Empirical analysis of Reinforcement Learning to alleviate the new user problem in e-commerce Session Based Recommender Systems.

Mees Lindhout, 500772613

June 19, 2024

Master Thesis Tool Development
Master Digital Driven Business
University of Applied Sciences Amsterdam
Accomplished in cooperation with AI Systems BV

Supervisor:	Diptish Dey	Doctor of Philosophy
Second Supervisor:	Frederik Situmeang	Doctor of Philosophy

Contents

1 Introduction

Recommender Systems (RS) have become indispensable tools for navigating the vast array of choices available in various domains, including e-commerce, content streaming, and job matching platforms (Lian et al., 2017; J. Lu et al., 2015; Schafer et al., 2001). The inception of these systems can be traced back to pioneers such as Jeff Bezos and Amazon, which initially focused on book recommendations (Linden et al., 2003). Bezos notably emphasized the breadth of choices in books, stating, "There are more items in the book category than there are items in any other category, by far," highlighting the necessity for effective recommendation mechanisms to guide consumer choices amidst numerous options (Huddleston Jr, 2024; Martinez, 2017).

Among the techniques employed in RS, content-based filtering and collaborative filtering have emerged as the most well-known approaches, each with its distinct strengths and limitations (Park et al., 2012; Roy & Dutta, 2022). Content-based filtering recommends items by analyzing their inherent properties (features), whereas collaborative filtering leverages user interaction data to identify patterns and preferences (Breese et al., 2013; Koren, 2008). Consequently, hybrid systems, which combine elements of both content-based and collaborative filtering, have gained traction. This combination is not merely about preference but rather a strategic approach to overcoming inherent challenges such as the cold start problem (Çano & Morisio, 2017; Su & Khoshgoftaar, 2009).

The cold start problem, a significant hurdle in RS, arises when there is insufficient data to make accurate recommendations. This issue is particularly pronounced at the initiation of the system or with the introduction of new items or users (Wei et al., 2017). For instance, Facebook must manage new users and new posts, determining which posts to display to specific users (Meta, 2023). Similarly, an online grocery market must decide how to recommend a new product that has not yet been purchased (Slaff, 2020). Various methods have been researched to mitigate the cold start effects for both new users and new items across different domains (Lee et al., 2019; Man et al., 2017; Volkovs, Yu, & Poutanen, 2017; Volkovs, Yu, & Poutanen, 2017).

This study focuses on session-based RSs (SBRS), a subset of RS that recommends items based on a user's current session, which is a sequence of interactions with the system. SBRS is particularly relevant in e-commerce, where users often have short sessions and make quick decisions (Quadrana et al., 2017). The cold start problem remains a challenge in SBRS, as new users or items may not have enough data to generate accurate recommendations. This study aims to investigate the cold start problem in SBRS and explore potential solutions to mitigate its effects.

1.1 Research Questions

The research question that best encapsulates the problem statement is:

How can the cold start problem in session-based recommender systems be mitigated?

To address this question, the following sub-questions are proposed:

1. What types of cold start problems exist in SBRS?
2. What are possible solutions to mitigate the cold start problem in SBRS?
3. How could Reinforcement Learning be applied to mitigate the cold start problem in SBRS?
4. How does Reinforcement Learning perform in a hybrid SBRS?

1.2 Theoretical Angle and Methodological Approach

The study will adopt a theoretical angle that combines the principles of SBRS, the cold start problem, and RL. SBRS focuses on recommending items based on a user's current session, while

the cold start problem pertains to the challenges faced when there is insufficient data to make accurate recommendations. RL is a machine learning technique that enables a model to learn through trial-and-error interactions with an environment. The study will explore how RL can be applied to mitigate the cold start problem in SBRS and evaluate its performance in a hybrid SBRS. An experimental approach will be employed to evaluate the effectiveness of the proposed solutions.

1.3 Contribution to the Practice Community and Academic Contribution

The study aims to contribute to the practice community by exploring a novel approach to mitigating the cold start problem in SBRS using RL. The technique of RL is relatively unexplored in the context of SBRS, and this study seeks to bridge this gap by investigating its potential applications. The findings of this study can provide valuable insights to practitioners in the field of RS and e-commerce, enabling them to enhance the performance of their recommendation systems. From an academic perspective, this study contributes to the existing body of knowledge on RS and RL. More particularly, the body of knowledge of SBRS and Offline Batch RL sheds light on the effectiveness of the techniques.

1.4 Report Structure

This report is organized into five chapters, each addressing different aspects of the study on the cold start in SBRS. A brief overview of each chapter is provided below:

- **Chapter 1: Introduction** The first chapter introduces the research topic, providing context and background on the importance of the cold start problem. It also outlines the research questions and objectives that guide the study.
- **Chapter 2: Literature Review** The second chapter reviews existing literature related to SBRS, the cold start problem and Reinforcement Learning (RL). It synthesizes findings from previous studies, identifies gaps in the current knowledge, and establishes the theoretical framework for the research.
- **Chapter 3: Methodology** The third chapter describes the methodology used in the study. It includes details on the dataset and how RL is applied with Python to mitigate the cold start problem in SBRS.
- **Chapter 4: Results** The fourth chapter presents the results of the applied RL algorithm. It includes various visual representations of the data and the evaluation metrics and an interpretation of the findings concerning the research questions.
- **Chapter 5: Discussion** The fifth and last chapter discusses the implications of the findings, comparing them with the existing literature. It also addresses the limitations of the study and suggests directions for future research. The chapter concludes by answering the research questions and providing recommendations for practitioners.

2 Literature Review

The literature review in this thesis explores the cold start problem in RS and examines potential solutions to enhance performance in SBRS. The chapter section is divided into several key parts: types of cold start problems, session-based versus conventional RS, performance indicators for SBRS, possible solutions for cold start issues, RL, and deep RL, and finishes with an exploration of offline RL. These topics are essential to understanding the context of the research and the proposed solution. To conclude, the chapter provides a brief conclusion that summarizes the mentioned topics.

2.1 Type of cold start problems

The cold start problem can be split up into *system cold start*, *siem cold start*, and *user cold start* (Lika et al., 2014; Schein et al., 2002). Each of them deals with the problem that there are not enough interactions between users and items which influences the performance of the RS and eventually reduces user satisfaction and experience (J. Lu et al., 2015; Roy & Dutta, 2022) Users represent the persons that interact with an RS, while Items are the objects in the system which the user interacts with. An interaction could be a rating, a like or an action like adding to a basket or adding to favorites. For example, a user could be a person who searches for a hotel on Booking.com. The hotel represents the item and Booking.com hosts the RS where interactions like saving a hotel to a wishlist or paying for a booking are collected.

2.1.1 System cold start

The System Cold start occurs when there are no interactions at all between users and items. Chaimalas et al. (2023) and Silva et al. (2019) define system cold start as *Pure Cold Start* or *Complete Cold Start*, albeit employing different terminology, the essence remains the same. Bernardi et al. (2015) and Kiseleva et al. (2016) mention another definition, the *Continuous Cold Start* problem where only a few interactions over a longer period are available in the system. For example at Booking.com, many users visit and book infrequently because they only have one or two vacations per year, leading to a prolonged cold start and extreme sparsity. There can be argued, that this is a subcategory of an incomplete cold start, as the items have received very few ratings.

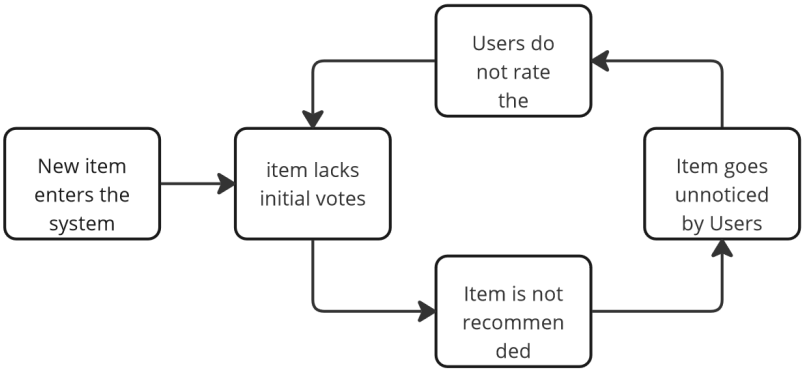
2.1.2 Item cold start

The items in an RS are the products or services that the system recommends to the users. The item cold start problem occurs when the system has no information about the items to recommend to the users. This can happen when the system is new and has not collected any data about the items, or only very limited information. The cold item start problem can be visualized as a vicious cycle as shown in Figure 1. Think for example of a new product that is being sold in a webshop. The new product is added to the list of products and has no interactions with the users because nobody has yet bought it. Therefore the item is not recommended, and the item goes unnoticed by the users. No transaction happens, and the product is also not reviewed. Therefore, the item still lacks information and it stays out of reach for the customer (Bobadilla et al., 2012).

2.1.3 user cold start

The User Cold Start problem occurs when a new user enters the system with limited information available about their preferences (Son, 2016; Yuan & Hernandez, 2023). This situation poses a challenge for the recommendation system as it has only a small number of user interactions

Figure 1: Vicious circle of item cold start



to accurately predict suitable items or with complete new user problem, no interactions at all (Yadav et al., 2020). While there is no specific threshold defined in the literature to classify it as a cold start, typically, a range of 0 to 15 interactions in e-commerce is considered. From a business perspective, the lack of knowledge about the user's preferences makes it difficult to satisfy their needs, and recommending the wrong products could further decrease user satisfaction. Moreover, people are becoming more aware of their privacy, meaning that they are less interested in making an account or registering (Himeur et al., 2022; Zangerle & Bauer, 2023). Additionally, the recent EU Data Act implemented a procedure where the user has to give permission to use their data, which can be a barrier for the RS to collect data (Kop, 2020). All these reasons suggest that RS should be more privacy-focused while still being able to recommend interesting items to satisfy new users.

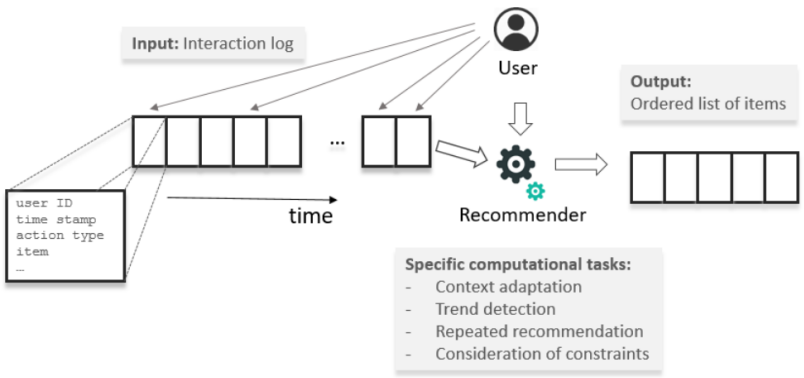
2.2 Session-based RS (SBRS) versus conventional RS

Session-based RSs (SBRS) are RS that only keep track of sessions of 30 minutes maximum instead of a user interaction history of multiple months (Quadrana et al., 2018). Moreover, user IDs are not tracked, meaning that when the customer visits the website the next day, it is seen as a new user. These aspects of SBRS make them suitable for the new user problem of e-commerce websites. SBRSs originate from sequential RS and use the same sequential data structure. Sequential RSs differ fundamentally from conventional matrix completion RSs in their approach to leveraging user-item interactions. While traditional matrix completion methods focus on predicting missing values in a user-item rating matrix, often treating each interaction as an independent event, sequential RSs take into account the order and timing of user interactions. See Figure 2 for a high-level overview of Sequence-Aware Recommendation systems. The sequential data is collected in the interaction log that contains the user ID (which is in SBRS anonymously), timing in a time stamp, the independent event as action type (for example click, add to favorites) and the item that it had the type of interaction with. As output, it gives a list of items to recommend to the user, ordered by descending probability. On an e-commerce website, this could be a list that is shown next to the product specifications list on the product page.

2.3 Performance indicators to measure cold start in SBRS

Performance indicators are essential to evaluate the performance of an RS because they provide a quantitative measure of how well the system is performing. The Mean Reciprocal Rank (MRR) and Hit rate (HIT) are two common performance indicators when a list of items is recommended

Figure 2: High-level overview of Sequence-Aware Recommendation systems (Quadrana et al., 2018)



and the order of the items must be taken into account, which is called ranking (Hernández del Olmo & Gaudioso, 2008; Jannach et al., 2017; Quadrana et al., 2018; Zangerle & Bauer, 2023).

The MRR is a measure used to evaluate the ranking quality of the recommended items. It considers the rank position of the first relevant item in the recommendation list (Voorhees, 1999). The MRR is calculated as the average of the reciprocal ranks of the first relevant item across all queries. This metric provides insight into how quickly a relevant item appears in the recommendation list, thus emphasizing the importance of high-ranking relevant items. The metric ranges between 0 and 1, a higher MRR value indicates better performance, with 1 representing perfect ranking where the first relevant item is always at the top of the list.

HIT, on the other hand, measures whether at least one relevant item appears within the top-N recommendations. It is a binary metric indicating a 'hit' if a relevant item is found within the specified range, otherwise a 'miss'. This metric evaluates the ability of the recommendation system to present at least one relevant item in the top-N list, regardless of its exact position. Similarly to MRR, HIT ranges between 0 and 1, with higher values indicating better performance.

The primary difference between MRR and HIT lies in their focus and insight into the recommendation system's performance. MRR is concerned with the ranking of the first relevant item, providing a detailed view of how well the system prioritizes relevant items at the top of the list. In contrast, HIT assesses the overall effectiveness of the recommendations by simply checking if any relevant item appears within the top-N recommendations. Therefore, while MRR gives more weight to precise ordering, HIT focuses on the presence of relevant items within a given recommendation list length.

In the SBRS, especially during the cold start phase where historical data is sparse or non-existent, both MRR and HIT are crucial. MRR helps in understanding how quickly the system can surface relevant items even with limited data, which is vital for user satisfaction in initial interactions. HIT, meanwhile, ensures that users are presented with at least one relevant recommendation early on, which can significantly enhance the user experience and trust in the system. Thus, these metrics together provide a comprehensive evaluation of the recommendation system's effectiveness during the cold start phase.

2.4 Possible solutions to solve cold start in RS

Extensive analysis has been done to identify solutions that could solve the cold start problem. All details can be can be found in Appendix A. The scope of the analysis was beyond SBRS to get a better understanding of the various ways of solving the cold start problem. This paragraph

summarizes these possible solutions. In general, the solutions can be divided into two strategies: data-driven and model-driven (Yuan & Hernandez, 2023).

