Recommendations over Knowledge Graph Entities in Cold-Start Interviews

Anders Brams
abrams14@student.aau.dk
Department of Computer Science, Aalborg University
Aalborg, Denmark

Anders Jakobsen
alja15@student.aau.dk
Department of Computer Science, Aalborg University
Aalborg, Denmark

Theis Jendal
tjenda15@student.aau.dk
Department of Computer Science, Aalborg University
Aalborg, Denmark

SUMMARY

This work investigates the informational value of asking towards descriptive entities in cold-start user interviews. Recommender systems are used in a large variety of services and platforms and are responsible for providing users with recommendations, e.g., what products to buy in e-commerce services or people to connect with on social media platforms. For recommendations to be effective, some level of personalisation is typically necessary. This requires knowledge about the user's preferences, for instance in the form of previously observed interactions such as ratings or purchases. A cold-start user, however, is a user that the system has no knowledge of, and thus no such previous interactions are observed. To elicit the user's preferences, a typical approach is to conduct a brief interview, asking towards the most informative entities in the system. The key challenges in making cold-start recommendations are two-fold: first is the problem of determining what questions to ask in order to most effectively elicit a user's preferences without losing their interest. Second is the problem of utilising the user's answers effectively to generate high-quality recommendations.

Approaches to cold-start user interviews have received much attention in prior research, and have considered approaches ranging between fixed sets of questions as well as interviews that adapt to the user dynamically depending on their answers throughout the interview. Yet, we notice an apparent gap in this research in that the questions posed have been limited to the *recommendable entities* in the system (e.g., movies) rather than *descriptive entities* (e.g., genres and directors of movies). This contrasts the general consensus in the literature that users express their preferences more naturally through such descriptive entities. From an interviewing standpoint, it makes sense intuitively to ask broad questions such as "do you like horror movies?" rather than immediately asking towards entities the user may not be familiar with, for instance "do you like Hush?".

This observed gap in prior research can largely be attributed to the lack of available and appropriate datasets allowing for the evaluation of systems asking towards such descriptive entities. In our prior work we built MindReader, a recommendation and data collection platform designed to address this lack of datasets specifically. In total, the MindReader users built a dataset of $\sim 102,000$ ratings from $1,174$ real users on recommendable- and descriptive entities within the movie domain. We further find that jointly graph-based and collaborative recommendation models can significantly outperform the more standard collaborative filtering models (e.g., matrix factorisation) upon which the majority of the proposed interview approaches are based on.

In addition to our primary objective, we describe how we have extended MindReader to remain performant under heavy traffic, improved the recommendation quality, and increased the diversity of entities for which data is collected, extending the MindReader dataset to contain $174,872$ ratings from $1,736$ real users in the process.

We conduct a comprehensive study of how different interviewing strategies and recommender paradigms can affect each other when conducting user cold-start interviews and generating recommendations in what, to the best of our knowledge, is the first such study. Our interviewing strategies range from both fixed and adaptive interviewing approaches to question selection policies powered by deep reinforcement learning. We combine such interviewing strategies with recommender models from different paradigms of recommendation, and present a new model of recommendation, PPR-LINEAR, based on a linear combination of personalised PageRanks over different graphs learned through descending the gradient of a ranking-based loss function. We compare all our combined models against a set of state-of-the-art baselines in interviews restricted to descriptive- and recommendable entities separately to assert the informational value afforded by descriptive entities.

We find that, with few exceptions, all models are able to improve the quality of recommendations made when conducting interviews with descriptive rather than recommendable entities. Furthermore, we find that when interviewers are allowed to ask towards descriptive entities, the interview can be shortened by ~ 4 questions while still obtaining the same recommendation quality as obtained with interviews on recommendable entities. We demonstrate how fixed- and adaptive interviewing strategies generally incur a trade-off between accuracy and diversity, both of which are important qualities of recommender systems. We show that our PPR-LINEAR recommender can make effective use of the descriptive entity interview answers, outperforming every baseline considered in terms of ranking quality, diversity, and serendipity of recommendations made. Finally, we find that graph-based models of recommendation, especially when navigating a knowledge graph, can significantly increase the diversity and serendipity of recommendations while maintaining ranking quality. This greatly motivates future research in utilising the expressive power of knowledge graphs in making recommendations, and we provide concrete suggestions for possible directions of such research.

This page intentionally left blank.

ABSTRACT

A key challenge in recommender systems is how to provide recommendations for cold-start users about which the system has no prior knowledge. A common approach to this problem is to conduct a brief interview with the user to elicit their preferences on a number of informative entities. While most proposed approaches have focused on eliciting item-specific preferences from users, users may be able to better opine on broader and more descriptive properties of items, denoted as descriptive entities. While this focus on items, denoted recommendable entities, can be attributed largely to the lack of available and appropriate datasets, the recently published MindReader dataset alleviates this issue. In this work, we perform a comprehensive study of interviewing strategies and models of recommendation including state-of-the-art methods, and evaluate the effectiveness of allowing interviewing systems to ask towards descriptive entities in the MindReader dataset, which we further extend to $1,736$ users and $174,872$ ratings. In order to construct optimal interviews, we propose a novel, adaptive interview learning approach, as well as approaches based on deep reinforcement learning. For making recommendations from user answers, we further propose a linear combination of Personalised PageRank which learns the importance of knowledge- and collaborative graphs through a pairwise ranking loss. Our findings show that almost all models can improve performance with broader questions, allowing the interview to be shortened by ~ 4 questions when asking towards descriptive rather than recommendable entities. Our findings show that especially knowledge-aware approaches can benefit greatly from descriptive entity preferences in cold-start interviews and outperform state-of-the-art methods in both recommendation quality and diversity.

1. INTRODUCTION

Recommender systems have been widely employed in IT systems in the industry, be they systems for entertainment, service provision, or social media, to name a few [1]. The overall purpose of recommender systems is to increase user loyalty, satisfaction, and retention within the given system [2]. While the specific approaches used in recommender systems can be grouped into those of Collaborative Filtering (CF), Content-Based Filtering (CBF), and hybrid approaches [3], the specific objective of any recommender system is to determine the match between a user and an item the user might be interested in as accurately as possible. In warm-start recommender systems, the system has access to explicit observations regarding users' preferences towards items in the domain [4]. Conversely, in cold-start recommender systems, the system must generate recommendations for a user of which the system is initially unaware [4, 5, 6, 7].

In order to overcome the inherent sparsity of information in the cold-start setting, auxiliary information such as item-specific attributes and metadata has been incorporated in learning, CF-based models [8, 9]. Another strategy is to conduct an interview with a cold-start user, eliciting the user's preferences towards specific items of interest in order to build an initial profile from which high-quality recommendations can be generated [5, 6, 7]. Existing models learn from Matrix Factorisation (MF)-based objectives yielding excellent capturing of the CF effect, but makes the inclusion of auxiliary data in the training process a non-trivial task. Additionally, MF-based models are effective in rating prediction tasks, but usually poorly performing in top-K recommendation tasks [10], although top-K recommendations have received the most attention in the industry [11].

In addition, such interviewing systems have primarily focused on questions regarding Recommendable Entities (REs) (e.g., movies) rather than broader, Descriptive Entities (DEs) in the domain (e.g., genres and actors), although DEs are deemed richer sources of information in preference elicitation problems [4, 6, 12, 13]. Some approaches work under the assumption that a user liking a movie tagged as comedy and action indicates that the user is likely interested in both of those genres [12]. However, we argue that this assumption can lead to a pitfall. For example, say a user likes "Batman: The Dark Knight" and "Man of Steel". While both movies are action movies, they also share a number of other DEs with other movies that the user might not like. For instance, the movie "Deadpool" shares the action and superhero DEs with the aforementioned movies, but is also a comedy movie. We might assume that the user likes all superhero movies, though we may find that the user dislikes comedy movies, rendering this assumption fallacious. Furthermore, it may be the case that the user does not like the two aforementioned movies because of their relation to superhero movies, but rather because they like dark, dystopian dramas or simply because they like Christopher Nolan as a screenwriter. Knowing the user's explicit preferences towards these DEs can yield a more comprehensive profile of the user's preferences.

In [4], we collected a dataset of explicit user ratings on REs as well as DEs, and found that a user is significantly more likely to provide useful feedback (i.e., like or dislike) when asked about genres and subjects rather than specific movies. We further found that in the warm-start setting, models explicitly modelling the relations between descriptive and recommendable entities (e.g., a movie is related to its actors, genres, etc.) can outperform both MF and translational distance embedding-based models in the warm-start recommendation task.

In a cold-start interview, it is important to keep the interview as short and rich in information as possible as to keep the experience enjoyable for the end-user [5, 6, 7, 13]. Furthermore, in [14, 15, 16] they find that users are willing to engage in an initial survey if it increases the accuracy of recommendations. As mentioned, users are more likely to be able to opine on DEs. In observing that models can maintain or increase performance with RE ratings replaced by DE ratings [4], this makes DEs potential means of reducing the interview length. However, while previous interviewing cold-start models have not asked towards DEs in their experiments, this is a limitation pertaining to the at-the-time available datasets rather than the models themselves. Given access to observations on DE ratings, an MF-based interviewing model would be able to query for opinions on DEs, although the MF-based objectives presented in previous works [5, 6, 7] do not facilitate explicit modelling of entity inter-relations, which may serve as important information in selecting appropriate interview questions.

In this work, we assert whether different interviewing models for cold-start recommendation can conduct shorter and/or more effective interviews when allowed to ask questions towards both REs and DEs. We further investigate how explicitly modelling the relations between REs and DEs can be utilised in improving the quality of recommendations made. Specifically, we pose the following research questions:

1. How can different question selection strategies affect the quality of cold-start recommendations following an interview?
2. How can descriptive entity ratings affect the quality of recommendations in interview-based cold-start recommender systems?
3. How can descriptive entity ratings affect the required interview length in generating recommendations of sufficient quality?
4. How can the explicit modelling of a knowledge graph over the entities affect the quality of recommendations made?

The contributions of this work are four-fold. First, we show how the MindReader data collection platform can be improved and collect an extended version of the dataset, which we employ in the evaluation of this work. Second, we propose a novel decision tree-based interviewer which, unlike our considered baselines, is recommender- and metric agnostic. Third, we present a novel, hybrid recommender for generating high-quality recommendations by means of learned weights between rankings scores from different graphs. Finally, we perform a comprehensive study of cold-start interviews to show the relative effectiveness of DE-based interviews on the recommendation task across a variety of interviewers and recommenders, including State-Of-The-Art (SOTA) approaches.

2. COLD-START INTERVIEWS

We define the overall problem of cold-start recommendations as the problem of providing recommendations for new users. We first define $\mathcal{E}$ as the set of all entities in the domain. We further define $\mathcal{I} \subset \mathcal{E}$ as the set of REs, i.e., the entities we are interested in recommending to a user (e.g., movies). These are complemented by the set of DEs $\mathcal{E}' = \mathcal{E} \setminus \mathcal{I}$, i.e., the set of entities we are not interested in recommending to a user (e.g., genres and actors). These entities are interrelated. We model the relations between entities $\in \mathcal{E}$ by means of a Knowledge Graph (KG).

Definition 2.1 (Knowledge Graph)**.** A KG is defined as a triple $\langle \mathcal{E}, \mathcal{R}, \mathcal{L} \rangle$ describing a directed labelled multi-graph, where the nodes are represented with entities $\mathcal{E}$, the edges $\mathcal{R}$ capture the relationships among them, and the labels $\mathcal{L}$ are the names for the entities and relationships. Given a finite set of relations we can define the edges as $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{L}' \times \mathcal{E}$, where $\mathcal{L}' \subseteq \mathcal{L}$ captures the relationship types. Thus, we have a mapping to labels $\Phi : \mathcal{E} \cup \mathcal{R} \mapsto \mathcal{L}$. An example edge $\langle h, l, t \rangle \in \mathcal{R}$ is $\langle \text{Deadpool}, \text{Has Genre}, \text{Action} \rangle$.

Let $\mathcal{U}$ be the set of all users. All users have ratings for entities $\in \mathcal{E}$ as defined by R. Let $R : \mathcal{U} \times \mathcal{E} \to \mathcal{C}$ be the rating function where $\mathcal{C} = \{\text{Like}, \text{Dislike}, \text{Don't know}\}$ is the set of possible ratings from a user to an entity. With ratings, users also have a set of specific preferences. Users express their preferences as preferring some entities over others. We follow a similar definition of preference as proposed in [4]:

Definition 2.2 (Preference)**.** Given a user $u \in \mathcal{U}$, their preferences are given by the weak, non-strict ordering of entities $(\mathcal{E}, \leqslant_u)$ s.t. for any $e, d \in \mathcal{E}$, we have that $e \leqslant_u d$, that is u prefers d over e, if either

1. $R(u, d) = \text{Like} \land R(u, e) \in \{\text{Don't know}, \text{Dislike}\}$, or
2. $R(u, d) = \text{Don't know} \land R(u, e) = \text{Dislike}$

There are two main classes of cold-start problems for recommender systems, namely the new user and new item (i.e., RE) problems [17]. The former is the primary focus of this work and is the more difficult problem of the two, as metadata is often available for new REs.

Conducting interviews is just one approach to solving the new user cold-start problem [17, 18]. Other approaches include using external information, e.g., social media and demographic data, referred to as an implicit approach [17], where interviews are referred to as explicit approaches to the problem. A major difference between the two approaches is the user effort required. For implicit methods, little effort is usually required [19]. Interviews require the user to interact with the system which can discourage users from completing the interviews [17], though [14, 15, 16] suggest that this initial effort does not matter for users.. However, methods generally incur a trade-off in the quality of information, as the information acquired through an interview is highly relevant, where external data is typically varying in quality and availability.

In summary, implicit methods attempt to understand a new user by obtaining a large amount of information that can be manipulated and filtered for further use, while explicit methods aim to collect enough information to make recommendations without overwhelming the user. We will be focusing on explicit methods, specifically by means of user interviews.

In interview-based cold-start recommendation, we are interested in eliciting the preferences of the user through a number of interview questions. For each such question, the interviewing system is allowed to ask a user for their rating on a chosen entity, thereby observing part of their preferences. The system is limited in the allowed number of questions before recommendations must be made. The challenge then becomes to determine how to conduct the interview to best approximate the full preferences of the user.

Problem 2.1 (Cold-start interviews)**.** *Given a cold-start user u, the system is allowed to ask u for their ratings on at most m entities in $\mathcal{E}$. The interview process can be written as*

$$I(u) = q_1 \xrightarrow{R(u,e_1)} q_2 \xrightarrow{R(u,e_2)} \dots \xrightarrow{R(u,e_{m-1})} q_m \xrightarrow{R(u,e_m)} \mathcal{O}_u$$

s.t. the interview, when conducted with u, results in the set of elicited observations $I(u) \to \mathcal{O}_u = \{(u, e_i, R(u, e_i)) \mid i \in \{1, \dots, m\}\}$ where e_i is an entity selected by the recommender system at question q_i for all $i \in \{1, \dots, m\}$.

Determine the optimal m-length interview I such that, when provided with the results of the interview and the KG, the difference between u's preferences and those predicted by an arbitrary recommender system S is minimised, that is, determine

$$\arg\min_{I \in \mathfrak{I}} D(\widehat{\leqslant_u}, \leqslant_u)$$

where $\widehat{\leqslant_u}$ is given by $S(\mathcal{G}, I(u) \cup \mathcal{O}_w)$, $\mathfrak{I}$ is the space from which I is selected, D is a given ordering difference function, and $\mathcal{O}_w$ are observed interactions from warm-start users.

