Empowering Recommender Systems with Agentic AI: Towards Adaptive Online Personalization

Ahmed Aly[1], Ahmed Ibrahim[2], and Rasha Kashef[3]

Electrical, Computer and Biomedical Engineering Department
Toronto Metropolitan University, Toronto, Canada
{ ahmed.aly, a5ibrahim, rkashef}@torontomu.ca

Abstract. Recommender systems are ubiquitous in today's digital online platforms, powering personalized experiences on platforms ranging from e-commerce giants like Amazon to media services such as Netflix. These systems utilize algorithms to recommend items tailored to individual user preferences and historical behaviour. Recently, the emergence of the Agentic AI paradigm, defined by goal-oriented behavior, contextual awareness, and minimal reliance on human instruction, offers a transformative path forward. This paper provides an overview of recommender system models, ranging from foundational algorithms to the latest innovations driven by Large Language Models (LLMs) and multi-agent frameworks. We explore how Agentic AI is reshaping the design of next-generation recommender systems by enabling dynamic interaction, real-time learning, and simulation-based personalization. Through a comparative analysis of current models, we highlight the benefits of the paradigm—such as improved adaptability and autonomy—alongside persistent challenges, including model interpretability and system complexity. By synthesizing current literature and identifying key trends, this paper aims to offer a roadmap for future research at the intersection of Agentic AI and recommender technologies.

Keywords: Recommender Systems, Agentic AI, Generative AI, conversational recommender system (CRS), Large Language Models (LLMs).

1 Introduction

Recommendation systems are essential tools for guiding users through the information overload found in nearly every app, social media, website, and service today [1]. Thus, recommender systems that can better learn users' preferences and provide recommendations in a more "human-like" manner are likely to lead to a more positive user experience. While there are many approaches to creating a "human-like" recommender system, one strategy that has been widely adopted is the implementation of Large Language Models (LLMs) into the recommendation process. LLMs have proven to be powerful tools, and their ability to reason and generate responses to user prompts has become the backbone of most modern recommender system models. One of the recent advancements in the LLM field is the rise of the Agentic AI paradigm, which opens new possibilities for creating "human-like" recommender systems. Agentic AI

refers to autonomous AI systems that are adaptive, context-aware, and require minimal to no human intervention or instruction [2]. Moreover, Agentic AI differentiates itself from older AI paradigms through its generalizability and applicability to multiple problems [2].

In this paper, we present a variety of recommender system models, encompassing both traditional and cutting-edge approaches to personalization. We begin by exploring classical methods such as Matrix Factorization, which leverages latent factor models to capture user-item interaction patterns through dimensionality reduction. These models have been foundational in collaborative filtering but often struggle with sparsity and cold-start problems. We then examine Deep Learning-based models, which have introduced non-linear representation learning capabilities, allowing systems to model complex user preferences and item attributes more effectively. Despite their power, these models typically rely on static datasets and lack adaptability in dynamic environments. Moving beyond traditional techniques, we highlight the transformative potential of Large Language Models (LLMs) in the context of recommender systems. LLMs bring advanced reasoning, multi-modal understanding, and contextual interpretation to the recommendation process, enabling conversational and context-aware recommendations. Finally, we introduce Agentic Recommenders—a novel class of systems inspired by the Agentic AI paradigm. These systems feature autonomous, goal-driven agents that can adapt to evolving user preferences, simulating user behaviour, and orchestrating multi-agent collaboration. In this paper, we have also addressed the comparative strengths and limitations of these approaches—highlighting, for example, the scalability and transparency of classical methods, the expressive power but data dependence of deep learning models, and the natural language fluency yet domain adaptation challenges of LLMs. Additionally, we examine how Agentic Recommenders, while promising in their goal-oriented flexibility and real-time adaptability, currently face implementation challenges, including system complexity, latency, and a lack of standardized design frameworks. By integrating these diverse methodologies, our paper provides a holistic survey of recommender systems and positions Agentic AI as a promising direction for achieving more adaptive, human-like, and smart recommendations.

The remainder of this paper is divided into the following sections. Section II provides an overview of the definition of a recommender system, including common paradigms. Section III covers a range of existing recommender system models. Section IV explains what Agentic AI is in relation to other AI paradigms and discusses common problems associated with it. Section V goes over Agentic Recommender System models. Section VI is our conclusion and closing remarks.