Data-driven strategies focus on collecting more data about the user or item. This could be by asking the user to fill in a questionnaire or by collecting data from other external sources (Fernández-Tobías et al., 2016; Sun et al., 2013). These methods can be time-consuming, expensive and can be seen as an invasion of privacy. Therefore it is not always the best solution. Concerning data types, there is a distinction between implicit and explicit data. Explicit data is data that is given by a user explicitly. For example, the number of stars a user gives when he reviews a product or answers a question like 'Did you like the movie?'. Implicit feedback on the other hand is derived from user behavior without explicitly asking the user. This could be a mouse click, the length it takes before someone clicks on a product, or for example purchasing an item again.

Model-driven strategies, on the other hand, focus on the model itself, reducing the need to collect more data. These strategies, such as Meta-Learning by Bharadhwaj (2019) try to use the behavior of a user and then model the RS to adapt to that behavior. For example, if a user clicks on a t-shirt, the RS will recommend more clothing items and not toys. This is a very simplified example, but it shows the idea of Meta-Learning. Overall, model-driven strategies are sometimes more complicated to implement and explainability can be an issue. Meaning that it is not always clear why the RS recommends a certain item.

Therefore, it could be interesting to create a combination of both strategies and having multiple models work together. When multiple models work together, the advantages of one model can compensate for the disadvantages of another model. This is called a hybrid model (Panda & Ray, 2022; Tahmasebi et al., 2021). Explicit feedback is more scarce than implicit feedback in e-commerce websites, therefore it is important to use implicit feedback in the RS. A specific type of model-driven strategy is RL which will be discussed in the next paragraph. This technology is an upcoming technology and has shown promising results in RSs (Afsar et al., 2022; Lin et al., 2024).

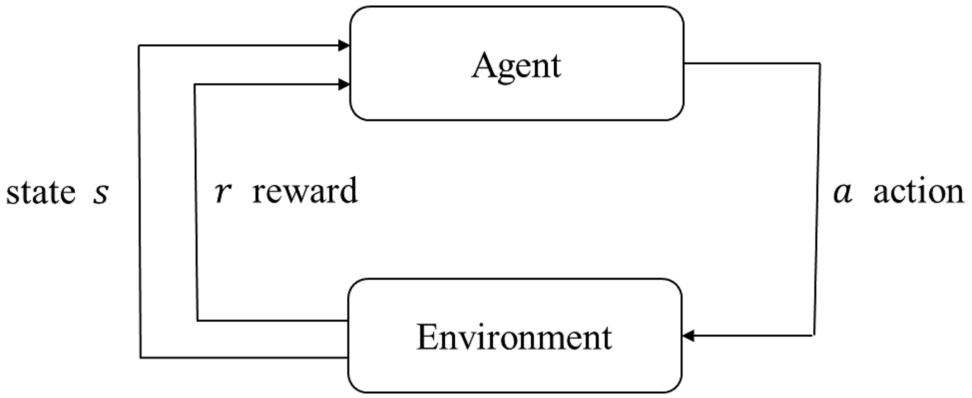
2.5 Reinforcement Learning (RL)

RL is a type of machine learning where an agent learns to make decisions by taking actions in an environment to maximize cumulative rewards (Li, 2018). In other words, it tries to learn from each explicit data point that is left behind by the user on the website and as an action returns a product as a recommendation. The agent takes a step by performing an action (see Figure 3), receives feedback in the form of a reward, and updates its strategy to improve future actions. The learning process is guided by a loss function, which measures the difference between the predicted and actual rewards. This strategy is called the agent's policy. An episode consists of a sequence of steps that ends when the agent reaches a terminal state, after which the agent's performance is evaluated based on the total reward accumulated during the episode. In other words, an RL agent learns to take action under certain conditions by exploring the environment by trial and error. It tries to learn each situation of the cold start problem by looking at the data and then tries to predict the best action to take. The technology originates from a biological standpoint where a baby is learning to walk: when the baby tries to stand up and move, the mom cheers and gives rewards like hugs, making the baby want to try more. Over time, the baby learns to walk better by figuring out which actions get the most cheers and rewards. Similarly, RL determines the different rewards but then with a reward function.

It sometimes looks similar to supervised machine learning, however, RL learns to sequentially make decisions based on the type of steps it takes in an environment, whereas supervised machine learning does not take sequential aspects into account.

These sequential aspects are particularly relevant in RSs, where the order and timing of user interactions play a crucial role in predicting user preferences. Sivamayil et al. (2023)

Figure 3: Reinforcement Learning visual representation (H. Huang et al., 2019)



mentions that RL is, therefore, an interesting research field with much potential and research to be carried out. Therefore, RL has already been extensively studied in the context of RSs and Afsar et al. (2022) and Lin et al. (2024) address that the sequential decision-making process is inherent to recommending items to users. For instance, Choi et al. (2018) used RL in their recommender system, focusing on predicting ratings (1 to 5 stars) of movies with the Movielens dataset. Although this dataset is relatively old and not directly applicable to e-commerce, it demonstrated the feasibility of RL in predicting user preferences. Another example is provided by Giannikis et al. (2024), who attempted to solve the cold-user start problem in RSs using RL techniques. L. Huang et al. (2021) distinguished between cold and warm start on the Movielens and Steam datasets and concluded that their proposed deep RL model works well in both situations. However, it should be noted that they only tested their model on two datasets which were not e-commerce related and were much smaller in size, than a real-world situation. Moreover, SBRSs are not explicitly mentioned in their research, which is a crucial aspect in the context of e-commerce websites.

2.6 Deep Reinforcement Learning

All the mentioned articles utilized deep learning (DL). DL, a subset of artificial intelligence, employs neural networks to model and process enormous sets of data. The emergence of DL has significantly influenced various fields within machine learning, leading to advancements in areas such as object detection, speech recognition, and language translation (LeCun et al., 2015). Mnih et al. (2013), for example, used Deep Q-learning to play Atari games and achieved human-level performance in many of the games. Previously, simple Q learning was used where a Q table was initialized. The Q table is a table where for each state-action observation a Q value is stored. See Figure 4 Deep Q-learning approximates the Q table with a neural network, Tabular RL does not scale to large complex problems such as RS, because the number of possible combinations is too large. There will be too many possible states in memory and it is too slow to learn all the values for each state-action pair.

2.7 Offline RL / Batch RL

There are various types of RL and depending on its use case there must be chosen which type of RL is most suitable. As explained before, RL learns in an environment. However, in some cases, it is not possible to learn in an environment, for example, when the environment is too complex or too expensive to simulate. In such cases, offline RL, also known as batch RL, can be used. Offline RL learns from a fixed dataset of interactions between the agent and the environment, which is collected beforehand. This dataset is used to train the RL model, which then makes

Figure 4: Tabular Q Learning vs Deep Q-Learning (Ganesh & Xu, 2022)

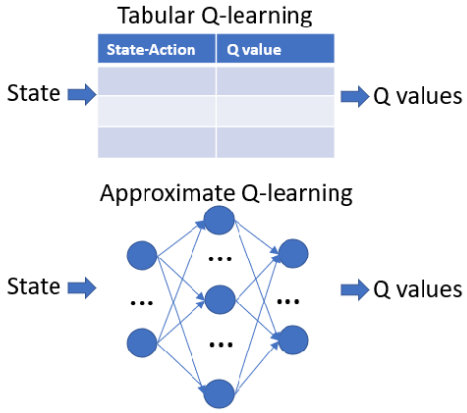
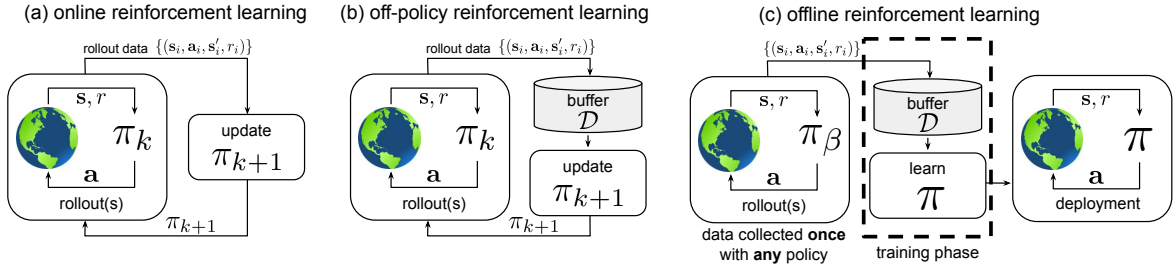


Figure 5: Variations in RL (Levine et al., 2020)



decisions based on the learned policy. Offline RL has been used in various applications, such as robotics, healthcare, and RSs (Levine et al., 2020).

The differences between the three types of RL are illustrated in Figure 5. Online RL learns from interactions with the environment in real-time and immediately uses the feedback to update the policy (for example the Deep Q-learning network). Off-policy RL creates a buffer of state, action rewards, and next-state tuples. The buffer, sometimes also called the replay buffer or experience replay, is used to train the model / Agent. By storing experiences, off-policy methods can reuse past experiences multiple times for learning. This is particularly beneficial because collecting new experiences in a real environment can be expensive or time-consuming as each time the agent has to interact with the environment. Offline RL, on the other hand, learns from a fixed dataset of interactions that were collected from the environment.

In RS these are interaction logs of users who clicked on items or bought items. The dataset acts as a large replay buffer that can be used to train the model. Utilizing this form of RL could therefore be appropriate and will be implemented in this research. The underlying mechanisms of the RL algorithm and its application will be elaborated upon in the next chapter subsection 3.3.

2.8 Conclusion

The literature review has provided an overview of the cold start problem in RSs, focusing on the user cold start problem in SBRS. The review highlighted the importance of MRR and HIT as performance indicators for evaluating SBRS, especially during the cold start phase. Various strategies to address the cold start problem were discussed, including data-driven and model-driven approaches, with a particular emphasis on hybrid models. RL and deep RL were introduced as promising techniques for addressing the cold start problem in RSs, with offline RL offering a viable solution for learning from fixed datasets. The next chapter will delve into the

proposed hybrid RL model, detailing its architecture, training process, and used frameworks.

3 Methodology

3.1 Dataset

The Diginetica dataset is an e-commerce dataset provided by the e-commerce website Diginetica (Diginetica, 2016). It was created as part of the CIKM Cup 2016 competition, which focused on various challenges in the realm of e-commerce search and recommendations. The dataset contains implicit data; views and purchases which will be used to train the RL agent. Both types of interactions are in separate files and are concatenated into one file. Compared to other researchers, the dataset has been filtered to only include sessions with at least two interactions and items that have more than 5 interactions. Different from other researchers, the dataset contains both clicks and purchase transactions because the goal is to recommend items that are clicked on and bought. These two types of interactions are important as they are implicit feedback and can be used to train the RL agent. The data processing is done in Python and the preprocessing module of Malte et al. (2019, July 2/2024) is used to preprocess the data for reliability purposes. To be more specific, the preprocess command of session-rec is used with the default parameters in _diginitica.py. The dataset was split such that the last 7 days of interactions formed the test set, while the preceding interactions were used for training. This temporal split ensures that the evaluation reflects real-world scenarios where the model predicts future interactions based on past user behavior. These datasets can be found in the folder 'premade datasets' in this shared GitHub repository. Afterwards, the file '0. preprocessing Diginetica & EDA.ipynb' takes care of adding the implicit feedback to the dataset. This file also contains an exploratory data analysis of the dataset to understand the data better if needed.

3.2 Base model: Variable Sequence and Time Aware Neighborhood (VSTAN)

Ludewig et al. (2021) and Shehzad and Jannach (2023) mention that it is common within RS research to choose suboptimal baseline models or to inadequately tune hyperparameters. Researchers often select popularity-based or random models as a baseline and then propose a more complicated neural network-based model and conclude that their proposed model performs better. But when simple models are also taken into account, and all models are correctly hypertuned, the results are different. Ludewig et al. (2021) and Shehzad and Jannach (2023) conducted a comprehensive benchmark and found that less complex models often outperform more complicated neural net-based models. They compared item k-NN models, with neural network-based RS, such as GRU4REC, and observed that simpler RS outperformed GRU4REC on all used datasets. Therefore, for this research, an item kNN-based model is used. More, specifically, (Shehzad & Jannach, 2023) concludes that Variable Sequence and Time Aware Neighborhood (VSTAN) emerged as the most consistent performer across knn-based methods on all e-commerce datasets. It achieved the highest MRR and HIT on the RETAIL dataset and demonstrated strong competitive performance on the DIGI dataset. Thus, VSTAN is chosen as the baseline for its balanced effectiveness across different datasets.

The VSTAN builds upon the STAN (Sequence and Time Aware Neighborhood) model, which was originally proposed by Garg et al. (2019). The STAN method itself extends the Session-based k-Nearest Neighbors (SKNN) approach by integrating temporal and sequential information into the recommendation process. In an e-commerce context, consider a user who is browsing a website for electronics. The SKNN approach would recommend items based solely on the similarity of the user's current session to past sessions, without considering the sequence in which the items were viewed or the time when they were viewed.

However, with STAN, if the user has recently looked at a series of smartphones and then a laptop, STAN will prioritize recommendations for accessories related to smartphones over those for laptops, reflecting the user's most recent interests. Thus, STAN accounts for the recency of items within the current session. Moreover, VSTAN further enhances STAN by

incorporating techniques from the VSKNN (Variable Sequential k-Nearest Neighbors) model which was proposed by Jannach and Ludewig (2017). They included an Inverse Document Frequency (IDF) weighting scheme and a sequence-aware item scoring procedure. The IDF weighting scheme helps to emphasize items that are less common across sessions. In the context of recommendations, this means giving more importance to unique or less frequently interacted items, as they might be more indicative of a user's specific interests. The sequence-aware item scoring procedure recognizes the user's journey from general to specific topics. For example, if a user first looks at a smartphone, then a laptop, and finally a laptop bag, the sequence-aware item scoring procedure will prioritize recommending laptop bags over smartphones, as the user's interest has shifted from general to specific items.

3.2.1 hyperparameters

As Ludewig et al. (2021) already performed hyperparameter tuning based on the hit rate metric and the same dataset, the hyperparameters are taken from their research. The hyperparameters are as follows and can also be found in the session-rec framework which is later discussed in this chapter (Malte et al., 2019, July 2/2024):
k: 100
sample_size: 1000
similarity: 'vec'
stan:
lambda_spw: 4.9
lambda_snh: 80
lambda_inh: 9.8
vsknn:
lambda_ipw: 4.9
lambda_idf: 5

3.3 Proposed RL Model: Offline Batch RL

As has been explained in Chapter 2, the proposed model must deal with the complete cold user problem. Therefore the model cannot take into account any user features such as age or region. The model will focus on observed user behavior in the form of keeping track of which items the user has interacted with. The idea is best explained by following the example and understanding how a Markov Decision Process works.