3. THEORETICAL OVERVIEW

Before proceeding, we make a clear distinction between interviewers, recommenders, and interviewing recommenders. In the remainder of this work, we refer to *interviewers* as models responsible only for selecting questions, and *recommenders* as models responsible only for making recommendations. As such, we define interviewers and recommenders as functionally independent components that can be combined arbitrarily. We will also discuss some models that optimise the interview and recommendations in tandem, which we refer to as *interviewing recommenders.*

In this section, we provide a theoretical overview of the state of knowledge within the area of cold-start interviews. Specifically, the goal of this section is not to provide highly specific details on different models, but instead to cover the core principles of different interviewing strategies, their potential weaknesses, and why some might be preferred over others. An overview of these interview strategies is provided in subsection 3.1. Furthermore, since interviewers cannot provide recommendations on their own, we briefly describe different recommender paradigms in subsection 3.2 with a focus on their ability to accommodate new users. In later sections of this work, we cover how each strategy can be implemented in detail, in addition to implementation details on recommenders.

Notation. Throughout this work, we use a set of notation conventions shown in Table 1.

3.1 Interviewing strategies

An interview, in its most basic form, is an iterative process of posing questions to an interviewee and receiving answers for the questions. As such, we are situated in a feedback-loop between the interviewer and the interviewee, with each answer providing additional feedback to the interviewer.

An optimal interview should be kept brief and rich in information [5, 6, 13]. Therefore, it is important for the interviewer to choose questions that ensure that the system will receive the most amount of useful information in as little time as possible. For example, if a user has stated that they do not like horror movies, asking for their opinion on a specific horror movie will likely not lead to additional useful feedback. Conversely, if the user enjoys horror movies, asking for their opinion on a specific horror movie could allow the system to hone in on the specific qualities of horror movies the user enjoys. It is therefore reasonable to expect an optimal question selection process to be predicated by the answers the interviewer has received so far.

3.1.1 Fixed-question interviews

Analogous to naive recommenders which always provide the same recommendations, an approach to interviewing is to ask all interviewees the same questions, referred to as popularity-based, fixed-question, and static interviews in the literature [14, 20, 21, 22, 23]. As seen in Figure 1, the question selection is not predicated on the user's answers, thus the questions are fixed for all users.

In the remainder of this subsection, we will cover different approaches for question selection in fixed-question interviews.

Popularity. A common approach to fixed-question interviews is to ask the user about the most popular entities, thereby reducing the risk of the user being unfamiliar with the entities asked about. As shown in our previous work [4],

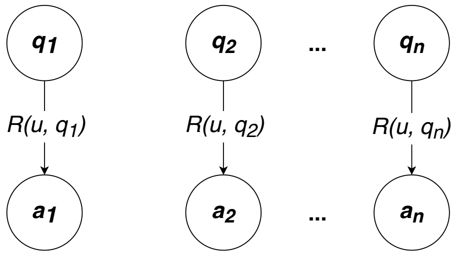

Figure 1: Representation of a fixed-question interview. A question q is not dependent on the answer a received previously.

both MindReader and MovieLens [24] follow long-tail distributions which fit well with a popularity-based approach. While popular entities can lead to higher user similarity for CF-based models, evidence on popular entities does not necessarily equate to high informativeness [14, 21, 23]. For example, if some entity has been liked by all users, there is little benefit in asking about it during an interview despite it being popular.

Contentiousness. To avoid the issue of asking mainly about well-liked entities, which can be the result of conducting a popularity-based fixed-question interview, it is favourable to ask about more contended entities [23]. The contentiousness of an entity can be quantified with the variance or entropy of its ratings, e.g., the entities can be selected in decreasing order of variance as to ask about more controversial entities first. However, ordering on this criterion alone is unwise, since controversial entities tend to receive fewer ratings, while popular entities tend to be well-liked [23, 25]. More specifically, the findings of [23] suggest that any selection criterion should be used in combination with popularity.

Informativeness. More generally, we wish to ask about entities with the highest *informativeness* at each step during an interview. Popularity and contentiousness are both heuristics for such informativeness, but as previously mentioned they both have shortcomings. To find entities of high informativeness, the authors of [23] propose treating the question selection problem as the optimisation problem

$$\mathcal{Q}_{\mathcal{E}} = \underset{\mathcal{Q}_{\mathcal{E}} \subset \mathcal{E}, |\mathcal{Q}_{\mathcal{E}}|=m}{\arg\min} \quad F(S(\mathcal{Q}_{\mathcal{E}})) \tag{1}$$

where $\mathcal{Q}_{\mathcal{E}}$ is the set of questions to ask about, m is the number of questions to ask, F is a performance scoring function, and S is a recommender function. In their work, they evaluate on a MF-based recommender to minimise Root Mean Square Error (RMSE), but the approach is applicable to any metric that can be computed efficiently. To select entities, they iteratively extend $\mathcal{Q}_{\mathcal{E}}$ with the next $e \in \mathcal{E}$ that would increase performance the most, i.e.:

$$e = \underset{e \in \mathcal{E} \setminus \mathcal{Q}_{\mathcal{E}}}{\arg\min} F(S(\mathcal{Q}_{\mathcal{E}} \cup e)) \tag{2}$$

While research has moved in the direction of adaptive methods [5, 26, 27], it might be worthwhile to revisit a fixed-question greedy approach for several reasons. First, works on adaptive methods have focused on the prediction task, and as a result, the existing methods are mainly designed

Concept	**Notation**	**Description**
Sets	$\mathcal{A}, \mathcal{B}, \mathcal{C}$	Sets denoted with calligraphic font
Functions	A, B, C	Functions denoted with italic font
Matrices	$\mathbf{A}, \mathbf{B}, \mathbf{C}$	Matrices denoted with uppercase letters, bold font
Vectors	$\mathbf{a}, \mathbf{b}, \mathbf{c}$	Vectors denoted with lowercase letters, bold font
Variables and elements	a, b, c	Variables and set/vector elements denoted with lowercase letters, italic font

Table 1: Notation conventions used throughout this work unless explicitly stated otherwise.

to minimise rating prediction RMSE which does not necessarily equate to high-quality rankings under the top-K recommendation problem [10]. Second, the greedy approach is untested on newer recommender approaches, and especially knowledge-aware recommenders such as Personalised PageRank (PPR) over a KG could be useful given the sparsity of ratings in a cold-start setting.

Meta-learning. A recent fixed-question approach proposed by [21] suggests another measure of question informativeness leveraged by their model, Meta-Learned User Preference Estimator (MeLU), which uses meta-learning to adapt to new users using as little data as possible and is able to identify the most informative entities from learned features rather than empirical measures as proposed by [23]. The MeLU model uses user-entity specific gradients resulting from a differentiable loss to select the entities with the largest gradients for all users, as this is indicative of large gains in information [21]. Following [23], the gradients are scaled by popularity.

3.1.2 Adaptive interviews

Fixed-question interviews provide a solid baseline, yet suffer some rather obvious limitations. A fixed-question interview can be modelled as a sequence of up to $m = |\mathcal{Q}_\mathcal{E}|$ questions, i.e., $I(u) = \mathcal{Q}^1_\mathcal{E} \rightarrow \cdots \rightarrow \mathcal{Q}^n_\mathcal{E}$ where $\mathcal{Q}^i_\mathcal{E}$ corresponds to the i^{th} entity ordered by some criterion, e.g., popularity, contentiousness, or informativeness. As a result, regardless of how two separate users answer question $\mathcal{Q}^i_\mathcal{E}$, they will always be asked about $\mathcal{Q}^{i+1}_\mathcal{E}$ as the next question. As argued and exemplified previously, it is reasonable to expect an optimal interview process to be adaptive as to avoid redundant questions and thus save time.

In adaptive interviews, the question selection is predicated on the answers received throughout the interview, as illustrated in Figure 2. We now explore several strategies for implementing adaptive interviewers.

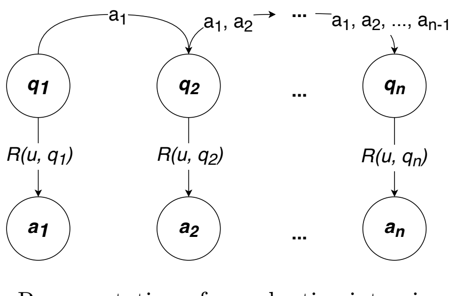

Figure 2: Representation of an adaptive interview. The selection of question q_n is predicated on the answers received on the prior questions $\{a_1, a_2, \ldots a_{n-1}\}$.

Greedy adaptation. In order to generate adaptive interviews, [5, 27] propose to model the interview as decision tree constructed by the models Functional Matrix Factorisation (fMF) and Local Representative-based Matrix Factorisation (LRMF), respectively. In the decision tree, each node represents an entity to ask about, and the interview branches to a sub-tree of the interview depending on the user's answer to a node entity. In this way, the interview is able to adapt to the user's answers as the interview progresses.

While fMF and LRMF are both capable of constructing the interview, they are also capable of making recommendations. This makes both models *interviewing recommenders* where both the interview and recommendation objectives are optimised in tandem. As each node of the tree, questions are selected greedily, choosing the question that best optimises the given objective at that node.

Exploratory adaptation. While greedy adaptation can be an effective strategy, we may be able to generate a more effective interview if we allow the interviewer to explore more freely in the questions selected [6]. Additionally, both fMF and LRMF learn the interview, and generate recommendations, by minimising the difference between predicted and observed ratings.

While this can be an effective strategy for some ranking tasks, we may be interested in learning an approximation of a non-differentiable function, e.g., those defined by more complex serendipity- or diversity-based metrics [28], which is not trivially possible through an MF objective.

Instead, we can leverage reinforcement learning, specifically deep Q-learning [29, 30, 31], to learn how to conduct interviews such that we reinforce any positive outcome including such non-differentiable metrics. The works of [29, 30, 31] propose a set of reinforcement learning strategies for a variety of problematic settings. While general purpose, we can utilise these strategies to construct a learning interviewer that reinforces any positive outcome of our choosing. The models learn by partly exploiting the action space, choosing the question with the highest predicted reward, or exploring it randomly, enabling the discovery of optimal interview sequences.

3.2 Recommenders

In this work, an important distinction between recommenders is their ability to accommodate new users. The authors of [23] recognise this distinction to some extent and refer to the accommodating recommenders as being incremental, in the sense that ratings can be incrementally added without a training phase. Yet, to the best of our knowledge, no existing work has studied the effects of diversifying recommenders in cold-start interviews. Given the high variance in performance when DE ratings are provided as observed in [4], we wish to diversify not only interviewers but the underlying recommenders as well.

While a deeper theoretical analysis of different paradigms has already been covered in [4], we briefly brush up the different recommender paradigms that we consider, and the problems these paradigms may face in the cold-start setting. Furthermore, we consider how the different paradigms have been applied in existing cold-start interview approaches.

3.2.1 Embedding-based

We consider embedding-based recommenders as those operating by means of learned, latent user- and entity representations that encode the high-level features necessary for predicting user preference, e.g., MF [32]. While effective especially in the warm-start setting, MF is a collaborative filtering model. For a cold-start user, the collaborative filtering effect is difficult to capture for obvious reasons, hence why MF-based models are not considered incremental. Yet, existing works on cold-start interviews have mainly focused on embedding-based recommenders as the underlying recommender [5, 21, 22, 27].

3.2.2 Neighbourhood-based

We consider neighbourhood-based recommenders as those operating on a notion of a user-entity neighbourhood. Typical implementations of such recommenders are user- and item k-Nearest Neighbour (k-NN) models that determine the k nearest neighbours to a user or RE, respectively [33]. Unlike embedding-based recommenders, k-NN-based recommenders do not operate with learned embeddings of users and entities, and are incremental models well fit for cold-start recommendations since a user's embedding is represented as an explicit ratings vector that can be extended freely.

3.2.3 Graph-based

We consider graph-based recommenders as those operating by explicitly modelling a graph of entities and/or users, e.g., PPR, for ranking entities [34, 35]. Similarly to k-NN-based models, PPR has no notion of learned user or entity embeddings. As in k-NN, a user is represented explicitly by their ratings, though in PPR the ratings are used as source nodes for computing the PPR of entities across the graph. Such source nodes, as representative of a user, can be added freely, and thus PPR is another incremental model well fit for cold-start recommendations. To the best of our knowledge, despite the ability to easily integrate auxiliary knowledge, no existing works on cold-start interviews have considered graph-based models as the underlying recommender.

4. MODEL IMPLEMENTATIONS

In order to determine the best overall strategy for conducting cold-start user interviews with the goal of generating high-quality top-K recommendations, we now cover how each interviewing strategy can be implemented in models ranging from simple, naive question selection algorithms to more complex deep learning models. Furthermore, we cover the implementation of specific recommenders that can be arbitrarily coupled with interviewers, and how these are adapted to the cold-start setting. In Appendix B, we cover the implementation of additional models for which testing remained preliminary due to time constraints.

4.1 Fixed-question interviews

As described in subsubsection 3.1.1, one way to conduct interviews is to select a fixed set of questions that are posed to the user in no particular order. We explore the following implementations of this strategy.

4.1.1 Naive interviewers

As the simplest form of interviewer, we can create a *naive interviewer* that simply selects a fixed set of m questions from entities that are the most popular and/or contentious among users. This form of interview is naive as the question selection process does not adapt to a user throughout the interview and does not depend on an underlying recommender. In this work, we consider a popularity-based naive interviewer, where m questions are selected from the m most popular entities. Due to space constraints we do not consider contentiousness and its combination with popularity in this work, though a preliminary study indicated that the performance is similar to or worse than a purely popularity-based approach on our dataset.

4.1.2 Greedy interviewers

While asking towards popular and contentious entities is intuitively a reasonable approach, there is no guarantee that these metrics allow an arbitrary recommender to correctly infer user preferences. Instead, questions can be selected from the performance afforded by a recommender when given users' answers to the questions.

A *greedy* interviewer exhaustively tests all candidate questions according to the information they afford. Specifically, by providing the user answers to an underlying recommender, the interviewer records the ranking performance of the recommender given the information and greedily chooses the top-m entities that afford the best performance.

While more computationally complex, this strategy can adapt to the recommender employed by the recommendation system, a feature not present in naive interviewers. Additionally, a greedy interviewer ensures that the questions chosen optimises the desired performance metric, regardless of the recommender. Much like how questions are selected in fMF and LRMF when building the tree, the informativeness of a candidate question is evaluated in the context of the previously selected candidates, though a fixed-question greedy interviewer is not adaptive.

While optimally all combinations of questions should be considered, this becomes intractable even for short interviews if the candidate question set $\mathcal{E}'$ is large, and thus the greedy selection strategy is warranted. At every step, the entity $e \in \mathcal{E}'$ selected for questioning by a greedy interviewer is the one maximising informativeness w.r.t. an arbitrary recommender system (see Equation 1) in the context of the previous questions selected in the same manner.

4.1.3 Meta-Learned User Preference Estimator

As briefly described in subsubsection 3.1.1, MeLU [21] proposes to use meta-learning to determine the informativeness of candidate questions. The model takes a user-entity pair as input represented by their content, as shown in Figure 3. In the model, an entity is represented by its related content, e.g., genres and actors, while a user is described using demographic data, e.g., age, geographical location, and gender. Since the MindReader dataset does not contain user metadata, we replace the user embedding with the average embedding of the entities rated by the user. The entity and user embeddings are concatenated and passed to a Fully Connected (FC) Neural Network (NN), which uses

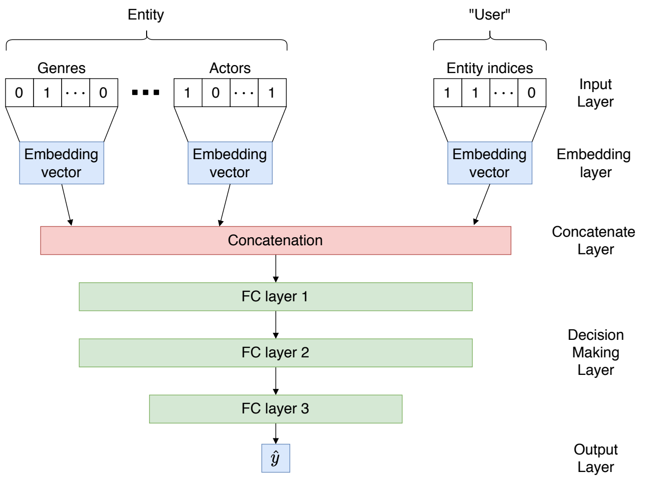

Figure 3: The MeLUN network where the user embedding is replaced with rated entities.

non-linear activation functions. The training data is partitioned into a support and query set. The support set is used for local, temporary updates of the decision-making layers, while the query set is used for global, permanent updates of the model[1]. For a user-entity pair, (u, e, o) in the support set, we only insert the index of e into our entity vectors, while for the query set we input all entities seen in our support set. The local updates match each user s.t. the model becomes better at predicting entity ratings in the query set. Since MindReader has fewer ratings per user as opposed the dataset used in [21], we determine the number of ratings n assigned to the support set as:

$$n = \begin{cases} 10 & \text{if } r >= 13 \\ \lfloor r/2 \rfloor & \end{cases} \tag{3}$$

where r is the number of ratings for a user. Our version of MeLU is referred to as MeLU Non-User (MeLUN) to highlight the absence of user content. In [21], they further use the average l_2 norm of a user's gradient on the support set. This gradient is normalised and added to the normalised popularity of each entity, resulting in a score that represents the information gain if a user's preference towards the entity was learned and the probability that users are familiar with the entity. The questions asked by MeLUN are optimised for the model itself, and are not applicable to any arbitrary recommender. As such, MeLUN is another instance of an interviewing recommender.

