sample 2

Please note that all conversations in this prototype are stored anonymously for evaluation and analysis.

New session initialized

Hi there! I'm your video recommendation assistant. What would you like to watch?

► Dialogue State

I enjoy playing computer games, can you recommend any?

Sure! What kind of games do you like? Puzzle? Action? Strategy? Here are some popular computer games for you to try.

► Dialogue State

► Recommendation Summary

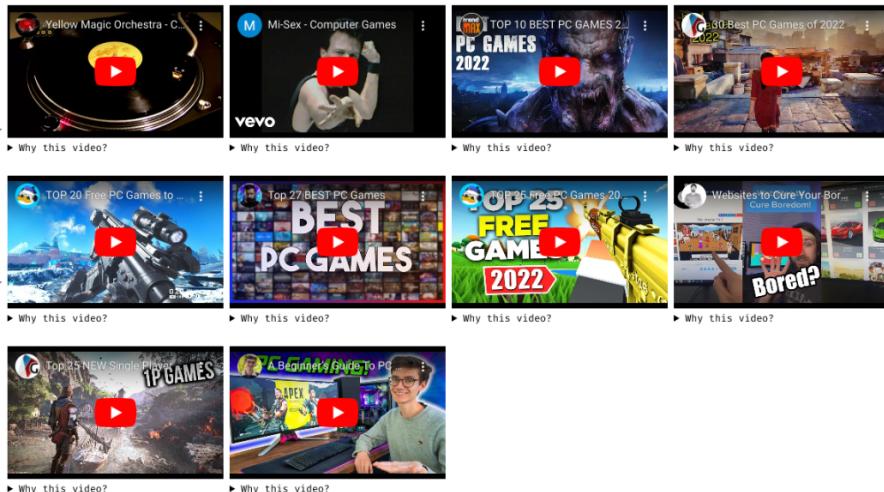


Figure 21: Exploration Sample 3

# User Profile

Add to your user profile.

```
I love the outdoors.  
Pottery is fun.  
I like basketball.  
I have a cat.  
I like listening to Dvorak.
```

Figure 22: An example user profile

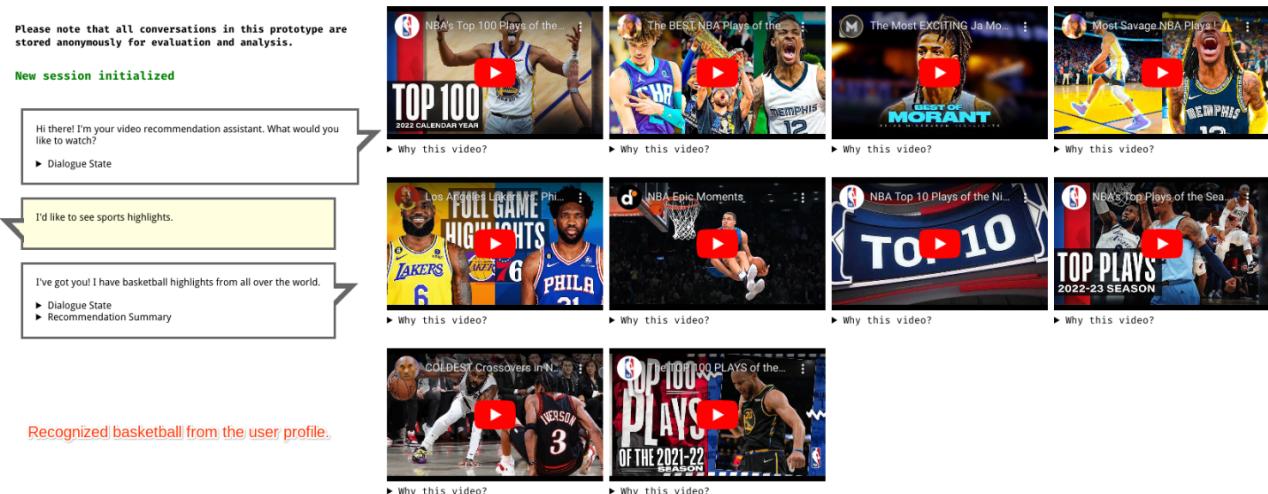


Figure 23: Incorporation of User Profiles Sample 1

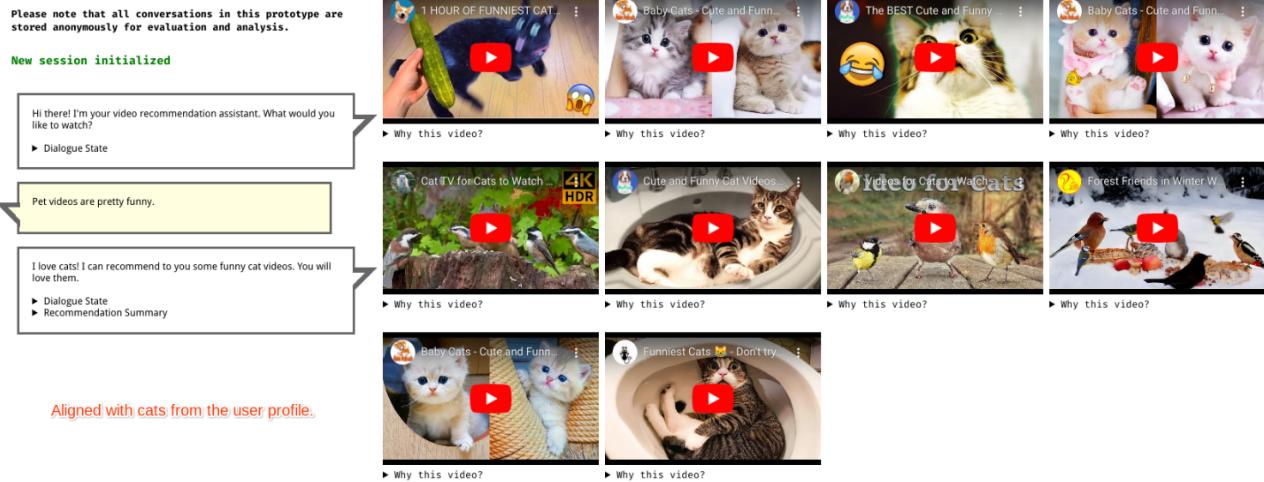


Figure 24: Incorporation of User Profiles Sample 2