3.3.1 Markov chain as base of RL

The theory of a Markov Decision Process is the base of Offline RL and has the following six components (Lange et al., 2012; Levine et al., 2020; Xiao & Wang, 2021):

1. **State** $\mathcal{S}$: This is observed from the environment by the agent. The proposed model has a state of length n. For example, when $n = 5$, the state might look like: *[item1, item232, item429, item1, item349]*. At time $t = 0$, when the user has not yet interacted with any items, the list will be filled with zeros. Thus, $s_{t=0} = [0, 0, 0, 0, 0]$. Each time, the user interacts with a new item, the oldest item id gets removed from the list and all elements move one step to the left. The latest item that has been interacted with will always be at the end of the state space list.

   Action $\mathcal{A}$: The action $a_t \in \mathcal{A}$ represents the item ID that the agent predicts. The agent chooses the item ID with the highest Q-value, which is obtained by training the RL offline agent. As the user interacts with more items, the state space accumulates more item id

combinations. The maximum number of different state spaces can equal the number of unique item IDs in the entire training dataset raised to the power of the history length n. Consequently, the state space can grow very rapidly. Therefore, the proposed algorithm has a parameter *n_history* that limits the state space.

2. **Reward Function** $\mathcal{R}$: The agent must learn during training which items are good and which are not suitable to recommend. The reward function determines the appropriateness of actions taken in a given state. Therefore, $r_t = \mathcal{R}(s_t, a_t)$. Designing a reward function is more of an art than a scientific approach and also depends on the goal of the recommender system. RecSys (2023) provides an example where the objective was to prevent the recommender system from suggesting too many similar items. They modeled a reward function to track the characteristics of an item, such as whether it contains sugar and used this to avoid recommending too many sweet items. The proposed RL agent has a reward function designed to learn which items are optimal to recommend in a cold start situation. The reward function has been configured to give a reward of 3 to products that have been clicked and 10 if the product is bought. The function takes into account if the user has clicked multiple times on the product in the future. This means that the whole episode is checked if the product occurs later in the session and what kind of interaction happened.

3. **Discount Factor** γ: The discount factor $\gamma \in [0, 1]$ is used to control the influence of future rewards. Due to the sequential nature of RL, it weighs future rewards of steps in the episode to learn the optimal steps. When $\gamma = 0$, no future rewards are considered, and the RL problem tends to resemble a Bandit system (Wiering & Van Otterlo, 2012). The discount factor can be seen as a parameter on how much the products weigh that are interacted with by the customer at the end of its session.

4. **Offline Dataset** $\mathcal{D}$: The offline dataset $\mathcal{D}$ contains historical interactions. A set of interactions or steps taken is referred to as an episode. Ending an episode is termed 'terminating'. $\mathcal{D}$ needs to be created from the training set and should have the following structure to enable the agent to learn: SessionId, State, Action, Reward, and NextState.

3.3.2 hyperparameters

The programmed RL agent is built in an object-oriented way and has the following hyperparameters and some extra parameters that are used for training the agent on a dedicated server with a GPU.:

n_history: 5 As mentioned in the previous paragraph, this is the size of the state.
reward_dict: {0:3, 1:10} With a dictionary, the reward function can be determined. It is currently set to a reward of 3 for clicks and a reward of 10 for purchases.
event_key: 'Type' If the dataset has a different column name than 'Type' to store the type of interaction, it can be changed with this parameter.
mode: 'training' This parameter sets the RL agent in training mode, after the training has been finished, the model will be saved. When the mode is changed to 'evaluation', the agent could load the trained model with a special file_path parameter.
num_episodes: 3000 This parameter limits the agent to not train the full dataset and can be useful for parameter tuning. It allows the user to experiment with different training durations to find the optimal number of episodes for effective learning.
batch_size: 64 This parameter sets the size of each mini-batch used during training. A batch size of 64 means that the agent will update its Q-values based on the average error calculated from 64 episodes at a time. This helps in stabilizing the training process and allows the agent to learn more generalized policies.
target_update_freq: 1000 This parameter defines how frequently the target network is updated.

After every 1000 episodes, the weights of the target network are updated to match the weights of the current Q-network. This technique helps to stabilize training by providing a consistent target for the Q-learning updates.
memory: 100_000 The memory or buffer size is the maximum number of episodes that can be stored in the replay buffer.
learning_rate: 0.0003 The learning rate is a hyperparameter that controls the step size the agent takes when updating the Q-values. A learning rate of 0.0003 determines the magnitude of updates to the Q-values during each training step. Smaller learning rates typically result in more stable learning, while larger rates can speed up training but may cause instability.
gamma: 0.99 This parameter is the discount factor, which determines the importance of future rewards. A gamma value of 0.99 means the agent will prioritize future rewards almost as highly as immediate rewards. This encourages the agent to learn long-term strategies that maximize cumulative rewards over time.
dataset_name: 'diginetica' By defining this parameter, a separate folder is created for the trained agent
custom_wandb_note: This parameter is optional, and could be useful to add notes to the logging interface wandb which is later explained in this chapter.

3.4 Python libraries

This section describes the Python libraries that have been used to implement the proposed model. The libraries are essential for the development of the model and provide the necessary tools for training and evaluating the model. The most notable libraries used in this study are Gym, PyTorch, session-rec framework, and wandb which will be discussed in the following subsections. Besides these libraries, other standard data science libraries, such as matplotlib and pandas have been used to preprocess the data and to visualize the results.

3.4.1 Gym

The Gym library is an open-source toolkit developed by OpenAI for developing and comparing RL algorithms. It provides a standardized interface for various environments, which allows for consistent benchmarking and comparison of RL models. Gym's simplicity and versatility have made it a popular choice in both academic research and industrial applications. The core of Gym revolves around the concept of environments, which are essentially simulations in which an RL agent interacts. Each environment is characterized by its state space, action space, and reward structure. The state space defines all possible states the environment can be in, the action space outlines the possible actions the agent can take, and the reward structure provides feedback to the agent based on its actions. Each time an episode with steps has been completed, the environment is reset and a new episode is started. In other words, it follows each item interaction of the session and when the session finishes, the episode is finished and the environment resets.

Gym does not have a premade RS environment available and existing environments lacked documentation or were last updated five years ago (Ie et al., 2019; Rohde et al., 2018). A specific custom environment has been developed that takes into account the n history parameter. An environment has always an action space and an observation space variable. The action space stores all the possible actions (item IDs) of the recommender system. The observation space size is equal to n history

3.4.2 Pytorch

The DQN model is implemented using the PyTorch library, which is an open-source machine learning library developed by Facebook's AI Research lab. PyTorch provides a flexible and efficient framework for building deep learning models and is widely used in both research and industry (Paszke et al., 2017). The library provides support for GPU acceleration, allowing for faster training and inference on compatible hardware. PyTorch's extensive documentation and active community make it a popular choice for deep learning practitioners and researchers.

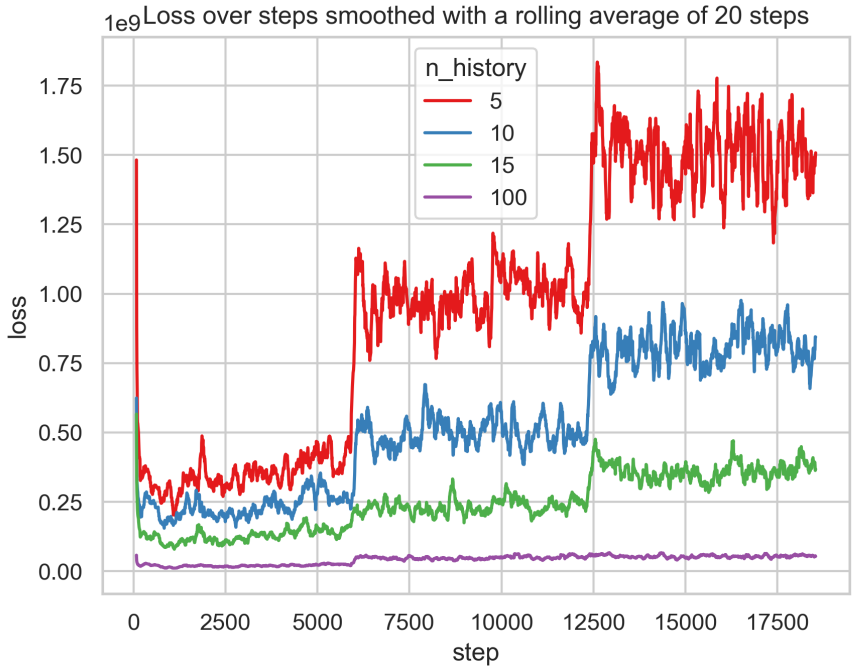
3.4.3 session-rec framework

The session-rec framework is a GitHub repository initiated by several RS researchers who wanted to easily compare various RS models (Latifi et al., 2021; Ludewig & Jannach, 2018; Ludewig et al., 2019, 2021; Malte et al., 2019, July 2/2024). The goal of the framework was to equally benchmark various SBRSs. The framework has been used to simplify the implementation of the VSTAN model, as the code is available in the repository and has been optimized. Moreover, the framework also contains the evaluation metrics MRR and HIT rate, and a class to create hybrid models. By using this framework, the reliability of the results is increased and the implementation time is reduced. To conclude, other combinations of hybrid models can be tested with the RL agent.

3.4.4 wandb

Wandb is a tool to monitor the training of machine learning models and can visualize in real-time the results in a dashboard. In this study, the tool is used to keep track of the training process of the DQN model and to evaluate the training performance by comparing the loss function. At the first run, the tool asks for an API code and afterwards, it automatically creates a project called RecSys RL. Each time the model is trained, the results are stored in the project and can be viewed in the dashboard.

Figure 6: Loss over steps



4 Results

4.1 Deep Q Learning Agent performance

4.1.1 Training without adjusting reward function

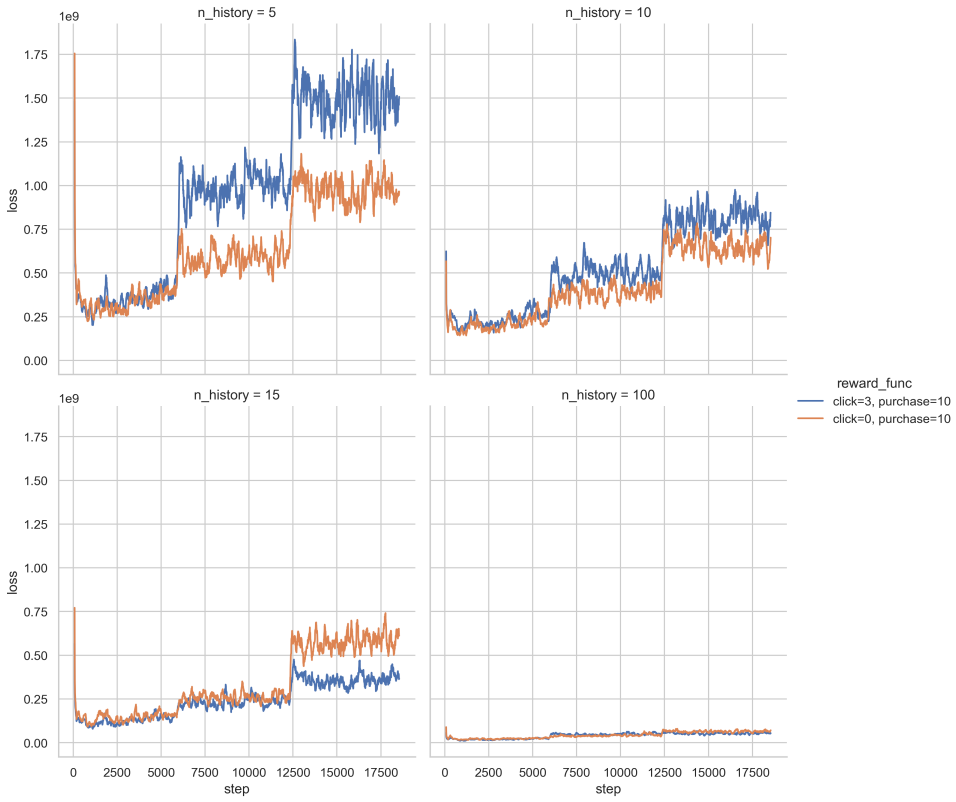
The Deep Q Learning agent was trained using four different scenarios, varying the value of n history (5, 10, 15, and 100). The reward function has been configured to give a reward of 3 for a product that has been clicked on and a reward of 10 if the product has been bought. The agent was trained for 3000 episodes. Each episode (equal to one session) can contain multiple steps, this is why the number of steps and episodes are not equal. In general, the loss increases as the number of steps increases (see Figure 6) which indicates that the loss function of the deep Q learning agent does not converge. This could negatively affect the performance of the SBRS, as a non-converging RL agent indicates that the agent is unable to learn an effective policy from the dataset.

The loss does tend to stabilize after a certain number of steps and afterward increases again. The results show also that as the value of n history increases, the variance in loss generally decreases. A history of 100 shows the least variance as it takes into account more historical interactions with the environment, indicating that the higher the number of n history is

4.1.2 Training with adjusted reward function

The Deep Q Learning agent was trained using four different scenarios; additionally, the reward function was adjusted by setting the click to a reward of 0 and the purchase to 10 to discover the effect of the reward function on the agent. As can be seen in Figure 7, the loss is different. To be more precise, as n history increases, the difference between the two lines also decreases, indicating that the reward function has less impact on the loss as the number of historical interactions increases. This indicates that the loss function demonstrates improved convergence behavior, suggesting that the agent's performance stabilizes more effectively when leveraging a

Figure 7: Loss over steps with different reward functions



dataset with more historical interactions.