4.2 Adaptive interviews

As previously argued, it might be beneficial to have selection strategy predicated by the answers received throughout the interview in a more dynamic fashion. A popular representation of adaptive interviews is by means of decision trees, and we will discuss different implementations of such interviews using greedy question selection, including the approaches proposed by [5] and LRMF [27]. We further discuss how question exploration can be leveraged in order to improve over greedy selection.

[1]We refer the reader to [21] for an in-depth description of these updates.

4.2.1 Functional Matrix Factorisation

The fMF model is based on standard matrix factorisation, modelling users and entities as latent feature vectors. These vectors are constructed as to minimising the RMSE between predicted and actual ratings from users to entities, where ratings are predicted as the inner products between the embedding vectors of the users and entities. The fMF model constructs a decision tree where nodes represent questions and edges represent answers. When answering a question, a user is routed to the appropriate sub-tree in order to continue the interview.

At each level of the decision tree, we select the question that allows the model to optimally split the users into groups according to their answers. Specifically, we select the question p as

$$\begin{aligned} p \leftarrow \arg\min_{p} & \sum_{u \in R_L(p)} \sum_{(u,e,r) \in \mathcal{O}} (r - \mathbf{u}_L^\top \mathbf{e})^2 \\ & + \sum_{u \in R_D(p)} \sum_{(u,e,r) \in \mathcal{O}} (r - \mathbf{u}_D^\top \mathbf{e})^2 \\ & + \sum_{u \in R_U(p)} \sum_{(u,e,r) \in \mathcal{O}} (r - \mathbf{u}_U^\top \mathbf{e})^2 \end{aligned} \tag{4}$$

where $u \in R_L(p)$, $R_D(p)$, and $R_U(p)$ is a user who has answered that they like, dislike, or do not know p, respectively, $(u, e, r) \in \mathcal{O}_w$ is an observation from u on e with rating r, and $\mathbf{u}_{\{L,D,U\}}, \mathbf{e} \in \mathbb{R}^k$ are the user and entity embeddings, respectively. The user embedding $\mathbf{u}_L$ is the embedding that optimally represents all users in $R_L(p)$, that is

$$\mathbf{u}_L \leftarrow \arg\min_{\mathbf{u}} \sum_{u \in R_L(p)} \sum_{(u,e,r) \in \mathcal{O}} (r - \mathbf{u} \cdot \mathbf{e})^2 \tag{5}$$

As such, every node in the decision tree is associated with an optimal user embedding $\mathbf{u}_{\{L,D,U\}}$. When the interview terminates and the user arrives as a leaf node, the optimal embedding at this node is assigned to the user. We denote this post-interview embedding of user u as $\mathbf{T}(\mathbf{u})$.

In our experiments, we add $\lambda_u \|\mathbf{u} - \mathbf{u_p}\|_2$ as a hierarchical regularisation term to the user embedding objective in order to reduce overfitting as the tree grows [5], where $\mathbf{u}_p$ is the user embedding assigned to the parent node. At the root node, we simply set this term as $\|\mathbf{u}\|_2$.

With $\mathbf{T}(\mathbf{u})$, we can predict ratings for u on entity e with the inner product between the vectors, that is $\mathbf{T}(\mathbf{u}) \cdot \mathbf{e}$. After constructing the decision tree, we can infer the optimal entity embeddings in a similar fashion, minimising the squared error between predicted and observed ratings, regularised by the l_2-norm of the entity embeddings:

$$\mathbf{e} \leftarrow \arg\min_{\mathbf{e}} \sum_{(u,e,r) \in \mathcal{O}} (r - \mathbf{T}(\mathbf{u}) \cdot \mathbf{e})^2 + \lambda_e \|\mathbf{e}\|^2 \tag{6}$$

The influence of the user- and entity embedding regularisation terms are controlled by the hyperparameters λ_u and λ_e, respectively. We determine λ_u and λ_e parameters using grid search, searching for configurations in $\{0.01, 0.03, 0.06\}$. During training, the user- and entity embeddings are updated by means of Alternating Least Squares (ALS) updates.

4.2.2 *Local Representative-based Matrix Factorisation*

LRMF leverages Representative-based Matrix Factorisation (RBMF) in order to both construct the interview and predict ratings [27]. In RBMF, a user is represented not by a latent feature vector as is the case in fMF, but by a vector of the user's ratings on a set of *representative entities.*

In LRMF, we have a notion of l_1 global and l_2 local questions, resulting in a $l_1 + l_2$-length interview. The global questions are questions posed to all users and are used to distribute the users across the nodes of the decision tree as in fMF. When the global interview terminates and the user has been assigned to a leaf user group by the decision tree, l_2 local, fixed questions are posed to the user.

A group of users $\mathcal{U}_g \subset \mathcal{U}$ with group index g is represented by the matrix $\mathbf{U}_g = \left[\mathbf{U}_g^{(1)}; \mathbf{U}_g^{(2)}; \mathbf{1}\right] \in \mathbb{R}^{n_g \times l_1 + l_2 + 1}$ where $\mathbf{U}_g^{(1)}$ and $\mathbf{U}_g^{(2)}$ are the vector representations of the users' $\in \mathcal{U}_g$ ratings for the l_1 and l_2 global and local questions, respectively, $\mathbf{1} \in \mathbb{R}^{n_g}$ denotes a column vector of ones as to capture group-level bias, and n_g is the number of users in $\mathcal{U}_g$. When the interview is done, the user is represented by the vector $\mathbf{u} = [1\ ; R(u, e) | e \in \mathcal{Q}] \in \mathbb{R}^{l_1 + l_2 + 1}$ where $\mathcal{Q}$ is the set of global and local questions.

As in fMF, entities are represented as latent feature vectors. The entity embeddings are represented by the matrix $\mathbf{V} \in \mathbb{R}^{k' \times m}$ where k' is the number of latent features and m is the number of entities. In order to predict ratings, the features and dimensionality between the user and entity embeddings are mapped by means of a group-specific transformation matrix $\mathbf{T}_g \in \mathbb{R}^{k \times k'}$ where $k = l_1 + l_2 + 1$. The complete objective of the LRMF model is presented in Equation 7 [27][2].

$$
\begin{aligned}
&\min_{\mathfrak{G}, \mathbf{T}_g, \mathbf{U}_g^{(1)}, \mathbf{U}_g^{(2)}, \mathbf{V}} \\
&\sum_{g \in \mathfrak{G}} \|\mathbf{R}_g - \left[\mathbf{U}_g^{(1)}; \mathbf{U}_g^{(2)}; \mathbf{1}\right] \mathbf{T}_g \mathbf{V}\|_F^2 + \alpha \|\mathbf{T}_g\|_F^2 + \beta \|\mathbf{V}\|_F^2 \\
&\quad \text{s.t.} \\
&\quad \cup_{g \in \mathfrak{G}} \mathbf{R}_g = \mathbf{R} \text{ and } \mathbf{R}_g \cap \mathbf{R}_{g'} = \emptyset, \qquad (7) \\
&\quad \mathbf{R}_g \in \mathbb{R}^{n_g \times m} \text{ and } \mathbf{R}_g \subset_{n_g, m} \mathbf{R}, \\
&\quad \mathbf{U}_g^{(1)} \in \mathbb{R}^{n_g \times l_1} \text{ and } \mathbf{U}_g^{(1)} \subset_{n_g, l_1} \mathbf{R}, \\
&\quad \mathbf{U}_g^{(2)} \in \mathbb{R}^{n_g \times l_2} \text{ and } \mathbf{U}_g^{(2)} \subset_{n_g, l_2} \mathbf{R}, \\
&\quad \mathbf{V} \in \mathbb{R}^{k' \times m}, \mathbf{T}_g \in \mathbb{R}^{k \times k'}
\end{aligned}
$$

where $\mathfrak{G}$ denotes a group division and $g \in \mathfrak{G}$ a group index, $\mathbf{R} \in \mathbb{R}^{n \times m}$ is the ratings matrix, n and m are the number of users and entities, respectively, and α and β are hyperparameters for controlling the influence of regularisation terms of the entity embedding and transformation matrices, respectively.

While the l_1 global questions are selected in similar fashion to that of fMF, i.e., by greedily selecting questions that minimise rating prediction error, the l_2 fixed local questions are selected by means of the Maxvol algorithm as proposed by [36]. While the details remain unclear[3], it seems that the l_2 local questions are selected by using Maxvol to find the maximal-volume $l_2 \times l_2$ submatrix of the entity embeddings matrix $\mathbf{V}$ where the columns represent the l_2 local question entities. However, in order to find an $l_2 \times l_2$ submatrix, the original matrix must be of shape $n \times l_2$ for some $n > l_2$ according to the specification of the Maxvol algorithm [36]. This forces the dimensionality of the entity embeddings k' to be exactly l_2, though this is not the case in the original paper [27]. As we cannot work around this issue trivially, we set the entity embedding dimensionality k' to l_2 in our experiments.

[2]We refer the reader to the original paper [27] for a detailed description of how the objective can be optimised.

[3]We have contacted the authors of [27] but have yet to receive a satisfactory explanation for this issue.

4.2.3 *Greedy adaptive interviewers*

Both fMF and LRMF choose questions using a greedy selection algorithm, selecting the questions that optimally minimise the objective at a given node in the decision tree. While the greedy selection strategy carries with it problems regarding optimality, the introduction of a decision tree is a simple and sound implementation of a dynamically adapting interviewer. We can apply the same principles in the aforementioned greedy interviewer, splitting users into groups according to their answers and building a decision tree, selecting the nodes that optimise an arbitrary performance metric over the rankings produced by an arbitrary recommender. An example decision tree generated by this approach is seen in Figure 4.

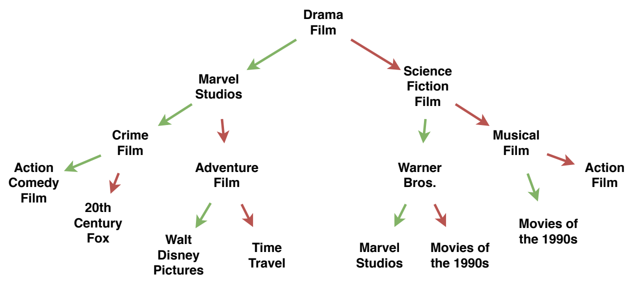

Figure 4: A 4-length interview generated by a greedy adaptive interviewer.

While simple in nature, we expect this approach to perform better than fMF and LRMF as we ensure that the questions selected optimise ranking performance rather than rating prediction which relate to separate tasks within recommendation. Furthermore, we impose no restriction on the underlying interviewer, allowing for cold-start interviews using recommenders that might perform better than MF-based models designed for rating prediction.

We formalise the greedy adaptation algorithm, Adaptive-Greedy (AG), for building an interview decision tree in Algorithm 1. When splitting users on a question, we follow the strategy of [27] and merge users disliking and not knowing the question. We also opt to select a set of fixed questions for the remainder of the interview if the set of users at a node becomes too small to warrant further splitting, which are determined through Equation 1 for the subset of users at that particular node. In a similar vein, for large datasets we can again follow the example of [27] and stop the adaptive constructing at a certain depth, using fixed questions for the remainder of the interview. In our experiments, we set the minimum number of users to 10 and set no maximum depth on the adaptive construction of the tree.

Algorithm 1: The AG algorithm.

```
input  : A set of users $\mathcal{U}$, question candidates $\mathcal{E}' \subseteq \mathcal{E}$, a
         recommender $S$ and a ranking quality function
         $F$.
output: A decision tree.
1  $\mathcal{O}_{\mathcal{U}} := \{\}$ # Initially empty answer set
2
3  return BuildTree($\mathcal{U}$, $\mathcal{E}'$, $\mathcal{O}_{\mathcal{U}}, F, S$)
4  Function BuildTree($\mathcal{U}'$, $\mathcal{E}'$, $\mathcal{O}_{\mathcal{U}'}, F, S$):
5      scores := QuestionScores($\mathcal{U}'$, $\mathcal{E}'$, $\mathcal{O}_{\mathcal{U}'}, F, S$)
6      if $\mathcal{U}'$ is too small or at max adaptive depth then
7          $n$ := number of remaining interview questions
8          questions := $\{q_i | (q_i, s) \in$ sorted(scores)$\}$ for $i \leq n$
9          return fixed-questions node
10     $q := \arg\max_q \{s | (q, s) \in \text{scores}\}$
11     if at max depth then
12         return node with $q$ as the question
13     $\mathcal{E}' := \mathcal{E}' \setminus \{q\}$
14     $\mathcal{U}'_L, \mathcal{U}'_D$ := split users liking/disliking $q$
15     $\mathcal{O}_{\mathcal{U}'_L}, \mathcal{O}_{\mathcal{U}'_D}$ := add users' ratings of $q$
16     L := Branch likes to BuildTree($\mathcal{U}'_L$, $\mathcal{E}'$, $\mathcal{O}_{\mathcal{U}'_L}, F, S$)
17     D := Branch dislikes to BuildTree($\mathcal{U}'_D$, $\mathcal{E}'$, $\mathcal{O}_{\mathcal{U}'_D}, F, S$)
18     return node with L and D as child nodes
19 Function QuestionScores($\mathcal{U}'$, $\mathcal{E}'$, $\mathcal{O}_{\mathcal{U}'}, F, S$):
20     scores = {}
21     foreach $q \in \mathcal{E}'$ do
22         $\mathcal{O}_{\mathcal{U}'} = \mathcal{O}_{\mathcal{U}'} \cup$ the answers of $\mathcal{U}'$ on $q$
23         score := $\frac{1}{|\mathcal{U}'|} \sum_{u \in \mathcal{U}'} F(S(\mathcal{O}_{\mathcal{U}'}, u))$
24         scores := scores $\cup \{(q, \text{score})\}$
25     return scores
```

As previously mentioned, the informativeness of a question is evaluated in context of the previously selected questions, as testing all possible combinations quickly becomes computationally intractable. Even so, greedily constructing a decision tree easily becomes a very complex procedure, as the computational complexity is exponential in the height of the tree. The algorithm must exhaust all question candidates at every node of the tree, and with a three-way node splitting procedure as in fMF, the number of nodes to exhaust with an interview length (i.e., tree height) of m is

$$1 + 3^1 + 3^2 + \dots 3^m = \frac{3^{m+1} - 1}{3 - 1} \tag{8}$$

Depending on the complexity of exhausting the candidate questions, long interviews will render this approach practically infeasible, which is explained in detail in Appendix D. As such, we deem it worth considering interview construction strategies that do not suffer under similar limitations.