2 An Overview of Recommender Systems

2.1 Recommendation Problem Definition

Assume we have our item set, I, which represents all possible items that can be recommended to a user, $u \in U$. The goal of a recommender system is to take a user's data and output a set of target items, $I_t \subseteq I$, such that each $i \in I_t$ is a product that user

u is likely to purchase. The user data inputted into the recommender system to achieve this typically involves the user's rated item history and, depending on the specific approach used to solve this problem, may also include user profile tags, item descriptions, user click data, and other relevant information. Note that each recommendation system model has a different way of quantifying how likely a user is to purchase a specific item, and multiple examples of how this is done are discussed later in section 3.

2.2 Common Paradigms in Recommender Systems

There are two major paradigms in the field of recommender systems: the collaborative filtering paradigm and the content-based filtering paradigm. However, do note that most state-of-the-art recommender systems implement aspects of each. The collaborative filtering paradigm involves outputting personalized recommendations for a user, $\boldsymbol{u}$, by examining the user's past interactions and then identifying users, $\boldsymbol{u_{similar}} \subseteq \boldsymbol{U}$, that have a similar interaction history [3]. The collaborative filtering approach would then proceed to recommend the user, $\boldsymbol{u}$, items that they have not interacted with but that users in $\boldsymbol{u_{sim}}$ have interacted with. The underlying assumption of the collaborative filtering approach is that users with similar preferences will likely share similar tastes [3]. On the other hand, Content-based filtering approaches create a user profile by utilizing tags and keywords to generate personalized recommendations for the user [1]. Similarly, items are also assigned tags and keywords based on their description [1]. Heuristic functions, such as cosine similarity, are used to calculate the similarity score between the user and the item. The items with the highest similarity score are then recommended to the user [1].

3 Traditional Vs LLM-Based Recommender Models

One of the most fundamental and common approaches to address the recommendation problem is matrix factorization (MF) algorithms, which fall under the collaborative filtering paradigm. The MF is a process where the user-item rating data is inputted into a matrix, $M \in R^{Users\ x\ Items}$, where each row vector represents the complete rating history of a single user [4]. Processes such as Singular Value Decomposition are then employed on the matrix to output two matrices, $U \in R^{Users\ x\ k}$ and $I \in R^{k\ x\ Items}$, which when multiplied will yield a matrix, $\widetilde{M} \in R^{Users\ x\ Items}$, that contains predicted ratings [4]. Do note that the two matrices U and I serve as user and item embeddings, k is often referred to as the embedding dimension, and it's much smaller than either of the initial matrix's dimensions [4]. While there are many drawbacks to a simple method, MF still offers decent performance in most real-world scenarios and interpretable results compared to deep learning approaches. The model proposed in [5] suggests a novel data preprocessing method that aims to address data sparsity resulting from a lack of user interaction or rating history, a common issue for matrix factorization (MF) approaches. This method involves calculating the Pearson Correlation Coefficient between user-item pairs and then matching similar users based on their ratings to generate synthetic data to fill up a sparse user-item matrix [5]. Another common approach to address the recommendation problem is the use of deep neural networks (DNNs). The model proposed in [6] is the DNN-LSTM model, which employs a deep neural