4.1.3 Reward per session and episode

Changing the reward function affects the agent training. As can be seen in Figure 8, the cumulative reward over steps is different among the two reward functions. Both reward functions take into account the future rewards within a session. Simply said, if a user gets a lipstick recommended and buys it 5 interactions later, it gets a reward. The n history has no effect on the cumulative reward, as demonstrated in the figure. The line stabilizes and does not decrease as the steps increase. This could indicate that the agent has converged into a local optimum, although not the right one probably, because the loss function line increases slightly and then stabilizes, suggesting that the agent has stopped improving and is stuck in a suboptimal policy without further learning or adjustments. Both reward functions have a similar inflection point around step 1000. As can be seen in Figure 9, most of the steps are given 0 as a reward. This means that the product that has been recommended in the sequence was not clicked on or bought in the future. The distribution follows the behavior of a customer in an e-commerce environment because most of the products are not clicked on or bought. When the reward function is adjusted from click = 3 to click = 0, the distribution of rewards changes notably. Specifically, the frequency of sessions with a reward of 0 increases. This is expected, as the new reward setting results in more sessions where a reward of 0 is assigned. As shown in the figure, the distribution is uneven, which may lead to instability for the deep Q-learning agent. Potential causes of this instability will be explored in the discussion chapter.

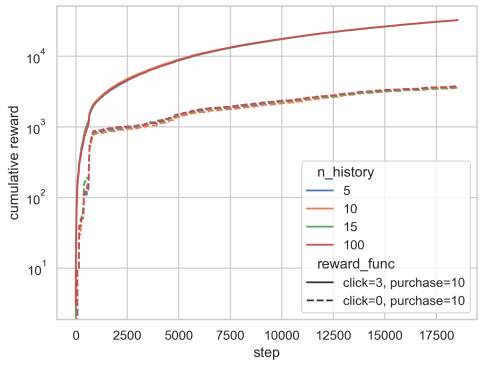


Figure 8: Cumulative reward over steps with different reward functions

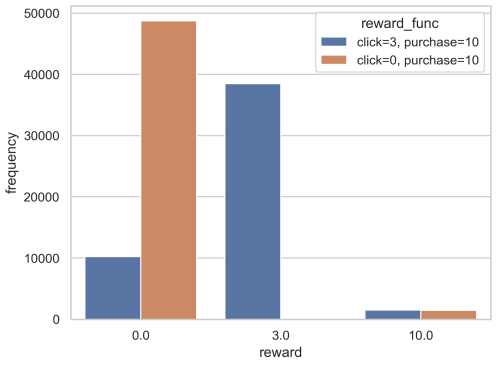


Figure 9: Distribution of given rewards per episode

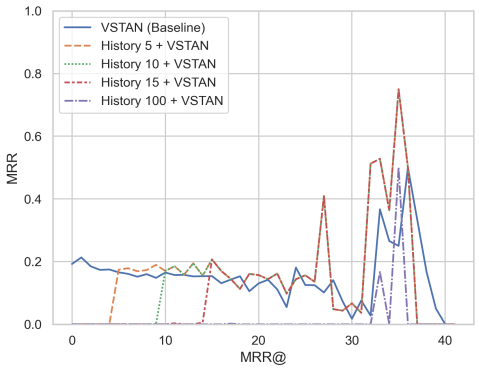


Figure 10: MRR vs recommendation list length

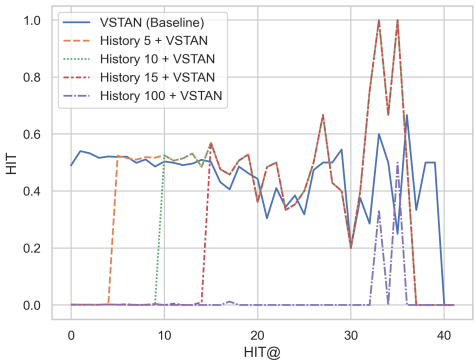


Figure 11: HIT vs recommendation list length

4.2 SBRS performance

The SBRS has two evaluation metrics, MRR and HIT. Both metrics are evaluated over different lengths of recommendation lists ranging from 0 to 40. A hybrid configuration of the SBRS is used, combining the developed RL agent with VSTAN. The agent is used for the sessions that it has been trained for, meaning that the RL agent with n history = 10 takes care of the first 10 interactions of the session. The VSTAN agent takes care of the rest of the session but uses also previous recommendations of the session that were given by the RL agent, due to the nature of how VSTAN works, which has been explained in the previous chapter. The metrics are visualized in Figure 10 and Figure 11, and the tables with all the metrics can also be found in Appendix B. Below, the results are discussed separately per metric. Note that for this evaluation, the non-adjusted reward function is used, as the performance of the DQN did not improve significantly.

4.2.1 MRR

The trained agents have been evaluated, and their performance is visualized in Figure Figure 10. Several noteworthy observations can be made regarding the MRR. The MRR of the baseline model exhibits a slight decrease as the length of the recommendation list increases (indicated at the x-axis). Consequently, beyond length 30, the MRR experiences a sharp increase followed by a decline at 40. Translating this to an RS in an e-commerce environment could indicate that the

baseline model is less accurate in providing recommendations for longer recommendation lists. Looking at the MRR of the first three RL agents (history 5, 10 and 15), all of them have a similar trend when the baseline model kicks in at the specified n history in hybrid mode and exactly behave the same. Looking at the table in Appendix B, the MRR is the same.

One thing to notice is that the baseline (blue) sometimes underperforms compared to the hybrid models except n = 100 as the list length increases. Note that no parameters have been changed in the baseline model so that all four hybrid models can be equally compared. For example, at 27, all the hybrid models outperform the baseline model. Translating these findings to the context of an RS within an e-commerce environment suggests that the baseline model demonstrates diminished accuracy when generating recommendations as the length of the recommendation list extends. This observation implies a potential limitation in the baseline model's ability to maintain recommendation relevance at increased list lengths.

4.2.2 HIT

The hit rate shows similar trends as the MRR. The RL agents have a score of almost 0 during each session until the cutoff rate (equal to their respective set history length), meaning that the baseline model performs better than the RL agents. The n = 100 agent does show a very slight increase in performance at 17, but this is only temporary and a very small number (1.205%). One notable observation is that the baseline model consistently underperforms compared to the hybrid model at cutoff rates 32, 33, 34, and 35. Particularly, at cutoff rates 33 and 35, the hybrid model achieves the highest scores with a hit rate of 100%. This suggests that the hybrid model, which combines the RL agent with VSTAN, outperforms the baseline model in terms of recommendation accuracy. However, reaching a HIT of 100% and similarly to MRR having sometimes exact same values for the first three RL agents indicates that the RL agents act inappropriately. This could be caused by the non-converging loss function of the RL agent, as discussed earlier.

4.3 Scalability and generalizability

The scalability of the SBRS has been covered by not using any sample methods that reduce the dataset size or the number of sessions. Although the baseline model, VSTAN, does use samples, as this is the nature of the model, it still performs similarly to non-sampling methods. Hyperparameter tuning by (Shehzad & Jannach, 2023) assured that performance does not significantly change. Furthermore, when designing the RL model, scalability was a key consideration. Tensors were utilized to store large amounts of data efficiently, allowing for seamless scalability while leveraging the power of the GPU. This design choice ensures that the SBRS can handle substantial amounts of data without compromising performance with regard to training time. Concerning generalizability, the model has been designed to handle different datasets that include various types of interactions, such as clicks, add to cart, add to favorites and more because the reward function can be adjusted. This design choice ensures that the model can effectively adapt to different scenarios and datasets, making it suitable for a wide range of applications in the field of recommendation systems. The various hyperparameters must be checked for each dataset, as they can have an impact on the performance of the model.

5 Discussion

The empirical analysis showed that the hybrid version of VSTAN combined with batch RL outperformed the baseline algorithm VSTAN on HIT rate and MRR when longer lists are recommended but heavily underperformed when shorter lists are recommended. This could be caused by the fact that the Deep Q Learning algorithm did not converge. According to X. Chen et al. (2024) and Nair et al. (2020), it remains exceptionally difficult to train a policy from an offline dataset. X. Chen et al. (2024) explains that a static dataset hinders the algorithm's ability to explore and discover high-reward state space regions if these are not adequately represented in the dataset. Meaning that the algorithm may be fundamentally incapable of uncovering these rewarding regions. The rewarding regions are the regions where the user interacts with the system. They further explain that RS, which, unlike traditional offline RL applications, has access to extensive offline datasets from sources like MovieLens, GoodReads, and Amazon. These datasets tend to better capture real-world user preferences effectively due to the explicit feedback data. However, SBRS are unique due to their reliance on implicit feedback, such as mouse clicks, which complicates their integration into the state space. Only using historical item ids and using the type of interaction as a reward, as this study did, may not be sufficient to provide the agent with the necessary information to make accurate recommendations. Arulkumaran et al. (2017) mentions that the discrete action space of the RL agent could be too large, meaning that there are too many products to recommend. Moreover, it could also be due distribution shift, which is a fundamental challenge for Offline RL (X. Chen et al., 2023, 2024) During distribution shift, the agent performs exactly the steps that it has learned on old data that is not relevant anymore for new users that would like to use the system. The recently published paper of Y. Wang et al. (2024) also mentioned the same issue when they used offline RL for cold-start recommendations.

This study contributes to the literature by investigating the use of batch RL in SBRS and can conclude that it is difficult to train a policy from an offline dataset. The relatively new field of batch RL is still in its infancy and has not yet been widely adopted in the SBRS domain. By publishing the results of this study, the research community can learn from the challenges faced in this study and potentially avoid them in future research. Moreover, the code of the model has been implemented into the session-rec framework which makes it easier for future researchers to use parts of the model in their research.

Due to time constraints, only one dataset was used for the empirical analysis which may increase the variance of the results. Moreover, the Deep Q Learning algorithm did not converge, which may have affected the results.

Future research could investigate the use of other batch RL algorithms or methods to increase the probability of convergence of the agent. Moreover, future research could also investigate the use of other datasets to determine if the results are consistent across different datasets. Avenues for future research include investigating the use of different reward functions and state spaces to provide the agent with more information to make accurate recommendations. Combining this together with a dataset with a lot of implicit feedback could potentially improve the performance of the agent.

6 Conclusion

In recent years, RSs have become popular algorithms to address the issue of information overload for users of online applications by recommending a selection from a large number of items available. RSs are also widely used in e-commerce to recommend products that users may be interested in adding to their shopping cart. It is important to recommend the right products to the right user as quickly as possible, as failure to do so can lead to customer churn, causing dissatisfaction for both the user and the e-commerce provider. This study aimed to investigate the problem of user cold-start, which occurs when the e-commerce provider has minimal or no information about a new customer. Recent privacy regulations, such as the EU AI Act, make it challenging for e-commerce providers to store user information. Therefore, this research focused on SBRS and did not utilize any user features. An empirical analysis was conducted using the Diginetica e-commerce datasets comparing VSTAN as the baseline algorithm with a hybrid version of VSTAN combined with batch RL.

As stated in the research question, the study aims to answer the research question *"How can the cold start problem in session-based recommender systems be mitigated?"* The study proposed four sub-questions to address the main research question:

1. What types of cold start problems exist in SBRS?
2. What are possible solutions to mitigate the cold start problem in SBRS?
3. How could Reinforcement Learning be applied to mitigate the cold start problem in SBRS?
4. How does Reinforcement Learning perform in a hybrid SBRS?

The study found that there are multiple types of cold start problems in SBRS, including new user cold start, new item cold start, and new session cold start. This study focused on the cold user start problem, where the e-commerce provider has no information about the user. The study proposed a hybrid solution to mitigate the cold start problem in SBRS by applying batch RL to learn the optimal policy for recommending items to new users. The data was pre-processed and a specific reward function was designed to train the RL model to recommend items to a new user who is likely to interact with. During the empirical research, the model did not converge which means that the RL model was not able to learn the optimal policy from the offline dataset. Various factors could have contributed to this, such as the complexity of the environment and model, the reward function, or the dataset. The trained agent was then used to recommend items to new users in the test set in a hybrid configuration with VSTAN. The results showed that the hybrid model does not perform better than the baseline model. Therefore, the study concludes that the batch RL model did not mitigate the cold start problem in the SBRS.

Bibliography

Afsar, M. M., Crump, T., & Far, B. (2022). Reinforcement learning based recommender systems: A survey. https://doi.org/10.48550/arXiv.2101.06286
359 citations (Google Scholar) [2024-06-01].

Agarwal, D., & Chen, B.-C. (2009). Regression-based latent factor models. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 19–28. https://doi.org/10.1145/1557019.1557029
310 citations (Crossref) [2024-04-18].

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Processing Magazine*, *34*(6), 26–38. https://doi.org/10.1109/MSP.2017.2743240
2227 citations (Crossref) [2024-06-12].

Bernardi, L., Kamps, J., Kiseleva, J., & Müller, M. J. (2015, August 5). *The Continuous Cold Start Problem in e-Commerce Recommender Systems*. arXiv: 1508.01177 [cs]. https://doi.org/10.48550/arXiv.1508.01177

Bharadhwaj, H. (2019). Meta-Learning for User Cold-Start Recommendation. *2019 International Joint Conference on Neural Networks (IJCNN)*, 1–8. https://doi.org/10.1109/IJCNN.2019.8852100
37 citations (Crossref) [2024-02-07].

Bobadilla, J., Ortega, F., Hernando, A., & Bernal, J. (2012). A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems*, *26*, 225–238. https://doi.org/10.1016/j.knosys.2011.07.021
350 citations (Crossref) [2024-03-05].

Breese, J. S., Heckerman, D., & Kadie, C. (2013, January 30). *Empirical Analysis of Predictive Algorithms for Collaborative Filtering*. arXiv: 1301.7363. https://doi.org/10.48550/arXiv.1301.7363
8319citations (Google Scholar) [2024-03-17].

Briand, L., Salha-Galvan, G., Bendada, W., Morlon, M., & Tran, V.-A. (2021). A Semi-Personalized System for User Cold Start Recommendation on Music Streaming Apps. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2601–2609. https://doi.org/10.1145/3447548.3467110
19 citations (Crossref) [2024-04-18].

Buffelli, D., Gupta, A., Strzalka, A., & Plachouras, V. (2023, August 16). *Is Meta-Learning the Right Approach for the Cold-Start Problem in Recommender Systems?* arXiv: 2308.08354. https://doi.org/10.48550/arXiv.2308.08354
1 citations (Crossref) [2024-02-13].

Cai, G., & Chen, N. (2018). Constrained Probabilistic Matrix Factorization with Neural Network for Recommendation System. In Z. Shi, E. Mercier-Laurent, & J. Li (Eds.), *Intelligent Information Processing IX* (pp. 236–246). Springer International Publishing. https://doi.org/10.1007/978-3-030-00828-4_24
3 citations (Crossref) [2024-04-17].

Çano, E., & Morisio, M. (2017). Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis*, *21*(6), 1487–1524. https://doi.org/10.3233/IDA-163209
157 citations (Crossref) [2024-03-10].