4.2.4 Exploratory adaptive interviewers

Instead of selection questions greedily, we may be able to generate a more effective interview if we allow the interviewer to explore more freely in the questions it selects. Additionally, both fMF and LRMF learn the interview, and generate recommendations, by minimising the difference between predicted and observed ratings. While this can be an effective strategy for some ranking tasks, we may be interested in producing rankings that optimise more complex measures such serendipity- or diversity-based metrics as those proposed by [28]. However, seeing as such metrics are not always directly differentiable, we cannot trivially construct a matrix factorisation objective that optimises these through gradient descent. We can, however, leverage reinforcement learning, specifically deep Q-learning [29, 30], to learn how to conduct interviews such that we reinforce any positive outcome including non-differentiable metrics. Finally, this form of deep learning should alleviate the problem of exponential computational complexity, as the structure and size of the models can be kept static regardless of the interview length.

Deep Q-Networks (DQNs). In deep Q-learning, the goal is to learn a Q function with domain $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ for predicting the reward $r \in \mathbb{R}$ for taking an action $a \in \mathcal{A}$ from state $s \in \mathcal{S}$. A DQN is a deep neural network that is trained to approximate Q.

In order to approximate Q through gradient descent, we need to define an objective for the DQN that is directly differentiable, regardless of the nature of the reward. To support this, [29] observes that every Q function respects the Bellman equation

$$Q^{\pi}(s, a) = r + \gamma Q(s', \pi(s')) \tag{9}$$

where $s \in \mathcal{S}$, $a \in \mathcal{A}$, $r \in \mathbb{R}$ is the reward of taking action a in state s resulting in the new state s', $\gamma \in [0, 1]$ discounts the accumulative reward, and π is the learned policy for selecting an action given some state $\in \mathcal{S}$. In essence, the Bellman equation states that the utility (in our case the reward) of taking an action a in a state s is the accumulative utility of taking the per-policy correct actions in all following states as discounted by γ. Note that when the interview is over, no future decisions exist, and thus γ is set to 0.

As Q must respect the Bellman equation, we can define the DQN objective as to minimise the DQN loss function:

$$L_{DQN} = \mathbb{E}_{s,a \sim p(\cdot)} \Big[(y - Q(s, a))^2 \Big] \tag{10}$$

where $y = \mathbb{E}_{s' \in E} \Big[r + \gamma \max_{a'} Q(s', a') | s, a \Big]$ is the target reward of taking action a from state s, $p(s, a)$ is a probability distribution over state-action transition pairs, and E is an emulating environment providing states as altered by actions [29].

We descend the gradient of Equation 10 w.r.t. the parameters of Q as to minimise the difference between the predicted reward of taking action a in state s, $Q(s, a)$, and the discounted sum of the actual reward observed and the highest possible reward the policy could obtain as predicted from the following state, $\gamma \max_{a'} Q(s', a')$. When L_{DQN} is minimised, the Q function represented by the DQN optimally respects the Bellman equation.

It is exactly the accumulative reward prediction that allows DQNs to overcome the optimality problems faced in greedy selection strategies as used in fMF, LRMF, and AG. The question selected at a given point in the interview does not have to be the most informative in isolation - instead, questions can now be selected if the network discovers that they afford much higher rewards when combined with other questions later on.

In training a DQN, we have two primary interacting components: an *agent* and an *environment* [29]. The agent, represented by the DQN, takes in a state representation and outputs an action in $\mathcal{A}$ to take. The environment maintains this state representation, and enacts the actions chosen by the agent, providing the agent with the new state as altered by the action taken. The flow of interaction between the agent and the environment is illustrated in Figure 5. In cold-start interviews, a state $s \in \mathcal{S}$ is simply a set of question-answer pairs.

The agent receives a state $s \in \mathcal{S}$ and outputs the next question q to ask in the interview as selected by a DQN with epsilon-greedy exploration. The environment receives q and retrieves the user's rating for the entity asked for. The environment adds q and the user's answers to s, resulting in the new state s'. If the interview is over, the environment uses an arbitrary recommender system S to rank all entities and computes a ranking quality metric, e.g., NDCG@k, as the reward for the interview. Otherwise, the reward is simply set to 0. From this interaction, we define a state transition tuple (s, q, s', r, t) with s being the original state, q being the question selected by the agent, s' being the state resulting from asking the question, r being the reward received, and t being a Boolean indicator of whether the interview is over or not. This tuple is stored in a replay memory $\mathcal{M}$.

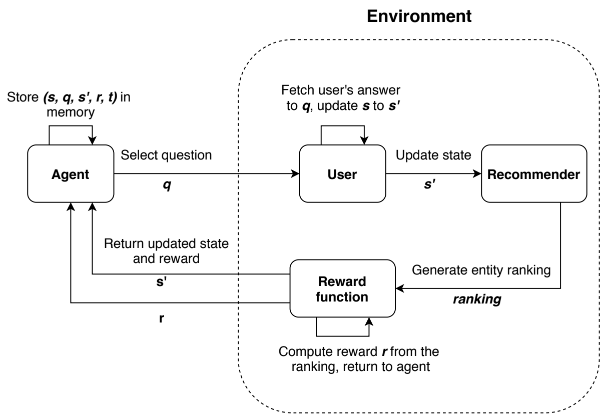
Environment
Store (s, q, s', r, t) in memory
Agent
Select question
q
Fetch user's answer to q, update s to s'
User
Update state
s'
Recommender
Return updated state and reward
s'
r
Reward function
Generate entity ranking
ranking
Compute reward r from the ranking, return to agent

Figure 5: Flow of interaction between the agent and environment in deep Q learning.

In order to eliminate temporal dependencies between training samples, we randomly sample mini-batches from $\mathcal{M}$ as proposed by [29] to decrease the likelihood of policy divergence. We define the learning algorithm formally in Algorithm 2. In the algorithm, ϵ_{dec} denotes the decay rate of the exploration probability ϵ. In our experiments, we set $\epsilon_{dec} = 0.999$.

We implement the DQN as a 3-layer Multi-Layer Perceptron (MLP) with 512 hidden units. The interview state is represented as a vector $\mathbf{s} \in \mathbb{R}^{|\mathcal{E}'|\cdot 2}$ where $\mathcal{E}'$ is the set of candidate interview questions, $\mathbf{s}_{i\cdot 2} = 1$ if the interviewer has asked for $\mathcal{E}_i$, and $\mathbf{s}_{1+i\cdot 2}$ is 1, -1, or 0 if the user answered like, dislike, or don't know to the question. Given a state, the DQN produces a vector of Q-values $q \in \mathbb{R}^{|\mathcal{A}|}$ with a predicted cumulative for all actions $a \in \mathcal{A}$. Formally, the predicted Q-values are defined as:

$$DQN(s) = L_3(L_2(L_1(\mathbf{s}))) \tag{11}$$

where $L_i(\mathbf{v})$ denotes the forward propagation of a vector $\mathbf{v}$ through the i'th layer of the DQN MLP s.t. $L_i(v) = \sigma(\mathbf{v}\mathbf{W}_i + \mathbf{b}_i)$ where $\mathbf{W}_i$ and $\mathbf{b}_i$ are the weights and biases of

Algorithm 2: DQN training algorithm.

```
input : A set of users $\mathcal{U}$, a recommender system $S$, a
        reward function $F$, interview length $k \in \mathbb{N}$,
        $\epsilon_{dec} \in [0,1]$ and $\alpha \in \mathbb{R}^+$
output: A trained DQN model
1  return TrainDQN($\mathcal{U}, 1, F, \epsilon_{dec}$)
2  Function TrainDQN($\mathcal{U}, \epsilon, F, \epsilon_{dec}$):
3      $Q$ := DQN with random parameters $\theta$
4      $\mathcal{M} := \{\}$
5      while not converged do
6          foreach $u \in \mathcal{U}$ do
7              $s$ := empty state
8              foreach $k' \in \{0, \ldots, k\}$ do
9                  $q$ := SelectQuestion($Q, \theta, s, \epsilon$)
10                 $a$ := get $u$'s answer to $q$
11                 $s'$ := update $s$ with $q$ and $a$
12                 $t := k' = k$
13                 if $t$ then
14                     $l$ := use $S$ to rank $\mathcal{E}$ given $s'$
15                     $r$ := determine quality of $l$ using $F$
16                 else
17                     r := 0
18                 $\mathcal{M} := \mathcal{M} \cup \{(s, q, r, s', t)\}$
19             $(s_b, q_b, r_b, s'_b, t_b)$ := sample mini-batch in $\mathcal{M}$
20             $target := \begin{cases} r_b, & \text{if } t_b \\ r_b + \gamma \max_{a \in \mathcal{A}} Q(s'_b, a|\theta), & \text{otherwise} \end{cases}$
21             $\theta := \theta - \alpha \cdot \nabla_\theta L_{DQN}$ given
                 $(target - Q(s_b, a_b|\theta))$ as described by
                 Equation 10
22             $\epsilon := \epsilon \cdot \epsilon_{dec}$
23     return $Q$
24 Function SelectQuestion($Q, \theta, s, \epsilon$):
25     $r$ := random([0, 1])
26     if $r < \epsilon$ then
27         return $\arg\max_{a \in \mathcal{A}}(Q(s, a|\theta))$
28     else
29         return random($\mathcal{A}$)
```

L_i, and σ is a non-linear activation function set to Rectified Linear Unit (ReLU) in our experiments.

Deep Deterministic Policy Gradient (DDPG). Generally, deep Reinforcement Learning (RL) is presented with a host of difficult-to-solve problems that lead to unstable and unreliable training sessions. While some of these can be alleviated by introducing target networks and replay memory buffers [29], resulting in a more stable learning process, the setting of cold-start interviews specifically introduces more problems. In DQN, questions are selected from $\mathcal{A}$ which, in theory, can be all the entities in the system. This is a very large action space, and since DQN chooses discrete actions from this space in order to build memories, a large amount of exploration is required for the model to properly construct effective interviews. Because of this, DQNs are generally not capable of handling high-dimensional action spaces [30]. To solve this, [30] introduces a generic, off-policy learning algorithm, DDPG, which in short facilitates the handling of high-dimensional action spaces in deep RL.

In DDPG, the agent maintains the interaction between an *actor* and a *critic*. The actor, $A : \mathcal{S} \to \mathcal{A}$ where $\mathcal{A} = \mathbb{R}^k$ for some k takes a state and produces an action representation, for example as represented by a latent feature vector. The critic, $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ takes a state and an action as produced by the actor, and outputs the predicted reward of taking that action in that state. While we will not cover the algorithm in detail, we will briefly describe the learning objectives of each component. Given a transition tuple $(s, q, s', r't)$, the critic's objective is to minimise the difference between the predicted reward and the reward observed for taking action a in state s, that is

$$\min(r - Q(s,a) + \gamma \cdot Q(s', A(s'))) \tag{12}$$

Under the assumption that the critic is able to learn how to predict rewards accurately, the objective of the actor is naturally to produce actions that lead to high rewards. By descending the gradient of the sampled policy gradient [30], the objective of the agent is essentially to maximise the rewards predicted by Q, that is

$$\max(Q(s,a)) \tag{13}$$

In [31], the DDPG algorithm is applied to facilitate effective action selection over large, discrete action spaces. In short, actions are represented as a collection of pre-defined information about the actions, for example, a feature vector $\in \mathbb{R}^n$ of n action-specific features. When the agent produces a proto-action vector $\in \mathbb{R}^n$, we first find the k-nearest action vectors $\mathcal{A}' \subset \mathcal{A}$ using a k-NN-like approach. With this reduced set of actions, we can choose the action with the highest predicted reward, that is $\arg\max_{a \in \mathcal{A}}(Q(s,a))$. In our implementation, we represent actions by means of MF-embeddings using 5 latent features, though any type of embedding can be used as long as the actions are represented within the same feature space.

We implement both the actor and critic as 3-layer MLPs with 512 hidden units. The interview state is represented as in DQN. The actor takes a state vector $\mathbf{s}$ and returns a proto-action vector $\mathbf{a} \in \mathbb{R}^k$ where k is the number of latent action features. Given a state vector $\mathbf{s}$ and proto-action vector $\mathbf{a}$, the critic passes $\mathbf{s}$ through the first layer of the MLP, concatenates the processed state- and action vectors, and propagates the concatenated vector through the rest of the network to produce a predicted reward $r \in \mathbb{R}$. Formally, the predicted reward from the critic is defined as:

$$Q(\mathbf{s}, \mathbf{a}) = L_3(L_2([L_1(\mathbf{s}); \mathbf{a}])) \tag{14}$$

where $L_i(\mathbf{v})$ denotes the forward propagation of a vector $\mathbf{v}$ through the i'th layer of the critic MLP $L_i(\mathbf{v}) = \sigma(\mathbf{v}\mathbf{W}_i + \mathbf{b}_i)$ where $\mathbf{W}_i$ and $\mathbf{b}_i$ are the weights and biases of the i'th layer, and σ is a non-linear activation function set to ReLU in our experiments.

4.3 Recommenders

As described in subsection 3.2, we consider three separate paradigms of recommenders in this work, with one not lending to incremental models, i.e., an embedding-based recommender such as MF. In the following, we consider how embedding-based, warm-start recommenders can support cold-start recommendations and the iterative addition of user ratings. We also outline some simple additions to PPR. Finally, we detail how cold-start recommendations can be provided using neighbourhood-based recommenders.

4.3.1 *Cold-start user embeddings*

In embedding-based recommenders (e.g., MF-based recommenders), a feature vector is learned for every individual user u and entity e. These embeddings are learned to facilitate that the dot product of the vectors $\mathbf{u} \cdot \mathbf{e} \sim R(u, e)$. Consequently, this also means that a cold-start user arriving at the system will not have a feature vector associated with them. In the following, we explore a strategy for handling this issue.

The rating from a user u and an entity e is predicted as $\mathbf{u} \cdot \mathbf{e}$. This product inherently encodes the similarity between the vectors, and the larger the distance between the vectors, the lower the predicted rating will be.

Following this, we want the user's embedding to be as similar as possible to the entities the user will like. Similar to the initial approach of [6], the user can instead by represented as the average embedding of the entities they have liked:

$$\mathbf{u} = \frac{1}{|\mathcal{E}_u^+|} \cdot \sum_{e \in \mathcal{E}_u^+} \mathbf{e} \tag{15}$$

where $\mathcal{E}_u^+$ are the entities that u has liked. For a completely new user with no observed ratings, the embedding can be expressed simply as the average entity embedding, that is

$$\mathbf{u} = \frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} \mathbf{e} \tag{16}$$

4.3.2 *Personalised PageRank*

As opposed to MF, PPR is a good fit for cold-start recommendations as it does not depend on a predefined or learned representation of a user. Since all users are represented by their ratings that serve as source nodes in calculating the PPR of entities, we can use PPR to generate cold-start recommendations just as easily as with warm-start users.

In our warm-start experiments with MindReader, we found that using PPR with a KG as the underlying graph performed well, particularly on the long-tail entities [4]. Yet, our hyperparameter tuning of PPR was limited to determining an optimal damping factor. As such, we here describe different approaches to improving PPR for top-K recommendations.

Graph variants. As in [4], we create three different PPR models utilising different graphs: PPR-KG, which uses the MindReader KG, PPR-COLLAB which uses a graph of all user-entity rating edges, and PPR-JOINT which uses a combination of the two. This division of graph variant is similar to those in [37, 38].

Non-uniform probabilities. In [4], we considered only the likes of users in our PPR implementation, using a uniform probability distribution for all liked entities in the teleportation probability vector, i.e., the random walk is only allowed to reset at a user's liked entities. Yet, previous work [38] showed that it is possible to modify the distribution of probabilities for increased performance on sparse datasets. In this work, we divide a user u's rating as their liked set $\mathcal{E}_u^+$, a disliked set $\mathcal{E}_u^-$, and the rest as $\mathcal{E}_u^r = \mathcal{E} \setminus (\mathcal{E}_u^+ \cup \mathcal{E}_u^-)$. Then, the random walk can reset as both a user's likes, dislikes, and non-rated entities, with a weight for each category determined through grid search.