network to predict user ratings, and the final recommendation list is generated using a Long Short-Term Memory (LSTM) neural network. The model begins by applying a matrix factorization algorithm to the user-item rating matrix, which outputs the latent user and item matrices. From there, the user and item latent vectors are then input into the DNN [6]. The DNN aims to reconstruct each user's rating vector and, in doing so, provide predictions for unrated items [6]. The association score is then calculated for item-item pairs to identify similar items. This score, along with the predicted ratings, is input into the LSTM [6]. The model proposed in [7] is like the DNN-LSTM model, with the distinction of incorporating content-based knowledge for the recommendation task. A DNN takes in the user and item vectors and aims to predict the user's rating on unseen items; this process is referred to as neural collaborative filtering [7]. The model also proposes the use of GloVe, an NLP technique that represents words as vectors, to create embeddings for text data, such as item reviews [7]. These embeddings are then input into a separate DNN, which provides recommendations based on content knowledge [7]. The final recommendation is based on the outputs of the neural collaborative filtering and the content-based filtering stages [7]. A recent approach to recommender systems that has gained popularity is the integration of Large Language Models (LLMs) into the recommendation process. An example of this approach is the Multi-Modal Recommender System (MMREC) [8], which builds upon the Deep Learning Recommender Model (DLRM) [9]. An LLM is used in this model to summarize the item review data into a single embedding [8]. It is the benefit of using LLMs compared to traditional NLP techniques (e.g. GloVe) to create embeddings, as they can leverage their reasoning abilities to summarize the data into fewer embedding tokens, allowing for smoother processing. The LLM's reasoning ability, alongside its multi-modal ability, is also employed to extract dense features (e.g. price) from user reviews and interpret and summarize image data [8]. Note that summarizing the images as text allows for the LLM to use its reasoning ability to combine the information from both text and image features effectively [8]. The resultant embeddings are then inputted into the DLRM model, which outputs the final candidate list of recommendations [8]. The MMREC model [8] utilized large language models (LLMs) to extract data from text and images, which were then leveraged to enhance the recommendation process. The A-LLMrec model [10] aims to create a conversational recommender system (CRS) by merging the latent space of an LLM with frozen weights with the latent space of a pretrained Collaborative Filtering Recommender System [10]. Moreover, the model also incorporates the much smaller LLM, Sentence BERT, to create embeddings from text reviews; these embeddings are then concatenated with the item data embeddings from the pre-trained recommender [10]. These embeddings are then scaled to match the input dimension of the LLM, ensuring they can be inputted properly. The LLM is then prompted to generate a candidate set [10]. Whereas A-LLMrec aims to modify an LLM and then implement it as a CRS, [11] aims to measure ChatGPT's ability as a recommender system without any fine-tuning. To test ChatGPT on the recommendation task, dialogue fragments from a dataset were input into ChatGPT, and it was asked to complete the fragments; its responses were then recorded [11]. These responses were examined by 190 study participants who were asked to rate each recommendation on a scale of 1-5 in terms of meaningfulness alongside recommendations from other models

[11]. 48% of respondents gave ChatGPT the maximum score, and respondents gave higher scores to ChatGPT than the state-of-the-art CRB-CRS model [11]. CRB CRS is an LLM-based CRS that relies on retrieval methods rather than generative methods [12].

Table 1. A Summary of Recommender System Models

Matrix Factorization Models	Deep Learning Models	Conversational Recommender System
MF-CF [5]	DNN-LSTM [6]	MMREC [8]
	GLoVe-DNN [7]	A-LLMRec [10]
	DLRM [9]	ChatGPT [11]
		CRB-CRS [12]

4 What is Agentic AI?

4.1 Agentic AI Relative to Other AI Paradigms

Recently, within the context of artificial intelligence, the term "Agentic AI" has come to label the newest paradigm shift in the field [2]. We identify three major paradigms in the field of AI: Classical AI, Generative AI, and Agentic AI. Table 1 summarizes the major differences between these three paradigms. Classical AI refers to AI models that are typically integrated as part of a larger system and are responsible solely for specific tasks within that system, such as image recognition or trend prediction [2]. These models typically employ supervised learning approaches and often function by attempting to learn and then predict underlying patterns in data, without regard to context. Moreover, these models are not adaptable to multiple use cases, and they lack autonomy [2]. An example of a classical AI model is a spam email classifier, which is integrated into an email service. Such a classifier would likely be trained using supervised learning approaches on a large data set of emails labelled "spam" or "real", with the model itself learning to classify the emails rather than "understanding" what fundamentally makes email spam; this differs significantly from how both the generative and agentic paradigm's function. Due to this, the model can also be said to be ungeneralizable, as it would be unable to be applied to similar tasks, such as fake review classification, for example.

Compared to classical AI models, generative AI models are more autonomous and adaptable, although they are still employed as a smaller component of a larger system. This paradigm is referred to as "Generative" due to the models' ability to combine pieces of information to generate content, such as text and images [2]. Generative AI models, such as GPT, Llama, or other similar large language models (LLMs), are often initially trained on a diverse and large corpus that encompasses data from various fields [13]. They require less user interaction to accomplish their tasks compared to classical AI models. Generative AI models can be further fine-tuned on smaller datasets for domain-specific optimization, resulting in adaptable and generalizable models. Since

LLMs can leverage the information learned from the large and diverse data corpus and combine it with the information learned from the fine-tuning data, this approach enables more effective domain-specific optimization. Additionally, most generative AI models are built using the Transformer architecture, which allows these models to fundamentally understand the relationship between words and their meanings relative to each other by mapping them to vectors in a vector space [13]. This gives generative AI models the ability to understand context, unlike classical AI models. An example of the generative AI paradigm is a spam email classifier within an extensive email system that employs an LLM (e.g., GPT) for the email classification task. Using GPT, as opposed to classical AI approaches, would enable a model that better understands the language-related characteristics of an email that lead to it being classified as spam. Note that the same base GPT model can be finetuned on different smaller datasets and be used in other applications.