Chaaya, G., Métais, E., Abdo, J. B., Chiky, R., Demerjian, J., & Barbar, K. (2017). Evaluating non-personalized single-heuristic active learning strategies for collaborative filtering recommender systems. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 593–600. https://doi.org/10.1109/ICMLA.2017.00-96
7 citations (Crossref) [2024-04-17].

Chaimalas, I., Walker, D. M., Gruppi, E., Clark, B. R., & Toni, L. (2023). Bootstrapped Personalized Popularity for Cold Start Recommender Systems. *Proceedings of the 17th ACM Conference on Recommender Systems*, 715–722. https://doi.org/10.1145/3604915.3608820
1 citations (Crossref) [2024-02-13].

Chen, C. C., Lai, P.-L., & Chen, C.-Y. (2023). ColdGAN: An effective cold-start recommendation system for new users based on generative adversarial networks. *Applied Intelligence*, *53*(7), 8302–8317. https://doi.org/10.1007/s10489-022-04005-1
7 citations (Crossref) [2024-04-18].

Chen, X., Wang, S., McAuley, J., Jannach, D., & Yao, L. (2024). On the Opportunities and Challenges of Offline Reinforcement Learning for Recommender Systems. *ACM Transactions on Information Systems*. https://doi.org/10.1145/3661996
1 citations (Crossref) [2024-05-06] Just Accepted.

Chen, X., Yao, L., McAuley, J., Zhou, G., & Wang, X. (2023). Deep reinforcement learning in recommender systems: A survey and new perspectives. *Knowledge-Based Systems*, *264*, 110335. https://doi.org/10.1016/j.knosys.2023.110335
41 citations (Crossref) [2024-05-21].

Choi, S., Ha, H., Hwang, U., Kim, C., Ha, J.-W., & Yoon, S. (2018, January 16). *Reinforcement Learning based Recommender System using Biclustering Technique*. arXiv: 1801.05532 [cs]. https://doi.org/10.48550/arXiv.1801.05532

Cremonesi, P., Tripodi, A., & Turrin, R. (2011). Cross-Domain Recommender Systems. *2011 IEEE 11th International Conference on Data Mining Workshops*, 496–503. https://doi.org/10.1109/ICDMW.2011.57
86 citations (Crossref) [2024-03-14].

Cunha, T., Soares, C., & de Carvalho, A. (2018). Metalearning and Recommender Systems: A literature review and empirical study on the algorithm selection problem for Collaborative Filtering. *Information Sciences*, *423*, 128–144. https://doi.org/10.1016/j.ins.2017.09.050
65 citations (Crossref) [2024-04-10].

Diginetica. (2016, August 5). *CodaLab - Competition*. Retrieved June 18, 2024, from https://competitions.codalab.org/competitions/11161

Elahi, M., Ricci, F., & Rubens, N. (2014). Active Learning in Collaborative Filtering Recommender Systems. In M. Hepp & Y. Hoffner (Eds.), *E-Commerce and Web Technologies* (pp. 113–124). Springer International Publishing. https://doi.org/10.1007/978-3-319-10491-1_12
22 citations (Crossref) [2024-04-09].

Feng, J., Xia, Z., Feng, X., & Peng, J. (2021). RBPR: A hybrid model for the new user cold start problem in recommender systems. *Knowledge-Based Systems*, *214*, 106732. https://doi.org/10.1016/j.knosys.2020.106732
44 citations (Crossref) [2024-02-08].

Fernández-Tobías, I., Braunhofer, M., Elahi, M., Ricci, F., & Cantador, I. (2016). Alleviating the new user problem in collaborative filtering by exploiting personality information. *User Modeling and User-Adapted Interaction*, *26*(2), 221–255. https://doi.org/10.1007/s11257-016-9172-z
90 citations (Crossref) [2024-03-10].

Fu, W., Peng, Z., Wang, S., Xu, Y., & Li, J. (2019). Deeply Fusing Reviews and Contents for Cold Start Users in Cross-Domain Recommendation Systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 94–101. https://doi.org/10.1609/aaai.v33i01.330194
75 citations (Crossref) [2024-02-15].

Ganesh, A., & Xu, B. (2022). A review of reinforcement learning based energy management systems for electrified powertrains: Progress, challenge, and potential solution. *Renewable and Sustainable Energy Reviews*, *154*, 111833. https://doi.org/10.1016/j.rser.2021.111833
102 citations (Crossref) [2024-06-13].

Gantner, Z., Drumond, L., Freudenthaler, C., Rendle, S., & Schmidt-Thieme, L. (2010). Learning Attribute-to-Feature Mappings for Cold-Start Recommendations. *2010 IEEE International Conference on Data Mining*, 176–185. https://doi.org/10.1109/ICDM.2010.129
177 citations (Crossref) [2024-02-15].

Garg, D., Gupta, P., Malhotra, P., Vig, L., & Shroff, G. (2019). Sequence and Time Aware Neighborhood for Session-based Recommendations: STAN. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1069–1072. https://doi.org/10.1145/3331184.3331322
76 citations (Crossref) [2024-06-17].

Giannikis, S., Frasincar, F., & Boekestijn, D. (2024). Reinforcement learning for addressing the cold-user problem in recommender systems. *Knowledge-Based Systems*, *294*, 111752. https://doi.org/10.1016/j.knosys.2024.111752
0 citations (Crossref) [2024-05-01].

Golbandi, N., Koren, Y., & Lempel, R. (2011). Adaptive bootstrapping of recommender systems using decision trees. *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, 595–604. https://doi.org/10.1145/1935826.1935910
108 citations (Crossref) [2024-04-11].

Gonzalez Camacho, L. A., & Alves-Souza, S. N. (2018). Social network data to alleviate cold-start in recommender system: A systematic review. *Information Processing & Management*, *54*(4), 529–544. https://doi.org/10.1016/j.ipm.2018.03.004
108 citations (Crossref) [2024-04-22].

Hernández del Olmo, F., & Gaudioso, E. (2008). Evaluation of recommender systems: A new approach. *Expert Systems with Applications*, *35*(3), 790–804. https://doi.org/10.1016/j.eswa.2007.07.047
145 citations (Crossref) [2024-06-17].

Himeur, Y., Sohail, S. S., Bensaali, F., Amira, A., & Alazab, M. (2022). Latest trends of security and privacy in recommender systems: A comprehensive review and future perspectives. *Computers & Security*, *118*, 102746. https://doi.org/10.1016/j.cose.2022.102746
32 citations (Crossref) [2024-03-01].

Huang, H., Huang, J., Feng, Y., Zhang, J., Liu, Z., Wang, Q., & Chen, L. (2019). On the improvement of reinforcement active learning with the involvement of cross entropy to address one-shot learning problem. *PLOS ONE*, *14*(6), e0217408. https://doi.org/10.1371/journal.pone.0217408
8 citations (Crossref) [2024-06-13].

Huang, L., Fu, M., Li, F., Qu, H., Liu, Y., & Chen, W. (2021). A deep reinforcement learning based long-term recommender system. *Knowledge-Based Systems*, *213*, 106706. https://doi.org/10.1016/j.knosys.2020.106706
67 citations (Crossref) [2024-04-11].

Huddleston Jr, T. (2024). In a 'lost' interview, Jeff Bezos revealed why he chose books as the 'best product' to sell on Amazon [newspaper]. *CNBC: Make It - Power Players*. Retrieved June 16, 2024, from https://www.cnbc.com/2024/01/13/in-1997-jeff-bezos-said-why-he-chose-books-to-sell-on-amazon.html

Ie, E., Hsu, C.-w., Mladenov, M., Jain, V., Narvekar, S., Wang, J., Wu, R., & Boutilier, C. (2019, September 26). *RecSim: A Configurable Simulation Platform for Recommender Systems*. arXiv: 1909.04847 [cs, stat]. Retrieved June 15, 2024, from http://arxiv.org/abs/1909.04847

Jannach, D., & Ludewig, M. (2017). When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 306–310. https://doi.org/10.1145/3109859.3109872
260 citations (Crossref) [2024-05-28].

Jannach, D., Ludewig, M., & Lerche, L. (2017). Session-based item recommendation in e-commerce: On short-term intents, reminders, trends and discounts. *User Modeling and*

User-Adapted Interaction, *27*(3), 351–392. https://doi.org/10.1007/s11257-017-9194-1
86 citations (Crossref) [2024-06-05].

Khan, M. M., Ibrahim, R., & Ghani, I. (2017). Cross Domain Recommender Systems: A Systematic Literature Review. *ACM Computing Surveys*, *50*(3), 36:1–36:34. https://doi.org/10.1145/3073565
87 citations (Crossref) [2024-02-08].

Kiseleva, J., Tuzhilin, A., Kamps, J., Mueller, M. J. I., Bernardi, L., Davis, C., Kovacek, I., Einarsen, M. S., & Hiemstra, D. (2016). *Beyond Movie Recommendations: Solving the Continuous Cold Start Problem in E-commerceRecommendations*. https://doi.org/10.13140/RG.2.1.2488.7288

Kop, M. (2020, June 22). *The Right to Process Data for Machine Learning Purposes in the EU*. https://doi.org/10.2139/ssrn.3653537
2 citations (Crossref) [2024-04-09].

Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 426–434. https://doi.org/10.1145/1401890.1401944
2436 citations (Crossref) [2024-03-17].

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *Computer*, *42*(8), 30–37. https://doi.org/10.1109/MC.2009.263
6961 citations (Crossref) [2024-04-18].

Lamche, B., Trottmann, U., & Wörndl, W. (2014). Active learning strategies for exploratory mobile recommender systems. *Proceedings of the 4th Workshop on Context-Awareness in Retrieval and Recommendation*, 10–17. https://doi.org/10.1145/2601301.2601304
9 citations (Crossref) [2024-04-17].

Lange, S., Gabel, T., & Riedmiller, M. (2012). Batch Reinforcement Learning. In M. Wiering & M. van Otterlo (Eds.), *Reinforcement Learning: State-of-the-Art* (pp. 45–73). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-27645-3_2

Latifi, S., Mauro, N., & Jannach, D. (2021). Session-aware recommendation: A surprising quest for the state-of-the-art. *Information Sciences*, *573*, 291–315. https://doi.org/10.1016/j.ins.2021.05.048
26 citations (Crossref) [2024-06-03].

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539
51562 citations (Crossref) [2024-06-12].

Lee, H., Im, J., Jang, S., Cho, H., & Chung, S. (2019). MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1073–1082. https://doi.org/10.1145/3292500.3330859
160 citations (Crossref) [2024-02-15].

Levine, S., Kumar, A., Tucker, G., & Fu, J. (2020, November 1). *Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems*. arXiv: 2005.01643 [cs]. https://doi.org/10.48550/arXiv.2005.01643
1710 citations (Google Scholar) [2024-05-07].

Li, Y. (2018, November 25). *Deep Reinforcement Learning: An Overview*. arXiv: 1701.07274 [cs]. https://doi.org/10.48550/arXiv.1701.07274

Lian, J., Zhang, F., Hou, M., Wang, H., Xie, X., & Sun, G. (2017). Practical Lessons for Job Recommendations in the Cold-Start Scenario. *Proceedings of the Recommender Systems Challenge 2017*, 1–6. https://doi.org/10.1145/3124791.3124794
15 citations (Crossref) [2024-02-13].

Lika, B., Kolomvatsos, K., & Hadjiefthymiades, S. (2014). Facing the cold start problem in recommender systems. *Expert Systems with Applications*, *41*, 2065–2073. https://doi.org/

10.1016/j.eswa.2013.09.005
398 citations (Crossref) [2024-02-08].

Lin, Y., Liu, Y., Lin, F., Zou, L., Wu, P., Zeng, W., Chen, H., & Miao, C. (2024). A Survey on Reinforcement Learning for Recommender Systems. *IEEE Transactions on Neural Networks and Learning Systems*, 1–21. https://doi.org/10.1109/TNNLS.2023.3280161
4 citations (Crossref) [2024-04-17].

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, *7*(1), 76–80. https://doi.org/10.1109/MIC.2003.1167344
3631 citations (Crossref) [2024-03-17].

Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments: A survey. *Decision Support Systems*, *74*, 12–32. https://doi.org/10.1016/j.dss.2015.03.008
1023 citations (Crossref) [2024-03-17].

Lu, Y., Fang, Y., & Shi, C. (2020). Meta-learning on Heterogeneous Information Networks for Cold-start Recommendation. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1563–1573. https://doi.org/10.1145/3394486.3403207
96 citations (Crossref) [2024-02-15].

Ludewig, M., & Jannach, D. (2018, October 30). *Evaluation of Session-based Recommendation Algorithms*. arXiv: 1803.09587 [cs]. https://doi.org/10.1007/s11257-018-9209-6
211 citations (Crossref) [2024-05-28].

Ludewig, M., Mauro, N., Latifi, S., & Jannach, D. (2019). Performance comparison of neural and non-neural approaches to session-based recommendation. *Proceedings of the 13th ACM Conference on Recommender Systems*, 462–466. https://doi.org/10.1145/3298689.3347041
63 citations (Crossref) [2024-06-18].

Ludewig, M., Mauro, N., Latifi, S., & Jannach, D. (2021). Empirical analysis of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction*, *31*(1), 149–181. https://doi.org/10.1007/s11257-020-09277-1
52 citations (Crossref) [2024-06-03].

Malte, Mauro, N., Latifi, S., & Motta, T. A. (2024, May 27). *Rn5l/session-rec*. Retrieved May 29, 2024, from https://github.com/rn5l/session-rec

Man, T., Shen, H., Jin, X., & Cheng, X. (2017). Cross-domain recommendation: An embedding and mapping approach. *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2464–2470
312 citations (Google Scholar).

Martinez, M. (2017, June 20). *Amazon: Jeff Bezos' juggernaut began with a recommender system that launched a thousand algorithms*. IEEE Computer Society. Retrieved March 17, 2024, from https://www.computer.org/publications/tech-news/research/amazon-jeff-bezos-juggernaut-began-with-a-recommender-system-that-launched-a-thousand-algorithms/

Meta. (2023, June 29). *The AI behind unconnected content recommendations on Facebook and Instagram*. Meta AI. Retrieved February 7, 2024, from https://ai.meta.com/blog/ai-unconnected-content-recommendations-facebook-instagram/

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013, December 19). *Deep Reinforcement Learning for Recommender Systems*. arXiv: 1312.5602 [cs]. https://doi.org/10.48550/arXiv.1312.5602

Nair, A., Dalal, M., Gupta, A., & Levine, S. (2020). Accelerating Online Reinforcement Learning with Offline Datasets. *ArXiv*. Retrieved June 14, 2024, from https://www.semanticscholar.org/paper/Accelerating-Online-Reinforcement-Learning-with-Nair-Dalal/0272b14dd471fe7b81df703a
435 citations (Scemantics Scholar) [2024-06-13].