Linear combination. In [39], they use a linear combination of PageRank scores in order to generate more accurate query results. Specifically, they compute PageRank scores for each topic to generate more context specific ranking scores for the documents to retrieve, thus ensuring that the query matches the retrieved documents better. Similarly, we propose a linear combination of PPR rankings on different underlying graphs in order to optimally combine ratings of different sentiments. Different from the approach of [39], we combine PPR scores from several different graphs, each personalised towards a user's answers. As such, we aim to learn a static combination of dynamic PPR scores, while that of [39] conducts a dynamic combination (weights determined dynamically as query similarity) of static PPR scores. A graph is created for each rating type, i.e., a "Like", "Dislike", and "Don't know" graph. Only nodes that have the same rating as the rating types graph are set as source nodes, resulting in PPR scores directly representing the importance for that rating type. We refer to the collection of graphs as $\mathcal{G}_\mathcal{C}$ and compute the combined PPR as:

$$PPR_\mathcal{C}(u) = \sum_{\mathcal{G}_c \in \mathcal{G}_\mathcal{C}} w_c \cdot PPR(\mathcal{G}_c, R_c(u)) \tag{17}$$

where PPR is a function that, given a graph and source nodes as per the ratings of u, obtains a vector of weights for each node, representing their PPR score of all entities. Furthermore, w_c is the weight assigned to rating type c, and R_c is the nodes with rating c from user u, where $c \in \mathcal{C}$. The weight indicates the effect different rating types have on the prediction score. Empirically, we observe that the "Like" graph is given a positive weight, "Dislike" a negative, and "Don't know" a weight between the two, which is in compliance with our intuition. Furthermore, we observe that "Don't know" ratings often have a small negative weight, indicating that these ratings may carry a slightly negative sentiment.

Just as ratings can be combined, so can PPR rankings over different graph variants. As stated previously, PPR-JOINT combines the graphs from PPR-COLLAB and PPR-KG. This forces the joint model to navigate both graphs, but one graph might provide more useful information than the other. We therefore consider a linear combination of PPR-KG and PPR-COLLAB defined as:

$$PPR_L(u) = PPR_\mathcal{C}^{\text{KG}}(u) + PPR_\mathcal{C}^{\text{COLLAB}}(u) \tag{18}$$

where $PPR_\mathcal{C}^{\text{KG}}$ is $PPR_\mathcal{C}$ using the KG as graph, and COLLAB is the collaborative graph. Intuitively, the model learns to weigh each rating type for the two graphs, as illustrated in Figure 6, with separate weights for each graph variant. Computing the combined PPR ranking is of course more complex than computing the PPR ranking from a single graph, though the increase in complexity is linear in the number of graphs.

Learning combination weights. While we can determine the optimal weights by means of grid search, this has complexity $\mathcal{O}(n^m)$ where n is the number of weights to test and m is the number of graph variants, thus rendering this approach intractable if the weights are selected from a continuous space. However, if we restrict the weights to a small set of discrete configurations, we run the risk of not including the optimal combinations in the restricted set. Instead, we can observe that Equation 17 describes a polynomial parameterised by the weight vector $\mathbf{w}$ where each weight $w_c \in \mathbf{w}$ is associated with a PPR-based ranking function.

In [40], they present a number of differentiable loss functions that allow us to measure how well a model is performing in the ranking task when provided with two samples for which the correct ranking is known. Formally, the generic

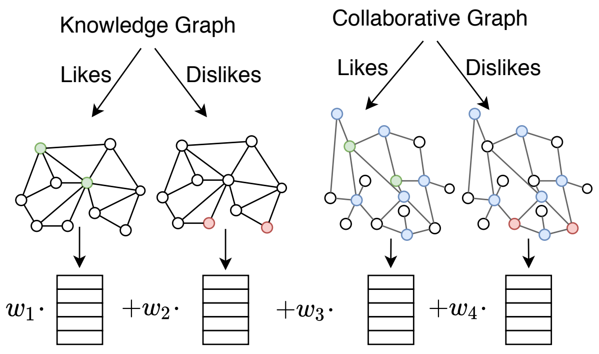

Figure 6: Illustration of PPR_L, using a user's likes and dislikes as separate source nodes. Blue nodes represent users, while other colours represent a user's ratings, and white nodes represent entities.

loss function is defined as

$$L_p(\theta, f, u, x, y) = \sigma(f(u|\theta)_x - f(u|\theta)_y) \tag{19}$$

where, f is a ranking function (e.g., PPR) parameterised by θ, σ is an arbitrary function defining the ranking error based on the difference between the scores of x and y as predicted by f given a user u, and x and y are two sample entity indices for the PPR scores, where $x > y$ is the ground truth ranking [40].

Thus, given two sample entities $e, d \in \mathcal{E}$ and a user $u \in \mathcal{U}$, we can approximate the optimal configuration of $\mathbf{w}$ by iteratively descending the gradient of L, that is

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_\mathbf{w} L(\mathbf{w}, PPR_L, u, e, d) \tag{20}$$

where $PPR_L(u)$ in this case is a function $PPR_L : \mathcal{U} \rightarrow \mathbb{R}^{|\mathcal{E}|}$ that returns the ranking score of all entities as defined in Equation 18, α is the learning rate, and $\nabla_\mathbf{w} L$ defines the vector of partial derivatives w.r.t. each weight $w_c \in \mathbf{w}$.

As for the σ function, there are several to consider. As an example, one could consider a pairwise ranking loss function such as a margin loss [40]:

$$\sigma(z) = [\gamma - z]_+ \tag{21}$$

where z is the difference between the scores of the sample entities and γ is the desired default margin between entities of different ranks. By descending the gradient of L using a margin loss, we ensure that the difference between the provided samples is maximised. Intuitively this enforces a margin between the positive and negative sample s.t.

$$f(u|\theta)_x + \gamma < f(u|\theta)_y \tag{22}$$

During training, we see a tendency for the weights to explode as the model learns that larger weights result in better separation. The lack of convergence may be due to uniform sampling of negative items, and may require many iterations in order to converge [41]. We therefore utilise a sampling strategy such that the most informative semi-hard sample is chosen, similar to the online approach in [42]. A semi-hard sample simply means a sample that violates Equation 22 while being ranked correctly compared to the positive sample. For each mini-batch we sample pairs that violate Equation 22. For each positive sample x, we find the semi-hard negative sample y that maximises the distance to x as it should be the easiest of the violating samples to solve. If no such sample exist, we choose the sample closest to the positive sample analogous to the approach in [43]. We see in practice that our sampling approach converges faster and achieves better results, though further testing and experimenting is needed for more stable results.

Although we utilise PPR to calculate the score for each graph variant and rating type, we could instead use node embeddings generated by node2vec and calculate the similarity from an unrated item to all rated for a user as in [37]. They divide their KG into sub-graphs containing specific relationship types extended with liked ratings of users. They further experiment with a listwise learning algorithm, LambdaMart, for combining similarity scores from different sub-graphs, but find that simple aggregate functions outperform this when utilising collaborative rating in the sub-graphs. We opt to use PPR over node2vec, as it took multiple hours to create node2vec embeddings for a single graph and they propose 270 different parameter combinations, making it impractical to find an optimal embedding. Furthermore, as we want to utilise the negative ratings available in our dataset, we cannot use an aggregate function and instead use a pairwise loss function.

4.3.3 Neighbourhood-based recommenders

We found neighbourhood-based models to perform very well in the warm-start setting in [4]. In particular the user-k-NN model utilised the DEs especially well, and as such we modify our existing implementation to handle new users in a cold-start setting. A new user u is represented as a vector $\mathbf{v}^u$, where $|\mathbf{v}^u| = |\mathcal{E}|$, s.t. the vector is defined as:

$$\mathbf{v}_i^u = \begin{cases} R(u, e_i) & \text{if } (u, e_i, R(u, e_i)) \in \mathcal{O}_u, \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

for all $e_i \in \mathcal{E}$. With a user vector representation, we can employ standard user k-NN similarity measures for recommendation, e.g., cosine similarity and Pearson correlation [33].

5. DATA COLLECTION

In our previous work we designed and implemented MindReader, a platform for data collection within the movie domain, in order to elicit both RE and DE ratings from real users [4]. In total, we collected 102,160 ratings from 1,174 users on 10,030 KG entities. Furthermore, we showed how the collection approach balances coverage and co-occurrence of ratings. However, as already pointed out in the previous work, there are several issues with the data collection approach. In the remainder of this section we will outline these issues and how we have addressed them in a new version of MindReader. Finally, we present rudimentary statistics on the latest version of the dataset, which will be used in the evaluation of this work.

5.1 Quality of recommendations

While providing high quality recommendations was not and still is not the primary concern of MindReader, providing such recommendations can be beneficial for the collection of data. With a perceived low quality of recommendations, we risk that many users will go through no more than one iteration of the interview. Contrary to this, a perceived high quality of recommendations may encourage users to not only complete more iterations, but to use the application on a regular basis as a means of getting recommendations. This is beneficial not only in terms of the overall number of ratings, but also for CF models that rely on co-rated entities between user pairs for finding similar users.

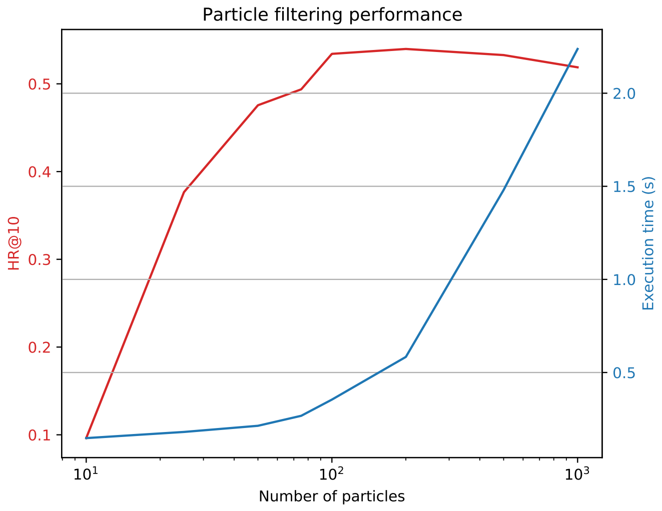

Figure 7: HR and running time in seconds relative to number of particles.

In our initial approach, recommendations were provided by sampling REs adjacent to the user's entity preference based on global PageRank. The approach was chosen over the SOTA approach of PPR due to its much lower computational complexity, with precision of recommendations and limitation to 1-hop neighbours as a compromise. However, a recent work suggests that computing the PPR values of a KG can be approximated with a Particle Filtering (PF) approach, thereby reducing running time while ensuring high quality of ranking [34]. As such, we consider whether PF can serve as an adequate replacement for PPR in our use case.

5.1.1 Particle filtering

Before implementing PF in MindReader, we experimented with different numbers of particles, comparing the quality of the ranking and running time. We used the Neo4j[4] implementation of PF provided in [34] and computed the Hit Ratio (HR) and average running time. For each user, we rank a positive sample (i.e., an entity liked by them) and 100 negative samples (i.e., entities they have not interacted with) and compute the HR@10 following Equation 25.

From Figure 7 we observe that both HR and running time increase with an increasing number of particles. Specifically we observe an increase in HR from 9.86% at 10 particles to 52.08% at 100 particles, while execution time increases from 0.14 seconds to 0.35 seconds, respectively. Following 100 particles, the increase in HR is relatively small compared to the increase in execution time. For reference, the Neo4j implementation of PPR yielded a HR of 32.63% and average running time of 1.04 seconds with $\alpha = 0.85$. As such, using PF with 100 particles should ensure high-quality recommendations within reasonable runtime.

[4]`https://neo4j.com/`

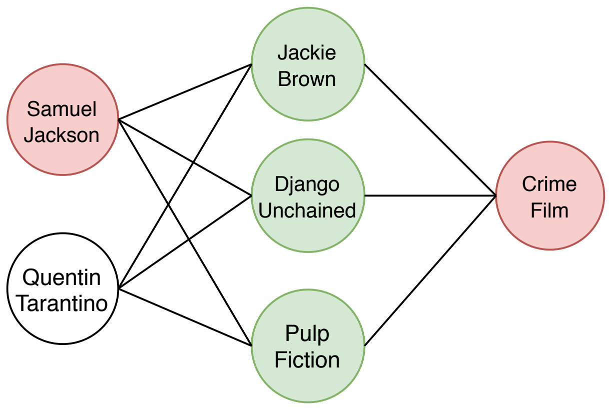

Figure 8: Limitations of 1-hop exploration.

5.2 Related entities

In our initial approach to finding related entities during the exploration phase, we used a 1-hop approach as when retrieving recommendations. The difference between the two approaches is that when exploring we sample from additional entity types (e.g., genres and actors).

An issue with this approach is that it tends to result in highly localised exploration of the graph. For example, consider a scenario as in Figure 8 where the user has provided feedback for the director Quentin Tarantino. Under 1-hop exploration, the next sequence of related entities would be limited to his directed movies, marked as green in the example. To further ask about entities related to Quentin Tarantino, we would require further feedback on one of the movies that are 1-hop connected to these entities, which may not be possible due to the constrained duration of the interview.

In the previous version of MindReader, localised exploration was mitigated by including random entities during the exploration phase. To better address this issue and to show more related entities, we use PF as in the new recommendation approach, sampling from the different entity types as before. Following this approach we avoid issues such as in Figure 8 since the related entities will not be constrained to 1-hop connected entities. Despite this change to the exploration phase, we will continue to include random entities in order to increase diversification of the exploration.

5.3 Scaling question frequency

In [4] we observed that the sampling approach during the exploration phase tended to result in high coverage for smaller entity groups (e.g., decades) and low coverage for larger entity groups (e.g., actors and directors). This is the result of following a round-robin approach, where we shuffle the entity groups $\mathcal{E}_g$ where $g \in \{Movie, Person, Decade, Company, Category\}$ and iteratively pick an element from each group. To ask more questions about larger groups of entities, we propose to sample each group with a probability based on the size of the entity groups, such that questions on larger groups are more frequent. With this approach, we sample an entity from entity group g with probability

$$P(g) = \frac{\log |g|}{\sum_{g \in \mathcal{E}_g} \log |g|} \tag{24}$$

Measure	Old KG	New KG
# nodes	18,707	
# edges	198,452	199,802
Minimum degree	4	
Median degree	10	
Average degree	21	
Maximum degree	4,454	
# connected components	1	

Table 2: Comparison between the new and old MindReader KG.

where $|g|$ denotes the number of entities in group g. We use the log frequency to avoid small groups (e.g., decade) being underrepresented during sampling. Furthermore, we impose no hard restrictions on the number of times a specific group can be sampled from.

5.4 Statistics

In Table 3 we compare rudimentary statistics for the new version of the MindReader dataset, called MR-170K, to the previous MR-100K version. As in [4], we follow the convention of appending ALL to the versions with all rating types, and BIN to the subset with only binary ratings (i.e., like and dislike), and for all cases we consider only users with a full completion of the interview.

In total, we collected an additional $85,330$ ratings from 782 new users. When comparing the variants with all ratings, the number of covered entities has increased by 44.09% for DEs and 2.97% for REs. As a result of the increased coverage, the density of ratings on DEs has decreased by 24.17%, while the density of ratings on REs is rather stable. We posit that the decrease in density of DE ratings can be attributed to the new exploration approach of MindReader, in addition to scaling questions with entity group sizes which results in more questions on the long-tail entities for larger groups.

In addition to extending the MindReader dataset, we updated the MindReader KG with “sequel” relations, allowing for triples such as $\langle \text{Deadpool 2}, \text{Sequel To}, \text{Deadpool} \rangle$. The updated statistics for the KG are shown in Table 2.