Agentic AI, compared to generative AI, enables models that are more adaptable and possess increased autonomy. Agentic AI models utilize Goal-Oriented Architectures to allow switching between tasks and facilitate "multi-tasking" [2]. This goal-oriented architecture enables the model's primary goals to be further broken down into subgoals, the relationships between which are then identified. This information is leveraged to choose which task or set of tasks the model should prioritize at any given moment. This eliminates the need for user supervision or input, allowing for a fully autonomous AI system. Additionally, reinforcement learning is a method of training machine learning models, where the model is trained to achieve specific goals by maximizing the reward or feedback when interacting with its environment (i.e., data). Reinforcement learning techniques are integral to agentic AI models, as they're employed during training and fine-tuning to adapt the model to various environmental or data variations. Adaptive control mechanisms, which can adjust the model's parameters (i.e., the weights of the layers) in response to data fluctuations, are also employed to enhance the models' adaptability. An example of the Agentic AI paradigm is an autonomous spam identification chatbot system integrated into an external email service. The chatbot would be able to flag emails that qualify as "spam" and provide a brief explanation to the human user, if prompted through text. Often, Agentic AI models are implemented as multi-agent systems, where each agent is a fine-tuned large language model (LLM), an external tool (e.g., a call to an external API), or a classical AI model. In such multi-agent systems, there's often one LLM agent that serves as the "brain" and is responsible for calling all other agents.

Table 2. Difference Between AI-Based Paradigms

Paradigm	**Context Awareness**	**Autonomy**	**Adaptability**
Classical AI	None of the models numerically learn underlying patterns in data and solely base its output on the propagation of that pattern.	None of the models require explicit programming or instructions to function.	None models can only accomplish a task or set of functions that they've been trained to do.
Generative AI	Transformer architecture enables models to comprehend the relationship	Limited, the model requires some explicit instructions to function	Some adaptability, such as finetuning LLMs (e.g. GPT) on smaller datasets,

	between words, providing limited context awareness.	and can usually be prompted.	allows them to leverage their existing large knowledge base in a new domain.
Agentic AI	Leverages generative AI agents' ability and other types of agents in multi-agent systems to comprehend context.	Fully autonomous through implementing a goal-oriented architecture.	Reinforcement learning and adaptive control mechanisms are implemented to allow for adaptability.

4.2 Problems with Agentic AI models

While Agentic AI offers valuable improvements over other paradigms, such as improved adaptability and increased autonomy, it is essential to note that a multitude of problems still arise when implementing Agentic AI models. This is due to agentic AI still being a relatively new paradigm within the field of AI. Some of the current common issues encountered when deploying agentic AI models include:

- Unpredictability when handling user input:
  - Ignoring user feedback, as LLMs often do.
  - Being unable to follow multi-step instructions.
  - Misinterpreting user input to mean something other than what the user intended.
- Complexity in Internal Function:
  - LLMs are often used as the "brain" of an agentic AI model, and there remains considerable ambiguity about their inner workings.

5 Agentic AI Recommender Systems

5.1 Agentic Recommender System Models

The approach taken in implementing most Agentic AI models is to implement them as multi-agent systems, with agents serving in specialized roles. The same is true for Agentic Recommender Systems. Figure 1 introduces a general framework for creating a Conversational Agentic Recommender System, where each agent is an LLM (e.g., GPT, Llama, Bard) assigned a specific role. One of the roles proposed in this framework is the Manager role; this refers to the LLM agent responsible for coordinating and calling on other agents when needed [15]. Another role is the interpreter LLM agent, which is responsible for creating a simplified embedding of the user's input prompt [15]. This framework also proposes a goal planner agent that aims to extract the intention behind the user's request (e.g., looking for recommendations, conversing, etc.) [15]. Additionally, the framework proposes Reflector, Responder, and Feedback agents, which enable the model to reflect on its output and determine if it matches the