Obadić, I., Madjarov, G., Dimitrovski, I., & Gjorgjevikj, D. (2017). Addressing Item-Cold Start Problem in Recommendation Systems Using Model Based Approach and Deep Learning. In D. Trajanov & V. Bakeva (Eds.), *ICT Innovations 2017* (pp. 176–185). Springer International Publishing. https://doi.org/10.1007/978-3-319-67597-8_17
6 citations (Crossref) [2024-03-05].

Panda, D. K., & Ray, S. (2022). Approaches and algorithms to mitigate cold start problems in recommender systems: A systematic literature review. *Journal of Intelligent Information Systems*, *59*(2), 341–366. https://doi.org/10.1007/s10844-022-00698-5
14 citations (Crossref) [2024-02-05].

Park, D. H., Kim, H. K., Choi, I. Y., & Kim, J. K. (2012). A literature review and classification of recommender systems research. *Expert Systems with Applications*, *39*(11), 10059–10072. https://doi.org/10.1016/j.eswa.2012.02.038
461 citations (Crossref) [2024-02-08].

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in PyTorch. Retrieved June 18, 2024, from https://openreview.net/forum?id=BJJsrmfCZ

Pu, L., & Faltings, B. (2013). Understanding and improving relational matrix factorization in recommender systems. *Proceedings of the 7th ACM Conference on Recommender Systems*, 41–48. https://doi.org/10.1145/2507157.2507178
10 citations (Crossref) [2024-04-18].

Quadrana, M., Cremonesi, P., & Jannach, D. (2018). Sequence-Aware Recommender Systems. *ACM Computing Surveys*, *51*(4), 1–36. https://doi.org/10.1145/3190616
243 citations (Crossref) [2024-05-03].

Quadrana, M., Karatzoglou, A., Hidasi, B., & Cremonesi, P. (2017). Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 130–137. https://doi.org/10.1145/3109859.3109896
423 citations (Crossref) [2024-06-18].

RecSys, A. C. M. (**typedirector**). (2023, February 20). *Tutorial 2B Hands On Reinforcement Learning for recommender systems* [Video]. Retrieved June 13, 2024, from https://www.youtube.com/watch?v=qJysTu1Xl5U

Rohde, D., Bonner, S., Dunlop, T., Vasile, F., & Karatzoglou, A. (2018, September 14). *RecoGym: A Reinforcement Learning Environment for the problem of Product Recommendation in Online Advertising*. arXiv: 1808.00720 [cs]. https://doi.org/10.48550/arXiv.1808.00720

Roy, D., & Dutta, M. (2022). A systematic review and research perspective on recommender systems. *Journal of Big Data*, *9*(1), 59. https://doi.org/10.1186/s40537-022-00592-5
72 citations (Crossref) [2024-02-07].

Sánchez-Corcuera, R., Casado-Mansilla, D., Borges, C. E., & López-de-Ipiña, D. (2024). Persuasion-based recommender system ensambling matrix factorisation and active learning models. *Personal and Ubiquitous Computing*, *28*(1), 247–257. https://doi.org/10.1007/s00779-020-01382-7
2 citations (Crossref) [2024-04-17].

Schafer, J. B., Konstan, J. A., & Riedl, J. (2001). E-Commerce Recommendation Applications. *Data Mining and Knowledge Discovery*, *5*(1), 115–153. https://doi.org/10.1023/A:1009804230409
860 citations (Crossref) [2024-03-17].

Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 253–260. https://doi.org/10.1145/564376.564421
840 citations (Crossref) [2024-02-15].

Sedhain, S., Menon, A., Sanner, S., Xie, L., & Braziunas, D. (2017). Low-Rank Linear Cold-Start Recommendation from Social Data. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*(1). https://doi.org/10.1609/aaai.v31i1.10758
33 citations (Crossref) [2024-02-15].

Sedhain, S., Sanner, S., Braziunas, D., Xie, L., & Christensen, J. (2014). Social collaborative filtering for cold-start recommendations. *Proceedings of the 8th ACM Conference on Recommender Systems*, 345–348. https://doi.org/10.1145/2645710.2645772
70 citations (Crossref) [2024-02-15].

Seo, Y.-D., Kim, Y.-G., Lee, E., & Baik, D.-K. (2017). Personalized recommender system based on friendship strength in social network services. *Expert Systems with Applications*, *69*, 135–148. https://doi.org/10.1016/j.eswa.2016.10.024
79 citations (Crossref) [2024-04-22].

Sethi, R., & Mehrotra, M. (2021). Cold Start in Recommender Systems—A Survey from Domain Perspective. In J. Hemanth, R. Bestak, & J. I.-Z. Chen (Eds.), *Intelligent Data Communication Technologies and Internet of Things* (pp. 223–232). Springer. https://doi.org/10.1007/978-981-15-9509-7_19
10 citations (Crossref) [2024-03-06].

Shehzad, F., & Jannach, D. (2023, December 27). *Performance Comparison of Session-based Recommendation Algorithms based on GNNs*. arXiv: 2312.16695 [cs]. https://doi.org/10.48550/arXiv.2312.16695

Silva, N., Carvalho, D., Pereira, A. C., Mourão, F., & Rocha, L. (2019). The Pure Cold-Start Problem: A deep study about how to conquer first-time users in recommendations domains. *Information Systems*, *80*, 1–12. https://doi.org/10.1016/j.is.2018.09.001
28 citations (Crossref) [2024-04-09].

Sivamayil, K., Rajasekar, E., Aljafari, B., Nikolovski, S., Vairavasundaram, S., & Vairavasundaram, I. (2023). A Systematic Study on Reinforcement Learning Based Applications. *Energies*, *16*(3), 1512. https://doi.org/10.3390/en16031512
18 citations (Crossref) [2024-04-17].

Slaff, T. (2020, October 13). *Data Science as a Product*. Medium. Retrieved March 18, 2024, from https://blog.picnic.nl/data-science-as-a-product-f383dead5aa4

Son, L. H. (2016). Dealing with the new user cold-start problem in recommender systems: A comparative review. *Inf. Syst.*, *58*, 87–104. https://doi.org/10.1016/j.is.2014.10.001
137 citations (Crossref) [2024-02-27].

Su, X., & Khoshgoftaar, T. M. (2009). A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, *2009*, e421425. https://doi.org/10.1155/2009/421425
2095 citations (Crossref) [2024-03-17].

Sun, M., Li, F., Lee, J., Zhou, K., Lebanon, G., & Zha, H. (2013). Learning multiple-question decision trees for cold-start recommendation. *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, 445–454. https://doi.org/10.1145/2433396.2433451
46 citations (Crossref) [2024-04-09].

Tahmasebi, F., Meghdadi, M., Ahmadian, S., & Valiallahi, K. (2021). A hybrid recommendation system based on profile expansion technique to alleviate cold start problem. *Multimedia Tools and Applications*, *80*(2), 2339–2354. https://doi.org/10.1007/s11042-020-09768-8
54 citations (Crossref) [2024-02-08].

Volkovs, M., Yu, G. W., & Poutanen, T. (2017). Content-based Neighbor Models for Cold Start in Recommender Systems. *Proceedings of the Recommender Systems Challenge 2017*, 1–6. https://doi.org/10.1145/3124791.3124792
34 citations (Crossref) [2024-02-13].

Volkovs, M., Yu, G., & Poutanen, T. (2017). DropoutNet: Addressing Cold Start in Recommender Systems. *Advances in Neural Information Processing Systems*, *30*. Retrieved Febru-

ary 15, 2024, from https://proceedings.neurips.cc/paper_files/paper/2017/hash/dbd22ba3bd0df8f385bdac3e9f8be207-Abstract.html
277 citations (Google Scholar) [2024-03-10].

Voorhees, E. M. (1999). The trec-8 question answering track report. *99*, 77–82.

Wang, C., Zhu, Y., Liu, H., Zang, T., Yu, J., & Tang, F. (2022, June 9). *Deep Meta-learning in Recommendation Systems: A Survey*. arXiv: 2206.04415 [cs]. https://doi.org/10.48550/arXiv.2206.04415

Wang, H., Amagata, D., Maekawa, T., Hara, T., Niu, H., Yonekawa, K., & Kurokawa, M. (2019). Preliminary Investigation of Alleviating User Cold-Start Problem in E-commerce with Deep Cross-Domain Recommender System. *Companion Proceedings of The 2019 World Wide Web Conference*, 398–403. https://doi.org/10.1145/3308560.3316596
12 citations (Crossref) [2024-03-06].

Wang, Y., Ge, Y., Li, Z., Li, L., & Chen, R. (2024). M3Rec: A Context-aware Offline Meta-level Model-based Reinforcement Learning Approach for Cold-Start Recommendation. *ACM Transactions on Information Systems*. https://doi.org/10.1145/3659947
0 citations (Crossref) [2024-06-10] Just Accepted.

Wei, J., He, J., Chen, K., Zhou, Y., & Tang, Z. (2017). Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications*, *69*, 29–39. https://doi.org/10.1016/j.eswa.2016.09.040
468 citations (Crossref) [2024-02-29].

Wiering, M., & Van Otterlo, M. (Eds.). (2012). *Reinforcement Learning: State-of-the-Art* (Vol. 12). Springer. Retrieved June 13, 2024, from https://link.springer.com/10.1007/978-3-642-27645-3

Xiao, T., & Wang, D. (2021). A General Offline Reinforcement Learning Framework for Interactive Recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(5), 4512–4520. https://doi.org/10.1609/aaai.v35i5.16579
29 citations (Crossref) [2024-05-08].

Yadav, U., Duhan, N., & Bhatia, K. K. (2020). Dealing with Pure New User Cold-Start Problem in Recommendation System Based on Linked Open Data and Social Network Features. *Mobile Information Systems*, *2020*, 1–20. https://doi.org/10.1155/2020/8912065
17 citations (Crossref) [2024-04-09].

Yuan, H., & Hernandez, A. A. (2023). User Cold Start Problem in Recommendation Systems: A Systematic Review. *IEEE Access*, *11*, 136958–136977. https://doi.org/10.1109/ACCESS.2023.3338705
0 citations (Crossref) [2024-04-11].

Zangerle, E., & Bauer, C. (2023). Evaluating Recommender Systems: Survey and Framework. *ACM Computing Surveys*, *55*(8), 1–38. https://doi.org/10.1145/3556536
36 citations (Crossref) [2024-04-01].

Zhang, M., Tang, J., Zhang, X., & Xue, X. (2014). Addressing cold start in recommender systems: A semi-supervised co-training algorithm. *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, 73–82. https://doi.org/10.1145/2600428.2609599
87 citations (Crossref) [2024-02-15].

Zhao, C., Sun, S., Han, L., & Peng, Q. (2016). Hybrid matrix factorization for recommender systems in social networks. *Neural Network World*, *26*(6), 559–569. https://doi.org/10.14311/NNW.2016.26.032
4 citations (Crossref) [2024-04-18].

Zhou, T., Shan, H., Banerjee, A., & Sapiro, G. (2012, April 26). Kernelized Probabilistic Matrix Factorization: Exploiting Graphs and Side Information. In *Proceedings of the 2012 SIAM International Conference on Data Mining (SDM)* (pp. 403–414). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611972825.35

Zhu, Y., Lin, J., He, S., Wang, B., Guan, Z., Liu, H., & Cai, D. (2020). Addressing the Item Cold-Start Problem by Attribute-Driven Active Learning. *IEEE Transactions on Knowledge and Data Engineering*, *32*(4), 631–644. https://doi.org/10.1109/TKDE.2019.2891530
70 citations (Crossref) [2024-03-06].

A Appendix A. Possible solutions to solve cold start

This appendix provides an overview of the possible solutions to mitigate the cold start problem in RSs.

A.1 Strategies to solve cold start problems

According to Yuan and Hernandez (2023), there are two main strategies to solve cold start problems in RSs: data-driven strategies and method-driven strategies. Data-driven strategies focus on collecting data about users and items to make better recommendations, while method-driven strategies focus on developing new RS algorithms to make better recommendations. The data-driven strategies include *explicit* data collection and *implicit* data collection, while method-driven strategies include various user-item matrix based RS.

A.1.1 Data-driven strategies: Explicit and Implicit data collection

Data-driven solutions rely on finding extra information or utilizing existing information in the RS to improve recommendations.

Explicit data collection Explicit data collection involves asking users to rate items or provide feedback on items to collect data about their preferences. Either calling it *Active Learning* or *interviewing*.

Active Learning (AL)

Elahi et al. (2014) proposed in 2014 a method called *Active Learning* (AL) that involves the system actively participating in the learning process by selecting informative training data (inputs) that are minimal yet highly relevant. The core principle of AL involves selecting a small set of items for the user to rate or provide feedback on, which then informs the system about the user's preferences. This selection process is crucial and is guided by various strategies aimed at maximizing the informativeness of the feedback received. For example, some methods focus on presenting items that are popular among users or that cover a wide range of interests, aiming to quickly pinpoint the new user's preferences in the RS (Zhu et al., 2020). For example, a movie recommendation system using AL might start by asking you to rate highly popular movies across different genres or movies with multiple different genres. Based on your ratings, it narrows down your preferences and then asks about more specific movies within your preferred genres to refine its recommendations further. The user's feedback is used to update the model and improve the recommendations over time, while the user's effort is minimized by only asking for feedback on the most informative items. AL can be applied through conversation-based strategies in mobile shopping RSs, where the system narrows down user interests by eliciting critiques on general recommendations. This method emphasizes the importance of diversity in the learning process, showing that users tend to prefer recommendations that offer a variety of options (Lamche et al., 2014). To specifically address the user cold start problem, AL can be initiated through an initial questionnaire that gathers the past interests of the users. This method dynamically selects the best questions for each user based on their responses, aiming to quickly build an accurate profile for personalized recommendations (Sánchez-Corcuera et al., 2024). One thing to note is that this study used a sector-specific dataset and did not test their method on for example the Movielens dataset. Table A.1 summarizes the advantages and disadvantages of using AL techniques.