6. EVALUATION

For our overall evaluation strategy we follow an approach similar to what has been used in existing works [5, 22, 27]. First, we split the users of our dataset into two disjoint sets for training and testing, containing 75% and 25% of users, respectively. For the training dataset, all ratings associated with the users are assumed fully available and used for learning each model, effectively constructing the interview process. Conversely, the ratings for test users are only available when provided during the interview process. To cover all users in the dataset as test users, we perform 4-fold cross-validation and report the average results. In all experiments, we produce a ranking of REs for all test users following the interview. However, the entities to rank differ amongst our experiments as we consider performance on both the recommendation and separation task.

Recommendation task. To measure how well the models recommend REs, we employ a Leave-One-Out (LOO) evaluation approach where a random positive sample, i.e., liked RE, is left out and ranked against 100 negative samples, i.e.,

Datasets	#Users	#Ratings		#Entities (coverage %)		Density	
		DE	RE	DE	RE	DE	RE
MR-170K-ALL	1,736	79,577	95,295	6,581 (47.80%)	4,879 (98.76%)	0.69%	1.12%
MR-170K-BIN	1,736	44,410	40,145	3,104 (22.54%)	3,648 (73.84%)	0.82%	0.63%
MR-100K-ALL	954	39,592	49,950	4,567 (33.17%)	4,737 (95.89%)	0.91%	1.11%
MR-100K-BIN	954	22,827	18,106	1,997 (14,50%)	3,022 (61.17%)	1.20%	0.63%

Table 3: Rudimentary statistics comparing the new and old version of the MindReader dataset.

REs that the user has not interacted with. We employ the LOO approach specifically to avoid removing too many RE ratings, which would otherwise favour DE questions during interviews. To quantify the quality of the recommendation, we evaluate according to the presence and position of the left out entity in the predicted ranking. Furthermore, we employ various methods for negative sampling as explained in subsection 6.4.

Separation task. To measure how well the models separate REs rated by the user, we rank three entities: an entity liked by the user, an entity disliked by the user, and an entity to which the user explicitly answered "don't know". To quantify the quality of the ranking, we compare the predicted ranking against the true ranking as defined in Definition 2.2.

6.1 Dataset

We evaluate on the MR-170K dataset whose statistics are presented in subsection 5.4. As in [4], we include only users with a full completion of all MindReader phases. To preserve as many ratings as possible, we perform limited preprocessing for the two tasks. For the recommendation task, all users must have at least one positive rating, while for the separation task, all users must have at least one positive, one negative and one neutral (i.e., "don't know") rating.

6.2 Reproducibility

To support full reproducibility of our evaluation, the experimental setup and cold-start framework is fully containerised with Docker[5]. As part of our cold-start framework, we provide a pipeline to facilitate downloading and partitioning of data, as well as conducting experiments from a single script. While the details of the framework are beyond the scope of the main body of this work, we refer the reader to Appendix A for further details and instructions on accessing the framework.

6.3 Metrics

As covered by the recommendation and separation tasks, we employ a holistic evaluation strategy in that we evaluate the quality of the recommendations made directly, as well as the models' ability to correctly separate entities of varying sentiments.

Recommendation task metrics. As per the problem definition, we are interested in model performance in the top-K recommendation task. Similarly to [4], we evaluate the quality of recommendations made using measures HR@K and NDCG@K as proposed by [44, 45]:

$$\text{HR@}K = \frac{\#\text{Hits@}K}{\#\text{Users}} \tag{25}$$

[5]`https://docker.com`

In HR@K, we consider a recommendation a hit in the LOO setting if the left-out entity appears in the top-K list. Since this is a coarse measure of quality that doesn't consider the position of the left-out entity in the list, we further consider NDCG@K:

$$\text{NDCG@}K = \sum_{i=1}^{K} \frac{2^{rel(i)} - 1}{\log_2(i+1)} \tag{26}$$

where $rel(i) = 1$ if the i^{th} entity in the top-K list is relevant to the user (i.e., is the left-out entity), otherwise 0.

Besides HR@K and NDCG@K, we consider a serendipity measure proposed by [28] in order to gain deeper insights into a recommender systems ability to avoid "obvious" recommendations and instead generate surprising and useful recommendations, which is an important quality pertaining to recommender systems [46, 47]

We follow the definition of [28] and define $\mathcal{PM}$ as the set of entities recommended by a primitive model, e.g., one that simply recommends the most popular entities. We conversely define $\mathcal{RS}$ as the set of entities recommended by a non-trivial recommender system. The non-obvious or unexpected set of entities is then $\mathcal{UX} = \mathcal{RS} \setminus \mathcal{PM}$. We now define the serendipity measure SER@K:

$$\text{SER@}K = \frac{\sum_{i \in \mathcal{UX}} rel(i)}{|\mathcal{UX}|} \tag{27}$$

where $rel(i) = 1$ if the entity i is relevant to the user, otherwise 0, and $|\mathcal{RS}| = |\mathcal{PM}| = K$.

Finally, we consider a diversity-based metric for evaluating the performance of the recommending models, as proposed by [28]. Continuing in the vein of non-trivial recommendations, we seek to determine whether a model simply recommends the same entities to every user, or if it is capable of recommending a wide variety of different entities to different users. We consider the aggregate diversity of entities given the entities recommended to every user $u \in \mathcal{U}$. Let $\mathcal{I}$ denote all REs, and let $\mathcal{I}_u$ denote the entities recommended by a recommender system to user $u \in \mathcal{U}$. Following the definition of [48], we define the aggregate diversity metric DIV@K as

$$\text{DIV@}K = \frac{|\bigcup_{u \in \mathcal{U}} \mathcal{I}_u|}{|\mathcal{I}|} \tag{28}$$

where $|\mathcal{I}_u| = K$. Intuitively, a naive recommender such as a popularity-based one will always provide the same ranking of REs and therefore score lowly, while a random recommender will score highly. As such, the diversity metric can be interpreted as a level of personalisation provided by each model. Some works refer to this measure as catalogue coverage, since it reflects how many REs were covered in the top-K lists [28].

Separation task metrics. For the separation task, we consider Kendall's Tau [49] to measure the agreement between the predicted and true rankings of liked, unknown, and disliked entities. While separating unknown and disliked entities may be of little importance in making recommendations, it is still interesting to see how well models can separate entities based on explicit user preferences. Given a predicted ranking A and the true ranking B, we define the Kendall's Tau metric τ as

$$\tau = \frac{\text{Num. concordant pairs} - \text{Num. discordant pairs}}{\text{Num. concordant pairs} + \text{Num. discordant pairs}} \tag{29}$$

such that a pair of observations (a_i, b_i), (a_j, b_j) where we have that $i > j$ and $a \in A$ and $b \in B$ are *concordant* if $a_i > a_j$ and $b_i > b_j$, otherwise they are *discordant*. For every user, we rank one liked, unknown, and disliked entity for the user, where the ground truth ranking B is defined by our definition of preference (see Definition 2.2). A recommender will reach the maximum τ score of 1 if it always ranks liked entities over unknown entities, and unknown entities over disliked entities.

6.4 Negative sampling

Many works have observed that popularity based recommenders perform similarly to SOTA methods on the recommendation task [4, 10, 50, 51]. In general, the relative performance between models is drastically affected by the rating distribution over entities. In [51], they further find that the number of positive ratings on an entity in the test set is proportionate to the number of ratings on that entity in the training set. Consequently, if a dataset is biased towards popularity, a model that satisfies the majority of the user population would perform well using metrics that average over all users' satisfaction. Such metrics might contradict the personalisation while still describing a model's ability to satisfy the majority of the user population, but may fail for the minorities. In [10], they attempted to alleviate this bias by removing the most popular entities from the test set (i.e., testing on the long-tail), but this only reduces the problem [51] as the proportional relationship remains. As an alternative, [51] proposes two methods for sampling data: percentile-based partition and uniform test entity profile. The former method partitions entities into m bins according to their popularity such that the entities in each bin have similar popularity. Consequently, the number of test entities that can be selected for a user per bin is limited to $|\mathcal{I}|/m$, which for a sparse dataset can have great effect as the long-tail percentiles have few users. The latter method uses a subset of entities that can be uniformly sampled from, i.e., each entity is sampled an equal amount of times. As a result, popular entities are not sampled more frequently, yet the method limits the possible test ratings to a fraction of the dataset. To alleviate these issues, we propose using two sampling strategies for sampling negative samples in LOO evaluation: random sampling and equal-popularity sampling.

Random sampling. As described in [51], the number of randomly sampled positive ratings for an entity correlates with the number of ratings for that entity in the training set. As such, the distribution of positive sampled entities for LOO evaluation will approximate the training ratings distribution while the negative samples create a uniform distribution over all entities. The negative samples ratings distribution and that of the training set are therefore dissimilar. Following this sampling strategy, we can determine model performance on the majority of the users.

Equal-popularity sampling. To reduce the popularity bias of positive samples, we propose an equal-popularity sampling strategy which does not limit the number of entities that can be sampled. Our method performs sampling of negative samples with popularity similar to that of the sampled positive entities $\mathcal{E}_{pos}$. Using this sampling approach, the performance of popularity based recommenders is drastically reduced. We define the number of ratings in the training set for an entity e as $R(e)$. We then compute the weight $w(i)$ of a negative sample i given the positive samples as

$$w(i) = \left(\left| R(i) - \frac{1}{|\mathcal{E}_{pos}|} \sum_{e \in \mathcal{E}_{pos}} R(e) \right| + 1 \right)^{-\alpha} \tag{30}$$

where $\alpha \geq 0$ is a parameter that describes how frequently entities with dissimilar rating count should be sampled. Higher values of α gives lower weights to negative samples with greater difference in ratings to the positive samples. If the popularity bias is completely removed, popularity based recommenders will have worse or similar performance to a random recommender [51]. As such, we increase α until our popularity-based recommender reaches random performance. For the MR-170K dataset, we find that $\alpha = 10$ fits this criterion, but the value will depend on the distribution of the dataset.

Effect of sampling. To understand the effect of sampling we plot the cumulative distribution of ratings in Figure 9. The random sampling strategy is a straight line, indicating a uniform sampling of entities where each entity is sampled an equal number of times. As α increases using the equal-popularity method, the ratings distribution of the negative samples becomes increasingly popularity biased and thereby closer to the training sample curve. However, as there are few very popular entities due to the long-tail distribution, it is impossible to sample 100 very popular negative samples, hence the equal-popularity sampling strategy never becomes as steep as the training samples.

6.5 Experiments

As we have established in the problem definition and research questions, the primary objective of this work is to evaluate the performance of cold-start interviews where the user is queried about DEs rather than REs. Besides the main experiment, we show the results of some additional experiments in Appendix C, such as the effect of limiting interviews to specific entity types.

We conduct a single experiment including all considered models where we restrict the interview questions to that of DEs and REs separately. Both settings use the same training and validation data, and differ only in what the interviewer is allowed to ask about. We measure the performance when ranking positive and negative samples sampled with both random and equal-popularity sampling, as well as the models' ability to correctly rank a positive, neutral, and negative sample as per the separation task. We measure model performances in the ranking and separation tasks when conducting interviews of lengths $m \in \{1, 2, \ldots, 10\}$.

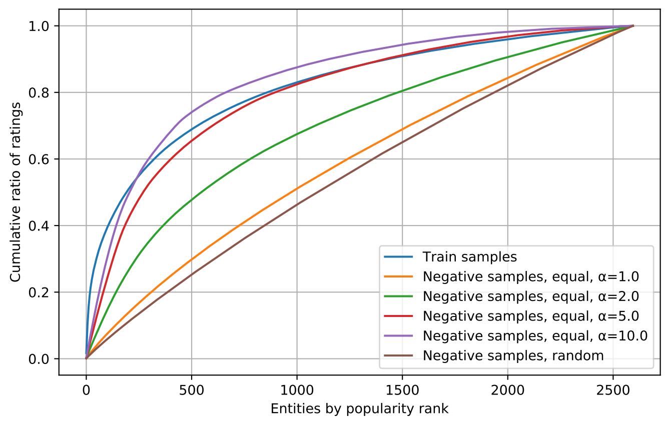

Figure 9: Cumulative ratio of total number of ratings from different negative sampling approaches compared to the training sample distribution. Entities are ordered by popularity rank in descending order.

6.6 Model selection

Since this work is the first to consider the user of DEs in cold-start interviews, we diversify both interviewing strategies and recommenders. Specifically, we consider the following interviewers for m-question interviews.

- **FixedPop (FP)**, selects the m most popular entities as questions.
- **FixedGreedy (FG)**, selects the m most informative entities by Equation 1.
- **AdaptiveGreedy (AG)**, constructs an m-height decision tree by Algorithm 1.
- **DQN**, a Deep Q-Network (DQN) that generates discrete actions from the action space (see Algorithm 2).
- **DDPG**, a DQN trained with Deep Deterministic Policy Gradient (DDPG) that generates continuously valued action vectors (see subsubsection 4.2.4).

Furthermore, we consider the following recommenders to be arbitrarily combined with the previously mentioned interviewers.

- **MF**, a Matrix Factorisation (MF) recommender which represents a user by the average embedding of their liked entity embeddings (see subsubsection 4.3.1).
- **KNN**, a user k-NN model which finds similar users based on the user vector representation.
- **PPR-KG**, a PPR model navigating the MindReader KG.
- **PPR-COLLAB**, a PPR model navigating a collaborative graph where each user has edges to their liked entities.
- **PPR-JOINT**, a PPR model navigating a combination of the MindReader KG and collaborative graph.
- **PPR-LINEAR**, a linear combination of PPR on different graph variants (see subsubsection 4.3.2).

Finally, we evaluate and compare against the following SOTA interviewing recommenders.

- **fMF**, a decision tree, MF-based interviewing recommender (see subsubsection 4.2.1).
- **LRMF**, a decision tree, RBMF-based interviewing recommender (see subsubsection 4.2.2).
- **MeLUN**, a deep meta-learning interviewing recommender (see subsubsection 4.1.3).

For all interviewers except MeLUN and FP, questions are selected from a candidate set of the top-100 most popular entities, following the approach of existing works [5, 27]. Furthermore, all models are validated on a left-out RE using NDCG@10.

6.7 Results

We now present the results of our experiments. Due to the large number of models and highly multi-faceted nature of our results, we focus on a subset of these which should aptly answer the research questions posed in section 1. Nonetheless, all results concerning this work are publicly available online[6].

6.7.1 *Impact of interviewing strategies*

In figures 10-13 we plot the differences in model performance for all interviewers and all recommenders on 5-length interviews. We observe that the same interviewers vary greatly in performance across different recommenders, which is indeed why we have diversified the selection of recommenders. As for differences between interviewers, the most notable differences are those between the FG and AG interviewers. For all PPR-based recommenders, the FG interviewer leads to the highest NDCG, while AG leads to higher diversity in the recommendations.

Furthermore, with the exception of k-NN and MF, FG outperforms FP in terms of NDCG. This is largely to be expected, as popularity is a less reliant metric for informativeness while recommender performance is more robust in this regard.

We observe that in the experiments using random sampling, the ranking quality produced by different interviewers on the same recommender is similar across interviewers, as opposed to experiments conducted with equal-popularity sampling. This is seemingly caused by the ranking quality being heavily related to popularity, hence why PPR-KG performs much worse than other models. However, on random sampling the interviewer does seem to have an effect on the diversity of recommendations.

Finally, we observe that DQN and DDPG performs badly regardless of the underlying recommender. Note that we have only included the RL interviewers with the k-NN, PPR-JOINT and MF recommenders due to time constraints.