user request, modify the response to better align with the user request, and incorporate user feedback to update the knowledge base [15]. Other roles proposed include User agent, item agent, and searcher, which are responsible for retrieving user information, item information, and information from the internet as required [15]. In [15], the authors have provided a useful framework for the design and analysis of Agentic Recommender Systems. Implementing all the proposed agents into a single Agentic Recommender System would likely lead to the model having problems, such as inference latency. Thus, the approach taken by most models is to minimize the number of agents and focus on designing the agents themselves. For example, the model proposed in [16], InteRec Agent, employs an LLM agent as the manager, which will call on a set of external tools to aid in the recommendation process. These tools include an information query module, an item retrieval module, and an item ranking module. The information query module is responsible for retrieving information about an item from the recommender system's knowledge base, which is accomplished using an SQL query tool [16]. The item retrieval module is responsible for generating a list of candidate items to recommend to a user based on their learned preferences. This is achieved through a similar SQL query tool for items with discrete attributes. However, for items with non-discrete attributes (e.g., a user requests a movie like Pulp Fiction), a tool that pairs similar items based on latent embeddings is used. The item ranking module employs a one-tower architecture to rank the candidate set based on the user profile.

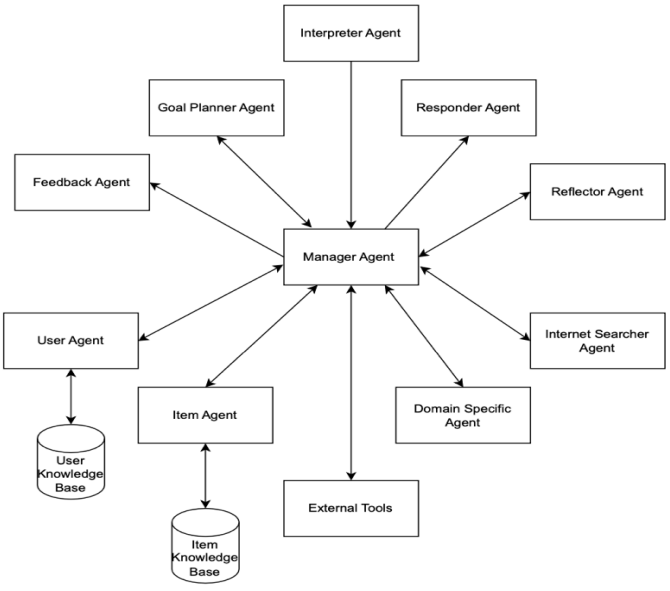

Fig. 1. A General Framework for Agentic Recommender Systems

Although the model proposed in [17], Chat-CRS, has the distinction of relying on the leading LLM Agent to make the recommendation rather than employing it as a manager, external tools are then called on by the LLM agent to retrieve knowledge to

aid in the recommendation task. Two external tools are employed in this model: a knowledge retrieval agent and a goal-planning agent. The knowledge retrieval agent undertakes the task of extracting knowledge on items in the user's requested recommendation category. This is achieved by first extracting keywords from the user's query, then searching through the model's relational knowledge base, and selecting a set of candidate Entity-Relation-Entity triples. The leading LLM agent then deduces which of these relations is relevant, and then knowledge from that relation is extracted to aid in the recommendation. The goal-planning agent aims to extract the user's intention from their dialogue and then create a plan to achieve that goal [17]. This agent is an LLM that has been further fine-tuned on a dataset of annotated conversational recommender system dialogue instances. Whereas [16] and [17] proposed models with a main LLM agent and external tools used to support it, [18] proposed the multi-agent CRS model, which takes a more integrated approach. Within the model's multi-agent act planning module, there are three agents specifically designed to serve users: a question-answer (QA) agent to answer users' questions about the items, a chat agent, and a recommender agent [18]. Each one of these agents has a memory module, a profile module and an action module [18]. The memory module stores each agent's past dialogue history. In contrast, the profile module contains instructions for how each agent should act, and the action module generates the response based on the contents of the other two modules [18]. Moreover, the multi-agent act planning module also contains a planner agent that selects the best response among the three different agents given the user's request [18]. Furthermore, this model also features a User-Feedback Reflection agent responsible for updating the memory modules of each agent with user preferences [18]. If the user provides feedback about a response that does not match their query, that feedback is directed only to the planner agent [18].