Table A.1 Advantages and Disadvantages of Active Learning

Advantages	Disadvantages
Improves Recommendation Accuracy: AL dynamically updates item representations based on user ratings, which can achieve the same level of accuracy as a fully trained model (Sánchez-Corcuera et al., 2024). **Efficient Data Utilization:** By focusing on high-quality data that represent user preferences, AL efficiently addresses the cold start problem, ensuring that the system learns effectively from limited information (Elahi et al., 2014). **Personalized User Experience:** AL incorporates personalized selection strategies, ensuring diverse user selections for unbiased predictions and enhancing user experience by avoiding repetitive selections (Zhu et al., 2020)	**User Cognitive Cost:** Selecting items for users to rate involves balancing informativeness against user cognitive cost can be challenging, because overloading users with requests for ratings may lead to disengagement (Elahi et al., 2014). **Potential for Inaccuracies:** Focusing on acquiring more ratings for popular items can reduce system error but may also lead to inaccuracies by overrepresenting certain types of items or user preferences (Elahi et al., 2014). **Dependence on Quality Data:** The effectiveness of AL is contingent upon the availability of quality data. In scenarios where such data are scarce or not representative, AL strategies may not perform optimally (Sánchez-Corcuera et al., 2024). **Complexity in Strategy Design:** Designing efficient AL strategies that select informative items without overwhelming users is complex. This involves a delicate balance of personalization, hybridization, and the cognitive load on users (Chaaya et al., 2017).

Interview based

Another explicit method is *interview-based*, where the system asks the user explicit questions to understand their preferences and then makes recommendations based on the answers (Sun et al., 2013). Interview-based approaches could also collect information from users by presenting them with a set of items and asking for their opinions (like, dislike, or unknown). This method can be static, where the list of items is fixed, or adaptive, where the list changes based on user responses. The goal is to minimize the number of items asked while maximizing the informativeness of the user's responses. For example, Pinterest asks to choose topics of interest when the user signs up for an account. In practical terms, this method can be implemented using decision trees, where users are asked to provide their opinions on a set of items or topics. Based on their responses, the system directs the questioning in a way that progressively refines its understanding of the user's preferences (Golbandi et al., 2011; Sun et al., 2013). For example, in a movie recommendation system, a user might be asked whether they like action movies. Depending on their answer, the next question could be about a specific sub-genre or a related question aimed at narrowing down their preferences further. This technique is adaptive, meaning it adjusts the questions based on the user's feedback, allowing for a more accurate and efficient preference learning process. It contrasts with static methods that ask all users the same

set of questions without adapting to individual responses. The adaptiveness of this approach is crucial for quickly converging on a user's preferences, especially when dealing with a large and diverse item space (Sun et al., 2013). Moreover, the use of decision trees and adaptive interviewing processes enables the system to handle heterogeneous preference information [1], optimizing the estimation of userprofiles and the selection of questions. This optimization leads to improved prediction accuracy and a better user interaction experience, as it minimizes the time and effort required from the user (Sun et al., 2013). Table A.2 summarizes the advantages and disadvantages of using AL techniques.

Table A.2 Advantages and Disadvantages of Interview based methods

Advantages	Disadvantages
Improved Prediction Accuracy if interview is conducted: Asking multiple questions, as opposed to a single question, can significantly enhance the prediction accuracy of the recommendation system by gathering more detailed information about user preferences (Lamche et al., 2014; Sun et al., 2013).	**User Cognitive Cost:** The process of asking multiple questions or conducting interviews can increase the time and effort required from users, potentially leading to user disengagement (Sun et al., 2013).
Optimized User Profiling and Question Selection: The approach of conducting interviews or asking multiple questions allows for simultaneous optimization of user profiling and question selection, leading to more personalized and accurate recommendations (Sun et al., 2013).	**Unbalanced Decision Trees:** When users have only seen a small percentage of items, it can lead to unbalanced decision trees, which may affect the efficiency and accuracy of the recommendation system (Sun et al., 2013).
Diversity in Recommendations: By showing diverse items for exploration and focusing on specific preferences when known, the system supports diversity in recommendations, enhancing user experience (Lamche et al., 2014).	

Implicit data collection On the contrary, implicit data collection involves tracking users' interactions with items, such as clicks, views, or purchases, to collect data about their preferences. Implicit data does not solely rely on the platform's data but also on the user's behavior on other platforms.

Cross-Domain

Cross-Domain RS are designed to tackle the user cold start problem by leveraging data from auxiliary domains to enhance RSs in a target domain. This approach is particularly useful in addressing issues of data sparsity and the absence of user interaction history, which are common challenges in providing accurate recommendations for new users (Cremonesi et al., 2011; Khan

[1] taking into account extra information such as demographic data or contextual data such as weather and location

et al., 2017). One method involves the fusion of review texts and item contents with the rating matrix using Stacked Denoising Autoencoders (SDAEs), which helps preserve the semantic information of the items for better recommendation. This model, known as RC-DFM, also employs a multi-layer perceptron to transfer user latent factors between domains, effectively mitigating the cold start and data sparsity problems by utilizing the shared information between the domains (Fu et al., 2019). Another approach utilizes social networking services (SNSs) to learn embeddings of new users through user-user connections, thereby addressing the cold start problem by exploiting the social connections and browsing histories of users. This method trains a NeuMF model based on bridge users from two domains, aiming to provide accurate recommendations for new users without previous interactions (H. Wang et al., 2019). Below in Table A.3 the advantages and disadvantages are described of using Cross-Domain methods. In summary, cross-domain RSs offer a promising avenue for addressing the user cold-start problem by improving recommendation accuracy through the integration of auxiliary domain data. However, the complexity and potential scalability issues associated with these systems present notable challenges that need to be addressed to fully realize their benefits.

Table A.3 Advantages and Disadvantages of Cross-Domain methods

Advantages	Disadvantages
Improved Recommendation Accuracy: Cross-domain RSs can significantly enhance recommendation accuracy by utilizing feedback from auxiliary domains, such as review texts, item contents, and ratings. This is particularly beneficial for new users who have limited interaction history in the target domain (Fu et al., 2019). **Leveraging Diverse Data Sources:** These systems can exploit diverse data sources, such as social network metadata and browsing histories, to infer user preferences more accurately. (H. Wang et al., 2019). **Semantic Information Capture:** By deeply fusing item contents, reviews, and ratings, cross-domain RSs can capture semantic information and user preferences, offering a more nuanced understanding of user needs (Fu et al., 2019).	**Complexity in Implementation and Maintenance:** The implementation and maintenance of cross-domain RSs can be challenging due to their complexity. This includes the difficulty of deeply fusing multiple sources of information and the computational overhead associated with such processes (Fu et al., 2019; H. Wang et al., 2019). **Potential Scalability Issues:** Some approaches within cross-domain RSs may face scalability issues due to their iterative nature, which could limit their applicability in large-scale settings (Sedhain et al., 2017).

Social Networks

The method of using social networking, also called SNRS, to solve the user cold start problem in RSs involves leveraging social network data to model user interests and item preferences (Gonzalez Camacho & Alves-Souza, 2018; Seo et al., 2017). Socal Networks can also be seen as a subtype of cross-domain RS because they utilize data from an auxiliary domain (H. Wang et al., 2019). One specific model, LoCo, tries to solve the problem by employing multivariate linear

regression to optimize social signals for preferences, alongside low-rank parametrization to manage the high-dimensional nature of the social data. This model also incorporates Singular Value Decomposition (SVD) to efficiently process social information, demonstrating significant improvements in recommendation performance over traditional methods (Sedhain et al., 2017, p. 2). Additionally, the method of determining user similarity through features like Linked Open Data (LOD) and collaborative filtering within social networks aids in clustering similar users, thereby improving the RS (Yadav et al., 2020, p. 18). Moreover, this hybrid recommendation system framework exemplifies the practical application of social networking data in the music industry. It performs extraction of user music preferences from platforms like Facebook, combined with the calculation of similarities between music items using DBpedia (Yadav et al., 2020). Below in Table A.4 the advantages and disadvantages are explained.

Table A.4 Advantages and Disadvantages of Social Network-based methods

Advantages	Disadvantages
Significant Improvement in Recommendation Quality: Leveraging social networking data, such as Facebook page likes, has been shown to yield up to a 3-fold increase in mean average precision (mAP) and up to 6-fold enhancements in Precision@k and Recall@k metrics compared to other cold-start recommenders (Sedhain et al., 2014).	**Potential for Sub-optimal Solutions:** There is a risk of sub-optimal solutions due to the complexity of accurately modeling user preferences and the potential for overfitting in models like LoCo (Sedhain et al., 2017).
Enhanced User Profiling: Social network features, including demographic information and user preferences extracted from platforms like Facebook, enable more accurate user profiling. This facilitates the development of personalized recommendations by understanding user interests more deeply (Yadav et al., 2020).	**Time-consuming Optimization:** The process of integrating and optimizing social network data within RSs can be time-consuming. This includes the challenges of dealing with large datasets and ensuring that the recommendation model accurately reflects user preferences (Sedhain et al., 2017).
Addressing the Pure New User Problem: The integration of Linked Open Data (LOD) and social network-based features allows for the construction of detailed user profiles even when no interaction data of the user available, effectively mitigating the complete new user cold-start problem (Yadav et al., 2020).	**Dependency on Social Network Data Availability:** The effectiveness of this approach is contingent upon the availability and accessibility of social network data. Changes in social network policies can limit access to this data, potentially reducing the effectiveness of the recommendation system (Yadav et al., 2020).
Scalability and Improved Accuracy: The LoCo model, which incorporates social information, demonstrates substantial improvements over state-of-the-art recommenders through scalable learning methods like randomized SVD, indicating that social information can be effectively scaled for large datasets (Sedhain et al., 2017).	**Privacy concerns** People are becoming more aware of their privacy, meaning that they start to hide their public (Himeur et al., 2022; Zangerle & Bauer, 2023). Moreover, the recent EU Data Act implemented a procedure where the user has to give permission to use their data, which can be a barrier for the RS to collect data (Kop, 2020).

A.1.2 Method-driven strategies

Yuan and Hernandez (2023) mentions three ways of method-driven strategies: *meta-learning*, *deep learning* and textitmatrix factorization. Method driven strategies try to improve the RS instead of adding extra information to the system.

Meta Learning (MeLa)

Meta-learning, specifically applied to solve the user cold-start problem in RSs, operates by

preparing a model that can quickly adapt to new users with minimal interaction data. This approach is grounded in the principle of learning to learn, where the model is trained not just on the task of recommending items but also on the meta-task of adapting to new users based on a few examples of their preferences (Bharadhwaj, 2019). One popular meta-learning algorithm used in this context is Model-Agnostic Meta-Learning (MAML). MAML trains a model on a variety of learning tasks, aiming to find a parameter initialization that can be effectively fine-tuned with a small number of gradient updates on a new task. In the case of RSs, these tasks are akin to predicting preferences for a set of users, and the new task is to predict preferences for a new user. The model thus learns to generalize across tasks (users) and can quickly adapt to a new user by adjusting its parameters based on a few interactions, such as item ratings or consumption history (Lee et al., 2019). However, despite its effectiveness in academic settings, the practical deployment of meta-learning for the cold-start problem faces challenges. These include scalability issues, latency in updating model parameters for each new user, and the computational cost of maintaining and updating a model for each user's unique preferences. These limitations suggest that while meta-learning can address the cold-start problem, its real-world application may be constrained by operational factors (Buffelli et al., 2023). Moreover, recent research indicates that when properly tuned, standard deep learning models can achieve comparable or even superior performance to meta-learning models for the cold-start problem (Buffelli et al., 2023; C. Wang et al., 2022). In summary, meta-learning offers a promising solution to the user cold-start problem in RS by enabling rapid adaptation to new users with limited data. However, its practical application is hampered by scalability and computational efficiency issues, and alternative approaches using deep learning and representation learning may provide viable solutions (Bharadhwaj, 2019; Buffelli et al., 2023; Lee et al., 2019; C. Wang et al., 2022). The main advantages and disadvantages are summarized in Table A.5.

Table A.5 Advantages and Disadvantages of Meta-learning Approaches

Advantages	Disadvantages
Rapid Adaptation to New Users: Meta-learning frameworks, such as Model-Agnostic Meta-Learning (MAML) and MetaHIN, enable rapid adaptation to new users or items with sparse interaction histories, effectively addressing the cold-start problem by learning from few examples (Bharadhwaj, 2019).	**Complexity in Implementation:** The implementation of meta-learning approaches, especially those leveraging HINs for semantic context, can be complex due to the need for designing co-adaptation meta-learners and handling multifaceted semantic information (Cunha et al., 2018).
Improved Recommendation Performance: Systems can outperform state-of-the-art RS in cold-start scenarios, providing effective recommendations for both new (cold-start) and existing (non cold-start) users therefore reducing sparsity and thus performance (Cunha et al., 2018; Y. Lu et al., 2020).	**Risk of Overfitting:** Could lead to models that perform well on the training tasks but poorly generalize to truly new users or items (Cunha et al., 2018).
Utilization of Semantic Contexts: Meta-learning approaches like MetaHIN incorporate multifaceted semantic contexts from Heterogeneous Information Networks (HINs), enabling the learning of semantic priors for new tasks and enhancing the recommendation quality by understanding the multifaceted nature of user preferences (Y. Lu et al., 2020).	**Computational Expense:** Optimization-based meta-learning methods can be computationally expensive due to the computation of second-order derivatives and the requirement to update model parameters for each user, making them impractical for large-scale systems (Buffelli et al., 2023).
General Effectiveness: These methods train models to be generally effective across a wide range of users, which can then be fine-tuned with minimal data, outperforming traditional RS in cold-start scenarios and maintaining high accuracy for general recommendations (Bharadhwaj, 2019).	**Scalability and Latency Issues:** Current meta-learning approaches face challenges in real-world systems due to scalability and latency issues, with the need to save and retrieve adapted parameters for each user leading to additional computational and latency costs (Buffelli et al., 2023).
Performance Improvement: Meta-learning has shown practical advantages in improving performance over fine-tuning methods, with specific frameworks like MetaHIN achieving significant improvements across all metrics and datasets by effectively learning prior knowledge and leveraging semantic contexts (Y. Lu et al., 2020).	**Alternative Methods:** Research indicates that simpler models, when properly tuned, can perform comparably to or even outperform complex meta-learning methods for the cold-start problem. This suggests that the benefits of meta-learning may not always justify its computational complexity and resource requirements (Buffelli et al., 2023).

In summary, while meta-learning offers a promising solution to the user cold start problem in RS through its ability to rapidly adapt to new users and improve overall performance, its practical application is hindered by computational, scalability, and latency challenges. Alternative methods, including simpler models and representation learning techniques, present viable solutions that can achieve comparable performance with potentially lower complexity and resource demands.