6.7.2 *Performance of RE and DE interviews*

We present the performance results of all AG-based models (similar tendencies are observed for FP and FG) where the interview questions have been restricted to REs and DEs, respectively, in figures 14 through 17. In all figures, stars indicate statistical significance between RE and DE performance ($\alpha = 0.05$). As before, we limit the scope

[6] `https://mindreader.tech/dataset/results`

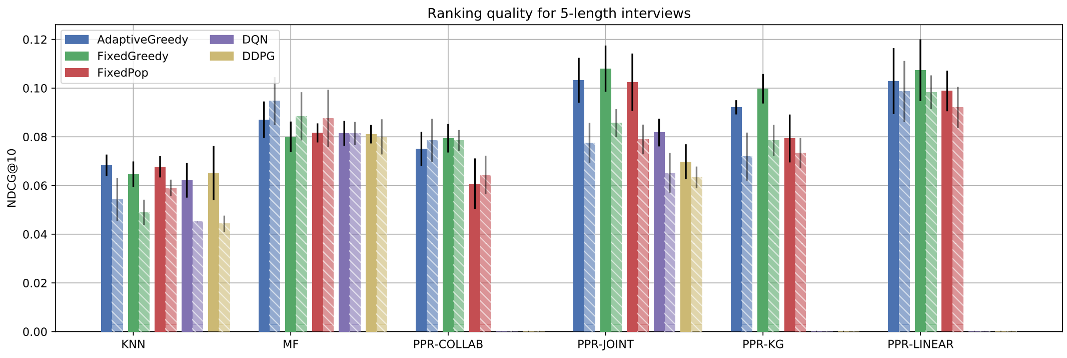

Figure 10: Ranking quality (NDCG@10) of interviewers with DEs- and REs-based interviews on equal-popularity sampling.

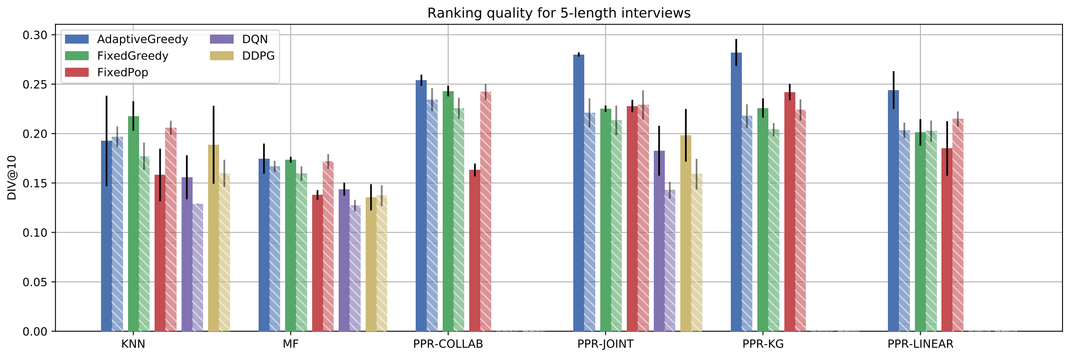

Figure 11: Recommendation diversity of interviewers with DEs- and REs-based interviews on equal-popularity sampling.

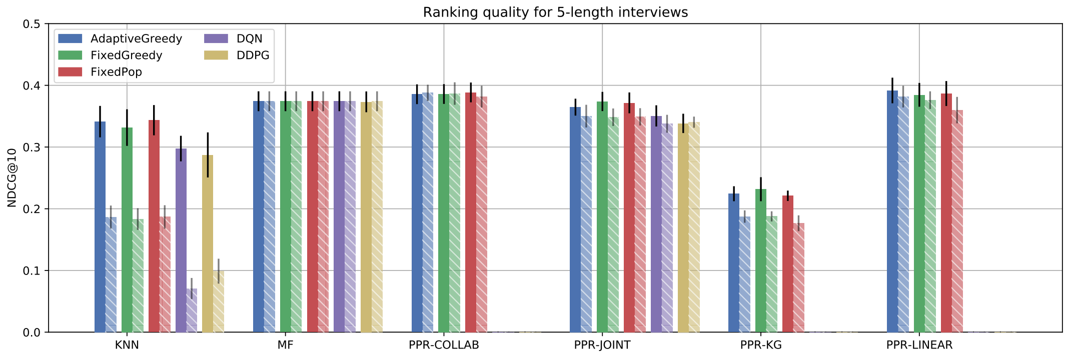

Figure 12: Ranking quality (NDCG@10) of interviewers with DEs- and REs-based interviews on random sampling.

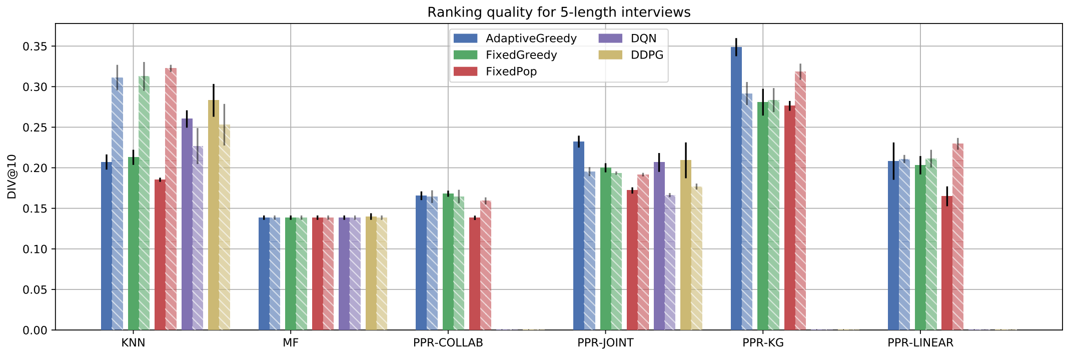

Figure 13: Recommendation diversity of interviewers with DEs- and REs-based interviews on random sampling.

Performance of the discussed interviewing strategies. Solid and striped bars denote performance in DE and RE interviews, respectively.

to 5-length interviews while interviews of different lengths are covered in the following section. This specific length is chosen as existing works focus on interview lengths in this range [5, 21, 22, 27].

In Figure 14 and Figure 15 we compare DE and RE interviews on equal-popularity sampling as to assert the informational gain afforded by allowing interviewers to ask towards DEs rather than REs. We observe that an AG interviewer using PPR-COLLAB, PPR-KG, PPR-JOINT, and k-NN recommenders is able to increase the ranking quality with statistical significance when asking towards DEs. Generally, the KG-based recommenders seem to make good use of the information provided by DE ratings, which follows observations made in our previous work where we saw similar results in the warm-start setting when replacing RE ratings with DE ratings [4]. The exception here is the k-NN recommender which does not model the KG, yet still sees a significant improvement in ranking quality, however this is also in line with our prior observations in the warm-start setting.

Under equal-popularity sampling, nearly all models are able to obtain higher recommendation diversity when provided with DE interview answers, with MeLUN and fMF being the only exceptions. Differing from our prior observations, MF is the only recommender that performs best on RE-based interviews, though we cannot show any statistical significance for this difference. In the same vein, the other MF-based model LRMF, performs better with DE ratings than RE ratings, though again this difference is not statistically significant.

Conversely, under random sampling, PPR-COLLAB, PPR-JOINT, PPR-KG, and fMF increase in diversity with DEs, while k-NN and PPR-LINEAR increase with REs, and MF, LRMF, and MeLUN achieve similar results with both. Additionally, k-NN scores very highly in diversity for RE interviews, which is a product of having few co-raters which we will cover further in the discussion.

Finally, we observe that most of our proposed recommenders outperform the two baselines MeLUN and fMF in terms of NDCG, and that MF and all collaborative PPR-based recommenders are able to outperform LRMF when provided with DE ratings.

In summary, an AG interviewer using the PPR-LINEAR recommender outperforms the best performing SOTA model, LRMF, in terms of NDCG and DIV when interviewing with RE and DE questions under equal-popularity sampling. Under random sampling, the two models are tied in NDCG, though PPR-LINEAR achieves significantly higher diversity.

6.7.3 *Improvement over longer interviews*

In Figure 18 and Figure 19 we show the performance of the baselines and the AG interviewer on all recommender models over increasing interview lengths $m \in \{1, 2, \dots 10\}$. Furthermore, we show the results on longer interview lengths in the appendix (subsection C.2).

We observe that nearly all models are able to achieve consistently higher NDCG when interviewing on DEs rather than REs with MF, fMF, and MeLUN as the only exceptions under equal-popularity sampling. We further observe that the greatest differences in performance are achieved by the PPR-JOINT and PPR-KG models, which again is in line with our prior observations in the warm-start setting that knowledge-aware recommenders make good use of the information afforded by DEs [4].

Although PPR-LINEAR performs the best in terms of NDCG, PPR-JOINT achieves similar ranking quality with much higher diversity. Generally, the graph-based recommenders reach the highest diversity and improve the most with increasing interview lengths. While all graph-based recommenders achieve better performance with DEs, PPR-COLLAB reaches its best NDCG at 10-length interviews on RE questions.

MeLUN remains static in NDCG and DIV with increasing interview lengths. We expand further in why this is in the discussion, but also note that the authors of MeLU show similar results with little or no improvement over longer interviews.

We observe that LRMF is among the best performing models in terms of NDCG under random sampling, especially for short interview lengths, but is among the least diverse models. Contrarily, PPR-LINEAR obtains similar ranking quality to LRMF, but is able to increase diversity significantly with longer interviews. Contrary to equal-popularity sampling, the embedding-based models including MF, LRMF, and MeLUN generally perform well in this setting.

Finally, while mostly consistent under equal-popularity sampling, we see a consistent decrease in diversity for the k-NN model under random sampling when interviewing with DEs.

6.7.4 *Serendipity of recommendations*

As argued in subsection 6.3, an important quality of recommender systems is the ability to generate “non-obvious” recommendations, as measured by how serendipitous (surprising and useful) the recommendations are. To measure this ability, we compute the SER@10 score for all recommenders in the top-K recommendation task. We present the results in Figure 20 for equal-popularity sampling. The results for random sampling are not shown as the change in performance in this setting simply corresponds to HR decreased by a constant factor.

We observe that the knowledge-aware models, specifically PPR-JOINT, PPR-KG, and PPR-LINEAR, perform well in generating serendipitous recommendations, with PPR-LINEAR being the best performing model on all interview lengths. We further observe that the embedding-based models MF and fMF also perform well in this task, though LRMF ranks below both at shorter interview lengths and increases with longer interviews, while MF and fMF remain relatively static. Finally, PPR-COLLAB performs the worst in this setting, but consistently improves as the interviews increase in length.

Generally, we observe that the graph- and neighbourhood based models are able to generate more serendipitous recommendations with DEs compared to REs, with the largest differences in performance being observed for the knowledge-aware models. Finally, all models perform better than a non-personalised, popularity-based recommender (denoted TopPop), which is in line with our expectations.

6.7.5 *Separation task*

We present the performance results of all AG-based models and baselines on the separation task in Figure 21. Generally, we observe that all models perform similarly or better in the separation task when conducting DE interviews as

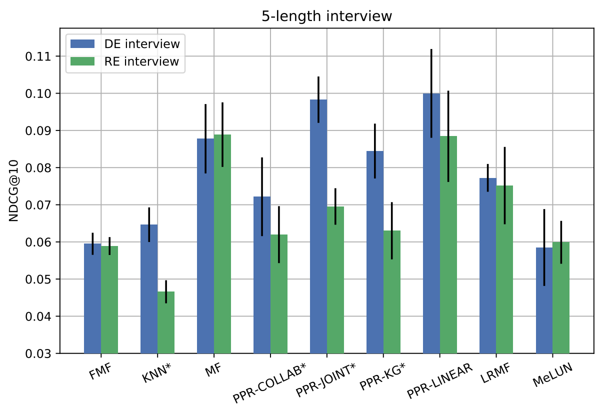

Figure 14: NDCG between RE and DE interviews on equal-popularity sampling.

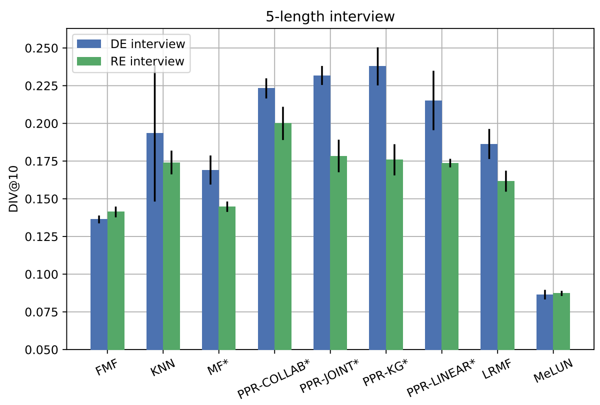

Figure 15: DIV between RE and DE interviews on equal-popularity sampling.

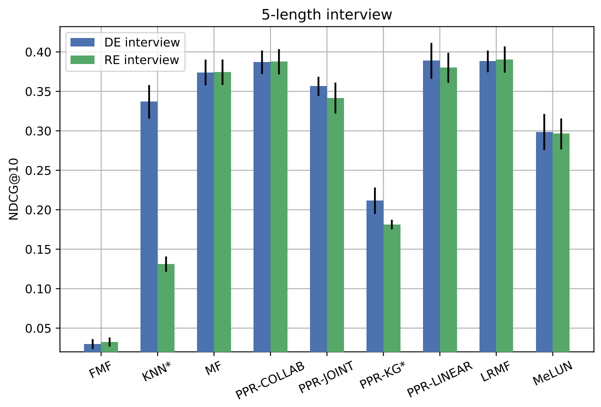

Figure 16: NDCG between RE and DE interviews on random sampling.

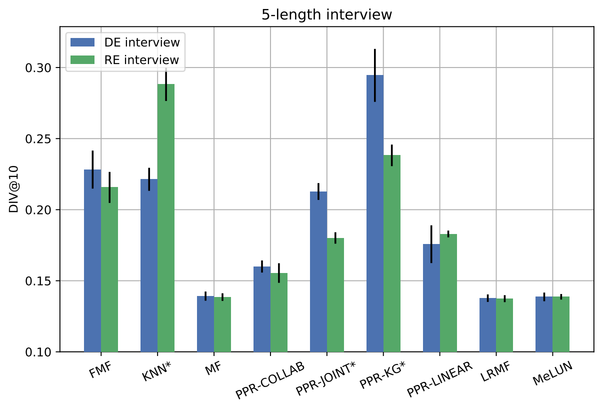

Figure 17: DIV between RE and DE interviews on random sampling.

compared to RE interviews.

We observe that the embedding models MF, MeLUN, LRMF and fMF outperform the graph-based models in the separation task, with MF being the top performer across all models. We further observe that, while initially poorly performing, k-NN sees a significant increase in performance as the interview length increases. Finally, we observe that the graph-based models perform comparably or worse than a non-personalised, popularity-based recommender on the separation task.

7. DISCUSSION

The results gathered provide substantial grounds for discussing the research questions posed and evaluating the quality of the models considered. In discussing our results, we firstly address each research question posed in the introduction, while we discuss more general observations and considerations in the remainder.

7.1 Impact of interviewing strategies

We now address our first research question:

"How can different question selection strategies affect the quality of cold-start recommendations following an interview?"

As described in subsubsection 6.7.1, we observe that for the PPR-based recommenders, the FG interviewers generally achieve a higher NDCG while the AG interviewers lead to more diverse recommendations. This observation is in line with that of [28], who find that diversity typically decreases as a function of accuracy. For the AG-based models, we posit that this has to do with the mechanisms of adapting to user groups by means of group splitting. While we will show later how such group splitting can quickly lead to overfitting for fMF, we argue that selecting a question that optimally increases the performance of an underlying recommender for some small user group is unlikely to be representative of most other users. In short, an interviewer is likely to overfit to the smaller user groups, which leads to lower NDCG. However, when measuring the difference between mean performances achieved by FG and AG, respectively, we find no statistically significant difference in NDCG for any of the interviewers over all recommenders in Figure 10. Furthermore, for the PPR-based recommenders, we find a statistically significant increase in diversity when using AG versus FG. This is because AG adapts the questions posed to a user depending on their answers, leading to more personalised and diverse post-interview recommendations. This is in spite of all model configurations being chosen as those with the best NDCG@10 score on the validation set.