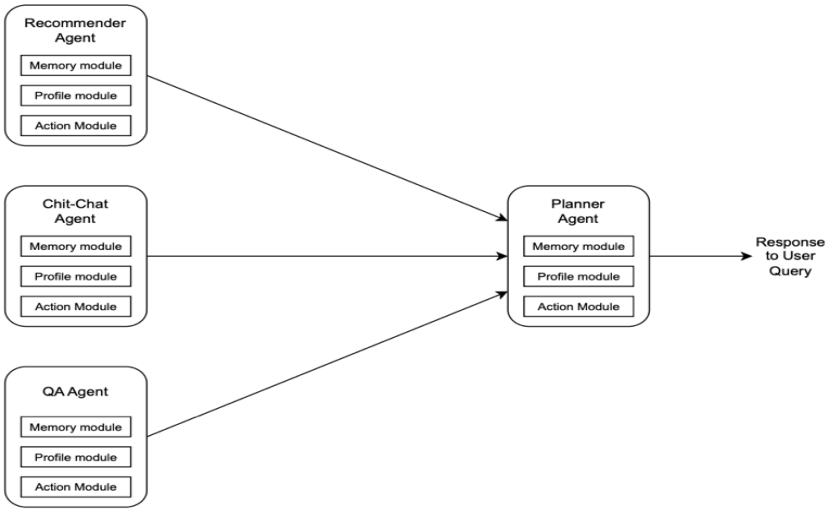

Fig. 2. Architecture of the Multi-Agent CRS Model [18]

Another model which uses a similar approach to the one proposed in [18] is the model proposed in [19], RAH (Recommender System – Assistant – Human). This model consists of only two agents: a recommender system agent that generates a list of

candidate items and an assistant agent [19]. The assistant agent further narrows down this list by retrieving the user's profile from its memory and then generating a tailored list [19]. The model also features a built-in feedback mechanism, where, after the user has been presented with a list of recommendations, they are prompted to provide feedback (e.g., indicating "like" or "dislike") [19]. The assistant will then store this feedback, which will influence the user's future recommendations [19].

Another approach to implementing Agentic AI in recommender systems is to use agents to simulate users' decision-making processes and generate personalized recommendations. The ToolRec model [20] takes this approach. The interaction history of a real user, the one seeking the recommendation, is inputted into an LLM agent [20]. The agent then extracts a key attribute from that interaction history, and an external retrieval tool is used to find items within the knowledge base that match that attribute [20]. The model also invokes a ranking tool to rank the candidate set, and this process of attribute extraction, retrieval, and ranking is repeated until the simulated user is satisfied with the recommended items [20]. An interesting aspect to note is that chain of thought prompting is used on the LLM to simulate the user [20]; this process involves prompting an LLM model with very specific, step-by-step instructions. Another paper which proposes the use of LLM agents to simulate users is [21], although in this case, the simulated users aim to interact with a recommender system to improve its adaptability. Like [18], each agent has a memory module, a profile module, and an action module, and the profile module would be unique for each agent, allowing for a variety of simulated personalities [21]. This also reflects the multitude of applications that Agentic AI has within the field of recommendation systems. Authors in [22] aimed to use LLM agents to model items. An LLM can generate these items through finetuning, or they can be constructed through user prompts [22]. The user will then interact with this simulated item, and through that, the LLM will begin to learn the user's preferences and tastes [22]. The item agent can share this user feedback with a recommender agent, which leverages this information to make a final recommendation to the user [22].

Table 3. Summary of Agentic Recommender Systems

Multi-Agent Systems	User Simulators	Item Simulators
InteRec Agent [16]	ToolRec [20]	Rec4Agentverse [22]
Chat CRS [17]	Agent4Rec [21]	
Multi-Agent CRS [18]		
RAH [19]		

5.2 Comparative Analysis of Recommender System Models

In this section, we present a comparison of the key models in our survey, as shown in Table 4. Generally, the matrix factorization and deep learning models can be considered traditional approaches, while agentic and conversational recommenders are considered the current state of the art. The shift towards agentic recommenders has led to models that are significantly better at leveraging LLMs' reasoning abilities to adjust objectives on the fly, consider user feedback, and enhance autonomy. However, there is a considerable lack of work focusing on the adaptability of agentic recommenders, where adaptability refers to a model's ability to handle interactions that it has not been specifically

trained for. Adaptability is considered a major feature of the agentic AI paradigm. It plays a pivotal role in creating recommenders that can provide better and more "human" recommendations. Therefore, we identify the lack of adaptable agentic recommenders as a research gap.