Deep Learning (DL)

Deep learning leverages its capacity to extract complex patterns and learn high-dimensional data representations from minimal information. One approach involves using deep neural networks to recommend popular items to new users and personalize these recommendations based on available contextual information, such as the time of the day or the device used for accessing the service (Sethi & Mehrotra, 2021). This method relies on the network's ability to infer user preferences from sparse initial data, gradually improving as more user interactions are collected. Another innovative technique is the development of ColdGAN, a Generative Adversarial Network (GAN)-based model, which trains on user ratings to generate plausible behavior data for new users. This method does not require side information, making it particularly useful for pure cold start scenarios. The generative network learns to mimic the rating distributions of existing users, while the discriminative network works to distinguish between real and generated ratings, thereby enabling the system to make accurate recommendations to new users (C. C. Chen et al., 2023). Furthermore, deep learning's application in RS is relatively new, with significant research momentum gained around 2013. This nascent field continues to evolve, as evidenced by ongoing studies and the development of novel deep learning-based recommendation models. These models not only aim to enhance recommendation accuracy but also seek to provide explainable recommendations, thereby making the systems more transparent and trustworthy to users (Sethi & Mehrotra, 2021). The exploration of deep learning techniques for multi-modality, cross-domain recommendations further underscores the breadth of current research efforts, highlighting the early but promising stage of deep learning in RS (Sethi & Mehrotra, 2021). Table A.6 contains a summary of the different mentioned deeplearning methods together with the advantages and disadvantages summarized.

Table A.6 Advantages and Disadvantages of Deep Learning methods

Advantages	Disadvantages
Improved Recommendation Accuracy: DL models, by leveraging user-item interactions and side information, can significantly enhance the accuracy of recommendations for new users who have no prior interactions with the system (Briand et al., 2021; Obadić et al., 2017).	**Increased Computational Complexity:** The computational complexity of DL models is a significant disadvantage, requiring substantial computational resources for training and inference (Briand et al., 2021; Obadić et al., 2017).
Scalability: These models are highlighted for their scalability, capable of processing various types of interactions and efficiently handling diverse user-item interactions, which is crucial for large-scale RSs (Briand et al., 2021).	**Need for Large Amounts of Data:** Effective training of DL models often requires large datasets, which may not always be available or feasible to collect, especially for new items or users (Obadić et al., 2017).
Ability to Process Side Information: DL models can utilize textual descriptions and demographic information, improving recommendations by incorporating a broader range of usage signals (Briand et al., 2021; Obadić et al., 2017).	**Potential Bias in Predictions:** There is a risk of bias in predictions due to assumptions made about missing data, which can affect the performance and fairness of the recommendations (Briand et al., 2021).
Discovering User Preferences without Side Information: Techniques like Generative Adversarial Networks (GANs) can generate plausible user behavior data, enabling the system to suggest items to new users without relying on side information (C. C. Chen et al., 2023).	

In summary, deep learning offers a robust framework for tackling the user cold start problem in RS by utilizing complex pattern recognition and data representation learning. Its relatively recent application to this domain signifies a vibrant area of research, with ongoing developments aimed at improving recommendation quality and system explainability.

Matrix Factorization (MF)

MF (MF) in RS is a technique used to predict user preferences over items by decomposing a user-item interaction matrix into two lower-dimensional matrices, capturing latent factors associated with users and items. This approach is fundamental in collaborative filtering, a method that predicts a user's interest based on past interactions without requiring explicit profiles. The decomposition reveals latent factors that represent underlying characteristics of items and users, which are not directly observable but can be inferred from interaction data (Koren et al., 2009). MF in RS is not a model-based approach where latent factors are learned from the data. Instead, it relies solely on the interaction data between users and items. This distinction is crucial because it emphasizes the method's dependency on the quantity and quality of the interaction data to uncover meaningful patterns (Koren et al., 2009). Several evolutions of MF have been developed to enhance recommendation accuracy and address

specific challenges in RSs:

Funk MF Proposed by Simon Funk, this iterative method for low-rank Singular Value Decomposition (SVD) focuses on minimizing the error between predicted and actual ratings, primarily using gradient descent for optimization. It's one of the earliest and most straightforward MF techniques tailored for RSs (Pu & Faltings, 2013).

SVD++ An extension of SVD that incorporates implicit feedback (e.g., user views or clicks) in addition to explicit ratings. This model improves prediction accuracy by considering both the explicit and implicit interactions between users and items (Koren et al., 2009).

Asymmetric SVD Focuses on asymmetrically treating the user-item interactions, often by differentiating between the roles of users and items in the factorization process. This approach can capture more nuanced patterns in the data (Koren et al., 2009).

Group SVD A variant that considers group-level interactions, allowing for recommendations to be made for groups of users based on shared preferences or behaviors. This method is particularly useful in scenarios where collaborative consumption or group decision-making is involved (Koren et al., 2009).

Hybrid MF Combines MF with additional information, such as user or item attributes (explicit features), to address limitations like the cold start problem. Hybrid MF methods can make recommendations for new users or items by leveraging this auxiliary information (Zhao et al., 2016).

Recent development trends in MF that are relevant to addressing the cold start problem in RS include:

Attribute-to-Feature Mappings: A method that maps user and item attributes to latent features in a MF model, enhancing the model's ability to provide accurate predictions for new users and items by incorporating additional information such as gender, age, genres, and product categories. This approach helps retain the advantages of MF models, such as speed and predictive accuracy, while mitigating the cold-start problem (Gantner et al., 2010).

Context-Aware Semi-Supervised Co-Training (CSEL): Named CSEL, this method utilizes a factorization model to capture fine-grained user-item context and proposes a semi-supervised ensemble learning algorithm. It constructs weak prediction models using examples with different contexts and employs co-training to allow models to learn from each other, significantly improving recommendation accuracy (Zhang et al., 2014).

Regression-Based Latent Factor Model (RLFM): Improves prediction accuracy for both old and new users by incorporating features and past interactions, addressing the cold-start problem through predictions for new user-item dyads via features. It regularizes latent factors to avoid overfitting and achieve a bias-variance trade-off (Agarwal & Chen, 2009).

Hybrid MF (HMF): Incorporates explicit and implicit attributes in users or items factors matrices instead of relying solely on latent factors. This method retains correlations among users and items, utilizes more information, and facilitates recommending new users and items, directly addressing the cold start problem (Zhao et al., 2016).

Kernelized Probabilistic MF (KPMF): A matrix completion algorithm that enhances performance by incorporating external side information using Gaussian Process (GP) priors. Unlike standard Probabilistic Matrix Factorization, KPMF captures nonlinear covariance structures across rows and columns, making it particularly effective in handling sparse data and leveraging side information to address the cold start problem (Zhou et al., 2012).

Unified Models for Enhanced Recommendation Performance: Development of models like URM, UOCCF, and SPR_SVD++ that combine various MF techniques with list-wise learning-to-rank, item ranking and rating optimization, and user-trust and item ratings influence optimization. These models aim to address the new user cold start problem by enhancing recommendation performance through innovative combinations of model approaches (Feng et al., 2021).

These trends demonstrate a shift towards more sophisticated and hybrid approaches in MF techniques, leveraging additional user and item attributes, context-aware models, and semi-supervised learning to effectively tackle the cold start problem in RSs.

All advantages and disadvantages are mentioned in Table A.7.

Table A.7 Advantages and Disadvantages of MF Techniques

Advantages	Disadvantages
Enhanced Prediction Accuracy: MF techniques, including variations like Kernelized Probabilistic MF (KPMF) and Hybrid MF (HMF), have been shown to outperform classical nearest-neighbor methods and other collaborative filtering algorithms in terms of prediction accuracy (Koren et al., 2009; Zhao et al., 2016; Zhou et al., 2012).	**Computational Resources:** Implementing MF techniques, especially more complex models like KPMF or those incorporating neural networks, may require significant computational resources and expertise (Koren et al., 2009).
Handling Sparse Data: MF is particularly advantageous for dealing with sparse datasets, a common issue in RSs, by effectively inferring user preferences from limited feedback (Koren et al., 2009).	**Overfitting and Regularization:** The risk of overfitting is present in MF models, necessitating careful regularization to avoid it. This involves tuning parameters, which can be a complex process (Koren et al., 2009).
Incorporation of Side Information: Certain MF models, such as KPMF and CPMF-NN, can incorporate side information (e.g., user or item attributes), which enhances the model's prediction capability and allows for the handling of missing data (Cai & Chen, 2018; Zhou et al., 2012).	**Cold Start Problem:** While some MF methods address the cold start problem by incorporating attributes, traditional MF methods may struggle with providing recommendations for new users or items that have little to no historical interaction data (Zhao et al., 2016).
Efficiency and Scalability: MF models are among the fastest state-of-the-art methods for recommendations, offering quick computation times for predictions after the initial factorization (Gantner et al., 2010).	**Interpretability:** MF methods, particularly those that do not incorporate explicit attributes, can lack interpretability. This means that the reasons behind certain recommendations or predictions may not be clear to end-users (Zhao et al., 2016).

B Appendix B. Results table MRR and HIT

MRR@	VSTAN (Baseline)	History 5 + VSTAN	History 10 + VSTAN	History 15 + VSTAN	History 100 + VSTAN
0	0.19234	0.00015	0.00006	0.00007	0.00053
1	0.21310	0.00015	0.00015	0.00013	0.00010
2	0.18494	0.00005	0.00032	0.00008	0.00012
3	0.17348	0.00033	0.00021	0.00010	0.00009
4	0.17470	0.00011	0.00011	0.00041	0.00057
5	0.16559	0.17406	0.00010	0.00023	0.00007
6	0.16082	0.17865	0.00013	0.00039	0.00000
7	0.15159	0.16910	0.00034	0.00000	0.00000
8	0.15983	0.17266	0.00000	0.00000	0.00005
9	0.14795	0.18952	0.00008	0.00041	0.00031
10	0.16484	0.17008	0.17008	0.00000	0.00000
11	0.15703	0.18517	0.18517	0.00298	0.00000
12	0.15757	0.15790	0.15790	0.00000	0.00000
13	0.15267	0.19459	0.19459	0.00000	0.00000
14	0.15322	0.15506	0.15506	0.00535	0.00000
15	0.15359	0.20676	0.20676	0.20676	0.00000
16	0.13087	0.16977	0.16977	0.16977	0.00000
17	0.14231	0.14501	0.14501	0.14501	0.00201
18	0.15313	0.11264	0.11264	0.11264	0.00000
19	0.10542	0.16064	0.16064	0.16064	0.00000
20	0.13040	0.15688	0.15688	0.15688	0.00000
21	0.14181	0.14378	0.14378	0.14378	0.00000
22	0.11122	0.16178	0.16178	0.16178	0.00000
23	0.05467	0.09775	0.09775	0.09775	0.00000
24	0.18074	0.14363	0.14363	0.14363	0.00000
25	0.12518	0.15619	0.15619	0.15619	0.00000
26	0.12446	0.13542	0.13542	0.13542	0.00000
27	0.10142	0.40895	0.40895	0.40895	0.00000
28	0.14042	0.04819	0.04819	0.04819	0.00000
29	0.07324	0.04333	0.04333	0.04333	0.00000
30	0.01736	0.06667	0.06667	0.06667	0.00000
31	0.07560	0.03651	0.03651	0.03651	0.00000
32	0.02755	0.51250	0.51250	0.51250	0.00000
33	0.36667	0.52778	0.52778	0.52778	0.16667
34	0.26562	0.36364	0.36364	0.36364	0.00000
35	0.25000	0.75000	0.75000	0.75000	0.50000
36	0.50000	0.50000	0.50000	0.50000	0.00000
37	0.33333	0.00000	0.00000	0.00000	0.00000
38	0.16667	0.00000	0.00000	0.00000	0.00000
39	0.05000	0.00000	0.00000	0.00000	0.00000
40	0.00000	0.00000	0.00000	0.00000	0.00000
41	0.00000	0.00000	0.00000	0.00000	0.00000

HIT@	VSTAN (Baseline)	History 5 + VSTAN	History 10 + VSTAN	History 15 + VSTAN	History 100 + VSTAN
0	0.48995	0.00099	0.00064	0.00064	0.00191
1	0.54008	0.00140	0.00140	0.00100	0.00080
2	0.53258	0.00057	0.00099	0.00099	0.00128
3	0.51622	0.00099	0.00118	0.00079	0.00099
4	0.52131	0.00134	0.00161	0.00161	0.00215
5	0.51964	0.52369	0.00075	0.00112	0.00112
6	0.52056	0.51539	0.00149	0.00248	0.00000
7	0.49898	0.51014	0.00068	0.00000	0.00000
8	0.51099	0.51902	0.00000	0.00000	0.00091
9	0.48596	0.51703	0.00122	0.00122	0.00365
10	0.50349	0.52517	0.52517	0.00000	0.00000
11	0.49923	0.50670	0.50670	0.00446	0.00000
12	0.49095	0.51453	0.51453	0.00000	0.00000
13	0.49614	0.53184	0.53184	0.00000	0.00000
14	0.50962	0.48485	0.48485	0.01010	0.00000
15	0.50207	0.56944	0.56944	0.56944	0.00000
16	0.43103	0.47664	0.47664	0.47664	0.00000
17	0.40580	0.45783	0.45783	0.45783	0.01205
18	0.48571	0.50725	0.50725	0.50725	0.00000
19	0.46250	0.52830	0.52830	0.52830	0.00000
20	0.44262	0.36111	0.36111	0.36111	0.00000
21	0.30435	0.48387	0.48387	0.48387	0.00000
22	0.41026	0.50000	0.50000	0.50000	0.00000
23	0.34375	0.33333	0.33333	0.33333	0.00000
24	0.38462	0.35294	0.35294	0.35294	0.00000
25	0.31818	0.40000	0.40000	0.40000	0.00000
26	0.47368	0.50000	0.50000	0.50000	0.00000
27	0.50000	0.66667	0.66667	0.66667	0.00000
28	0.50000	0.42857	0.42857	0.42857	0.00000
29	0.54545	0.40000	0.40000	0.40000	0.00000
30	0.20000	0.20000	0.20000	0.20000	0.00000
31	0.37500	0.40000	0.40000	0.40000	0.00000
32	0.28571	0.75000	0.75000	0.75000	0.00000
33	0.60000	1.00000	1.00000	1.00000	0.33333
34	0.50000	0.66667	0.66667	0.66667	0.00000
35	0.25000	1.00000	1.00000	1.00000	0.50000
36	0.66667	0.50000	0.50000	0.50000	0.00000
37	0.33333	0.00000	0.00000	0.00000	0.00000
38	0.50000	0.00000	0.00000	0.00000	0.00000
39	0.50000	0.00000	0.00000	0.00000	0.00000
40	0.00000	0.00000	0.00000	0.00000	0.00000
41	0.00000	0.00000	0.00000	0.00000	0.00000