Table 4. Comparison of Surveyed Models

Model Name	**Approach**	**Strengths**	**Weaknesses**
MF-CF [5]	Matrix Factorization	•Addresses the data sparsity problem. •Results are interpretable.	•Unable to learn users' preferences.
DNN-LSTM [6]	Deep Learning based	•Combines predicted ratings with content-based filtering • The use of an LSTM enables the learning of users' preferences.	•LSTM networks have a relatively shorter "memory" compared to state-of-the-art LLMs. •Lacks item metadata.
GLoVe-DNN [7]	Deep Learning based	•Combines both collaborative and content-based filtering.	•The use of DNNs to analyze the text embeddings leads to weaker performance compared to CRSs.
MMREC [8]	Conversational Recommender System	•LLMs are used to enhance the word embedding process, compared to GLoVe. •The LLM used can extract hidden features from text data (e.g., the Price of an item).	•LLMs' reasoning ability was not leveraged in the recommendation task.
A-LLMRec [10]	Conversational Recommender System	•Merges collaborative filtering and LLM-based approaches. •Incorporates review metadata.	• The LLM lacks domain-specific knowledge and fine-tuning.
ChatGPT [11]	Conversational Recommender System	•Potential to use generative and retrieval-based methods for recommendation.	•Lack of domain-specific knowledge.
InteRec Agent [16]	Agentic (Multi Agent Based)	•Can switch objectives on the go, as LLM is used as a brain, not a recommender. •LLM calls on external tools for ranking and retrieval.	•Lack of agent adaptability. •Not generalizable to multiple use cases/domains.
Chat CRS [17]	Agentic (Multi Agent Based)	•Goal goal-planning agent is employed to allow for on-the-go objective switching.	•Lack of adaptability to a variety of user prompts.

		• The user can directly interact with the recommender system.	
Multi-Agent CRS [18]	Agentic (Multi Agent Based)	•Multiple responder agents, leading to increased adaptability. • User feedback is used to update parameters.	•Large number of agents can lead to issues with inference latency
ToolRec [20]	Agentic (User Simulator)	• A recommendation is given to the user by simulating the user's behaviour. • External tools are utilized to support the recommendation process.	•Lack of user feedback mechanism.

6 Conclusion and Future Work

The introduction of the Agentic AI paradigm to recommender systems allows for more adaptable and autonomous recommender agents. Unlike current CRSs, Agentic recommenders can learn on the go using reinforcement learning techniques, creating recommender systems that can identify users' ever-evolving preferences and refine their strategies in real-time. This causes Agentic recommenders to be more adaptable to different situations compared to CRSs. Moreover, unlike the CRS model's, Agentic recommender systems require little to no direct programming or instruction, leading to more autonomous recommenders. This opens the door to recommenders that can shift their approaches and goals on the fly, leading to a more reactive user experience. The shift towards Agentic recommenders paves the way for more intelligent, independent, and reactive recommender systems, which will significantly improve users' experience.

In this paper, we synthesize various architectures proposed in recent literature and provide a comparative framework that evaluates their design, capabilities, and limitations. Importantly, we identify a notable research gap in the adaptability of existing agentic models—despite adaptability being a foundational tenet of Agentic AI. Most current implementations remain domain-specific, lack dynamic generalization, and struggle with integrating real-time user feedback. As such, future work should focus on (1) developing robust evaluation benchmarks for measuring adaptability and autonomy in recommender systems, (2) optimizing inference latency and system complexity in multi-agent architectures, and (3) exploring hybrid frameworks that combine LLM reasoning with classical and deep learning components to achieve scalable, explainable, and efficient recommendations.

References

1. Z. Fayyaz, M. Ebrahimian, D. Nawara, A. F. Ibrahim, and R. Kashef, "Recommendation Systems: Algorithms, Challenges, Metrics, and Business Opportunities,"

Applied Sciences, vol. 10, no. 21, p. 7748, Nov. 2020, doi: https://doi.org/10.3390/app10217748.

2. D. B. Acharya, K. Kuppan and B. Divya, "Agentic AI: Autonomous Intelligence for Complex Goals—A Comprehensive Survey," in IEEE Access, vol. 13, pp. 18912-18936, 2025, doi: 10.1109/ACCESS.2025.3532853.

3. R. Kumar, T. Dey and S. Das, "Empowering Recommendations: Survey of LLM-Based Recommendation Systems," 2025 8th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech), Kolkata, India, 2025, pp. 1-6, doi: 10.1109/IEMENTech65115.2025.10959591.

4. Y. Zhang, "An Introduction to Matrix Factorization and Factorization Machines in Recommendation System, and Beyond," arXiv preprint arXiv:2203.11026, Mar. 2022. doi: 10.48550/arXiv.2203.11026.

5. R. Barathy and P. Chitra, "Applying Matrix Factorization In Collaborative Filtering Recommender Systems," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2020, pp. 635-639, doi: 10.1109/ICACCS48705.2020.9074227.

6. S. K. Rachamadugu, J. R. Dwaram and K. R. Patike, "Recommender System based on Deep Neural Network and Long Short Term Memory," 2021 International Conference on Computing, Networking, Telecommunications & Engineering Sciences Applications (CoNTESA), Tirana, Albania, 2021, pp. 50-55, doi: 10.1109/CoNTESA52813.2021.9657131.

7. Y. Zhang, L. Wang, and M. Li, "Enhanced E-commerce Recommender System Based on Deep Learning," in Proceedings of the 2024 International Conference on Artificial Intelligence and Data Science (ICAIDS '24), New York, NY, USA: ACM, 2024, pp. 123–130. doi: 10.1145/3659677.3659747.

8. J. Tian, Z. Wang, J. Zhao and Z. Ding, "MMREC: LLM Based Multi-Modal Recommender System," 2024 19th International Workshop on Semantic and Social Media Adaptation & Personalization (SMAP), Athens, Greece, 2024, pp. 105-110, doi: 10.1109/SMAP63474.2024.0002.

9. M. Naumov et al., "Deep Learning Recommendation Model for Personalization and Recommendation Systems," arXiv preprint arXiv:1906.00091, 2019. [Online]. Doi: https://doi.org/10.48550/arXiv.1906.00091.

10. S. Choi et al., "An Efficient All-round LLM-based Recommender System," in Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD '24), 2024, pp. 1395–1406. doi: https://doi.org/10.1145/3637528.3671931.

11. M. Kouki et al., "ChatGPT as a Conversational Recommender System," Proc. 17th ACM Conf. Recommender Systems (RecSys '23), Sep. 2023, pp. 1–10. doi: 10.1145/3627043.3659574.

12. A. Manzoor and D. Jannach, "Towards Retrieval-based Conversational Recommendation," arXiv preprint arXiv:2109.02311, 2021. doi: 10.48550/arXiv.2109.023.

13. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," arXiv preprint arXiv:1706.03762, Jun. 2017. doi: https://arxiv.org/abs/1706.03762.

14. C. Han et al., “AI Agents Under Threat: A Survey of Key Security Challenges and Opportunities,” ACM Comput. Surv., vol. 57, no. 7, Art. no. 182, Feb. 2025. doi: 10.1145/3716628.
15. I. D. S. Portugal, P. Alencar and D. Cowan, "An Agentic AI-based Multi-Agent Framework for Recommender Systems," 2024 IEEE International Conference on Big Data (BigData), Washington, DC, USA, 2024, pp. 5375-5382, doi: 10.1109/BigData62323.2024.10825765.
16. X. Huang et al., “Recommender AI Agent: Integrating Large Language Models for Interactive Recommendations,” ACM Trans. Inf. Syst., Apr. 2025. doi: 10.1145/3731446,
17. C. Li et al., “Incorporating External Knowledge and Goal Guidance for LLM-based Conversational Recommender Systems,” arXiv preprint arXiv:2405.01868, 2024. doi: 10.48550/arXiv.2405.01868.
18. X. Huang et al., “Recommender AI Agent: Integrating Large Language Models for Interactive Recommendations,” ACM Trans. Inf. Syst., Just Accepted, Apr. 2025. doi: 10.1145/3731446.
19. Y. Shu, H. Zhang, H. Gu, P. Zhang, T. Lu, D. Li, and N. Gu, "RAH! RecSys-Assistant-Human: A Human-Centered Recommendation Framework with LLM Agents," arXiv preprint arXiv:2308.09904, Aug. 2023. doi: https://arxiv.org/abs/2308.09904.
20. Y. Zhao, J. Wu, X. Wang, W. Tang, D. Wang, and M. de Rijke, "Let Me Do It For You: Towards LLM Empowered Recommendation via Tool Learning," in Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), Washington, DC, USA, Jul. 2024, pp. 1796–1806. doi: 10.1145/3626772.3657828.
21. A. Zhang, Y. Chen, L. Sheng, X. Wang, and T.-S. Chua, "On Generative Agents in Recommendation," in Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), Washington, DC, USA, Jul. 2024, pp. 1807–1817. doi: 10.1145/3626772.3657844.
22. J. Zhang, K. Bao, W. Wang, Y. Zhang, W. Shi, W. Xu, F. Feng, and T.-S. Chua, "Prospect Personalized Recommendation on Large Language Model-based Agent Platform," arXiv preprint arXiv:2402.18240, Mar. 2024. doi: https://arxiv.org/abs/2402.18240.