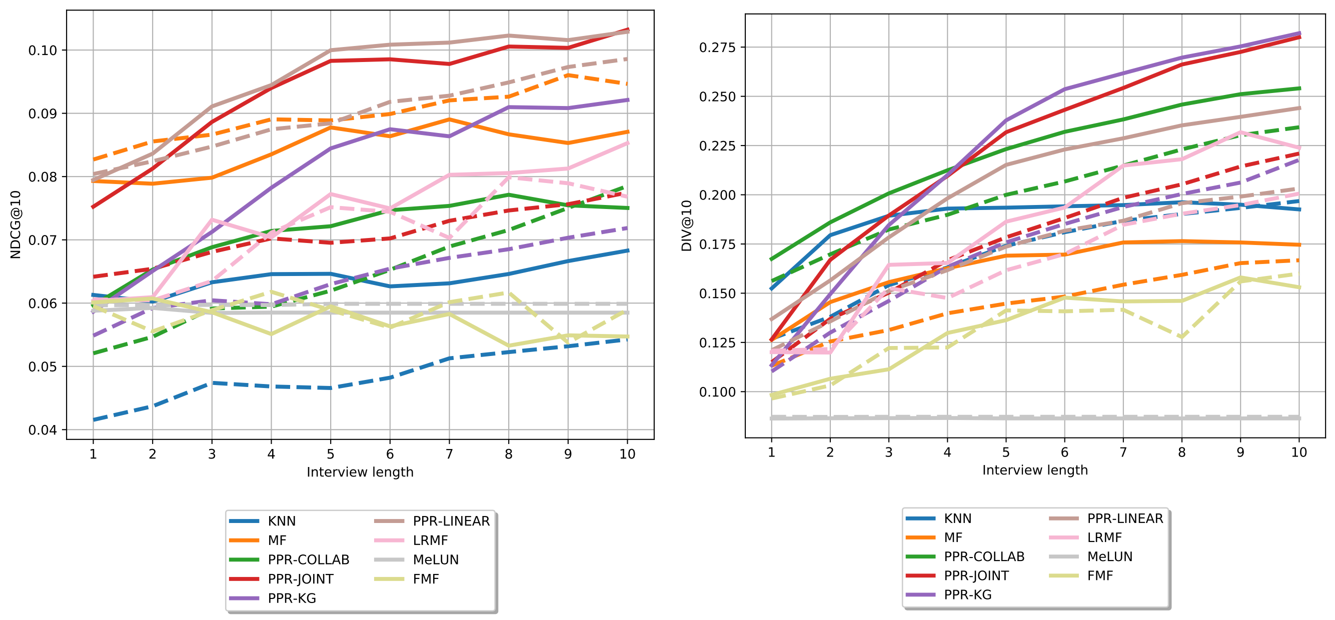

Figure 18: Recommendation quality and diversity from DE- and RE-based interviews (indicated by solid and dashed lines, respectively) of increasing number of questions under equal-popularity sampling. For the non-SOTA models, interviews are conducted using AG.

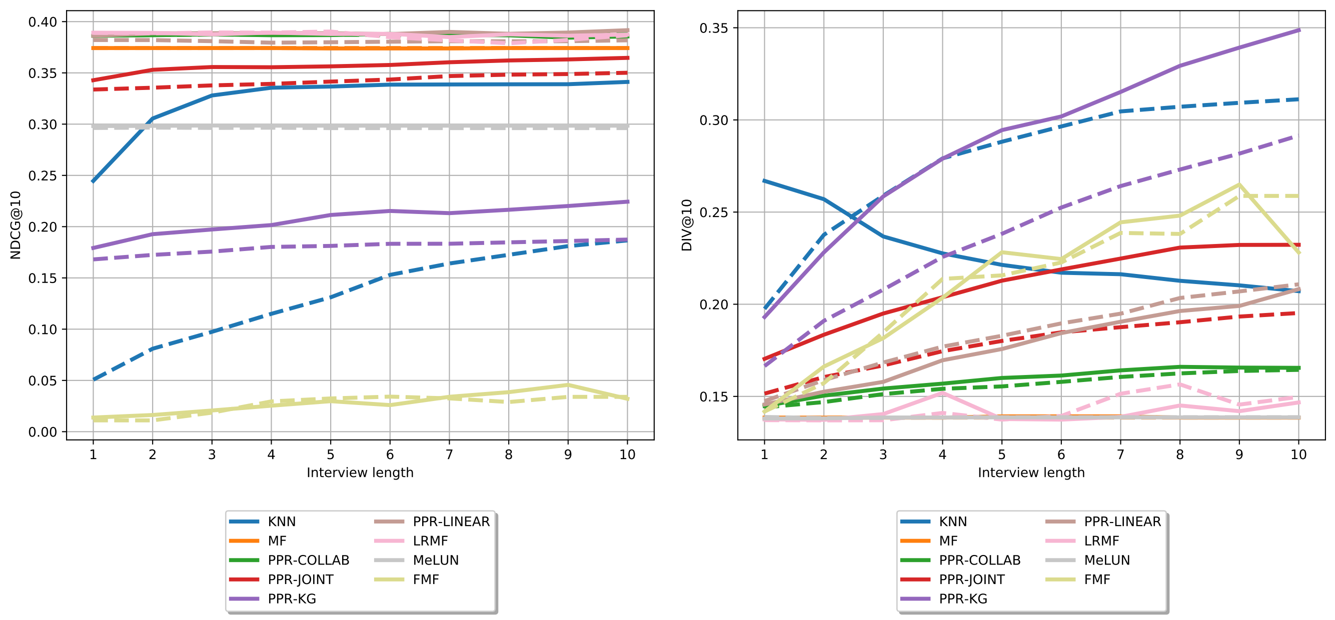

Figure 19: Recommendation quality and diversity from DE- and RE-based interviews (indicated by solid and dashed lines, respectively) of increasing number of questions under random sampling. For the non-SOTA models, interviews are conducted using AG.

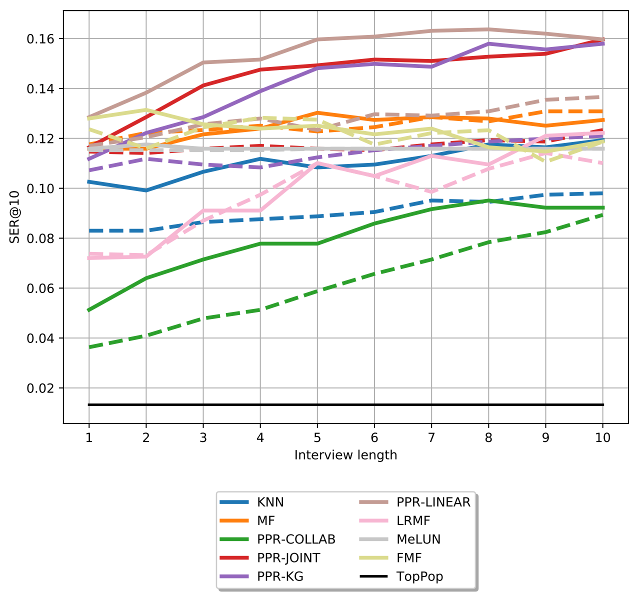

Figure 20: The SER@10 score for all models over DE- and RE-based interviews (indicated by solid and dashed lines, respectively) of varying lengths using equal-popularity sampling. The non-SOTA models conduct interviews with an AG interviewer.

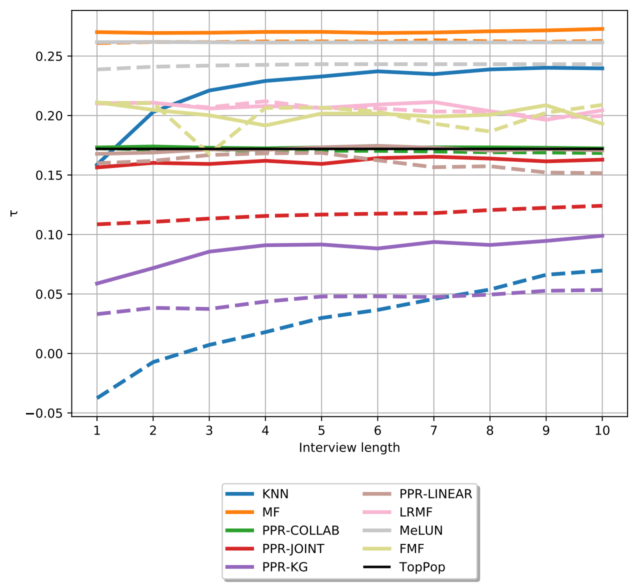

Figure 21: The models' ability to separate entities of varying sentiment from DE- and RE-based interviews (indicated by solid and dashed lines, respectively) of increasing lengths. For the non-SOTA models, interviews are conducted with the AG interviewer.

Following this, AG is the best performing interviewer when used with PPR-based recommenders as it is able to maintain NDCG performance similar to that of FG while significantly increasing the diversity of recommendations made.

Still focusing on the PPR-based recommenders, the exception to this general observation is our PPR-LINEAR recommender, which achieves the highest NDCG with an AG interviewer. Since PPR-LINEAR generates ranking scores from a linear combination of PPR rankings computed on different graphs, PPR-LINEAR is able to adjust the influence of graphs both containing and void of user ratings, allowing the recommender to overcome the problem of generalising over small user groups in the lower nodes of the interview decision tree.

For k-NN specifically, AG leads to higher NDCG while FG leads to higher diversity. While the difference is subtle and the standard deviation in diversity is large under equal-popularity sampling, the difference can be intuitively explained by how k-NN can achieve diversity. In k-NN, if we cannot determine any similar co-raters for a user-entity pair, the ranking score defaults to 0.0. The ordering between entities with the same score is arbitrary, and thus the k-NN recommender is able to diversify its recommendations if this occurs often.

The RL-based DQN interviewer performs badly with all recommenders. While the DQN action selection process could serve as an explanation, we see no improvement when using DDPG. The purpose of introducing DDPG was to alleviate the issues in selecting actions discretely from a large action space. However, we observe that the DDPG performance is on par with DQN except when using the PPR-JOINT recommender, in which case DDPG performs worse than DQN. While the RL interviewers should be able to determine the optimal interviews in theory, the RL training process is affected greatly by chance, namely the chance of exploring near-optimal interview traces in the initial exploratory phase where $\epsilon \sim 1$. We expand further on the instability of RL models and how we have attempted to overcome these issues in subsection 7.8.

In summary, our findings show that the AG interviewer leads to significantly higher diversity, while FG leads to higher NDCG without statistical significance. Besides these interviewers, FP is shown to be a strong baseline for question selection due to the well-known importance of asking towards popular entities. The effect of AG interviewers leading to more diverse interviews matches with our intuition, and is especially evident for DE-based interviews. Intuitively, the AG interviewer should offer more personalisation through adaptive question selection and asking broader questions leads to a larger set of candidate REs and thus higher diversity.

7.2 Performance in RE and DE interviews

We now address our second research question:

"How can descriptive entity ratings affect the quality of recommendations in interview-based cold-start recommender systems?"

As seen in Figure 14 and Figure 15, there is a clear improvement in performance when recommenders are provided with DE ratings from the interview rather than RE ratings under equal-popularity sampling. This is the case especially for the KG-based methods PPR-JOINT and PPR-KG. We saw similar results in our prior work, and we reason that the ability to make such effective use of DEs has to do with how DEs are situated in the KG. A rudimentary statistical analysis of the MindReader KG (see Table 4) shows that the DEs chosen as the most informative entities in the interview have a much higher degree than their RE-based interview counterparts. Being more central in the KG allows the DE-based interviews to provide more diversified recommendations while still being related to the user's preferences, including REs of high interest to the user that might not be ranked similarly were the personalisation restricted to RE nodes.

Question #	RE degree	DE degree
1	69	2,227
2	43	866
3	43	391
4	40	227
5	39	136
6	37	68
7	30	21
8	24	21
9	21	20
10	18	10

Table 4: Degrees of the interview questions (restricted to REs and DEs, respectively) in the MindReader KG as selected by the FG interviewer with a PPR-JOINT recommender.

The performance increases afforded by DEs are not as consistently present under the experiments conducted with random sampling (Figure 16 and Figure 17). We observe the same phenomenon when comparing across interviewers (Figures 10-13). We attribute the difference in performance under different sampling strategies to the popularity bias that equal-popularity sampling is designed specifically to alleviate. When the sampling is biased towards the positive samples as is the case with random sampling, most models can perform well by emulating a popularity-based recommender. Indeed, a non-personalised, popularity-based recommender performs best in this setting in terms of HR@10 despite disregarding the answers provided throughout the interview. However, in order to generate recommendations from the long-tail, which is a typically desired trait in recommender systems, the recommendations must be diversified from the most popular entities.

PPR-JOINT is unable to weigh the importance of the KG and collaborative parts of the graph individually. We observe that PPR-LINEAR, which is designed to address this limitation, performs the best under the AG interviewer in terms of NDCG, and almost achieves the same NDCG as MF when interviews are restricted to REs. While we observe no statistically significant increase in NDCG for PPR-LINEAR when comparing DE- and RE-based interviews, we observe a statistically significant increase in diversity.

While KGs appear as effective sources of auxiliary information in both DE- and RE-based interviews, we see that the RBMF-based LRMF model outperforms all other models in terms of NDCG under random sampling in DE-based interviews of lengths $1-5$. Interestingly, LRMF also outperforms the MF model in terms of diversity under the same setting, indicating that RBMF allows for embedding models to incur a better trade-off between accuracy and diversity in generating recommendations.

We observe that all interviewers perform nearly identically with MF as the recommender under random sampling. During experimentation, we observed that MF performed best with a single latent factor, indicating that not modelling user-specific features is an optimal strategy for this model.

Although we see no significant improvement in NDCG for any of the embedding models under DE-based interviews, this is not to say that such models are unfit for incorporating DE ratings. We also observe that LRMF increases in NDCG with DEs despite being an embedding-based model, though we attribute this to LRMF being based on RBMF rather than simple MF. In RBMF, users are represented as explicit rating vectors over a set of selected representative entities. In our prior work, we demonstrated how DEs have far more co-raters than REs, and thus the DEs chosen as the representative entities are more likely to have numerous observed ratings regardless of the user group. This makes the vectors less sparse and provides better support for accurate predictions.

7.3 Improvement over longer interviews

We now address our third research question:

"How can descriptive entity ratings affect the required interview length in generating recommendations of sufficient quality?"

As seen in Figure 18, models are able to improve their performance with increasing interview lengths. We also note that under random sampling (Figure 19), we do not observe a similar tendency to improve with longer interview lengths, as the model performances remain relatively stationary across all interview lengths. In order to assert how DEs can affect the interview lengths, we show in Table 5 the number of questions required by a RE-based interviewer to reach similar or better performance than its corresponding DE-based interviewer.

	Our models		SOTA models	
	Mean	Median	Mean	Median
Random NDCG	3.85	4.0	2.6	2.5
Random DIV	1.26	2.0	-1.16	-1.0
Equal NDCG	3.3	4.0	-1.9	-1.5
Equal DIV	3.68	4.0	-1.23	0.0

Table 5: Number of additional questions before an RE interviewer performs better than or equal to its DE counterpart.

For all interview length pairs $i, j \in \{1, 2, \ldots, 10\}$, we compare the metric of the DE interview at question i to the RE interview at question j, where $i \leq j$. Given this, we can determine the minimal number of additional questions to ask before the RE interviewer performs as well as or better than the DE interviewer. For instance, for PPR-COLLAB in Figure 18, we observe that the DE interviewer at question 1 is outperformed by the RE interviewer at question 4, thus the RE interviewer must ask 3 additional questions to achieve similar performance. In Table 5, a positive value indicates the minimal number of additional RE questions needed in order to reach DE performance. Conversely, negative values indicate the minimal number of additional DE questions required to reach RE performance. If the RE interview does not perform better at question 10, we assume it will perform better at question 11. This is, of course, a conservative estimate, but allows us to include models that never converge in performance between DE and RE interviews. We use a similar approach in the converse case where the DE interview does not perform better than RE at question 10.

As seen in Table 5, our models are able to perform better with DEs than REs, and that the interview can generally be shortened by ~ 4 questions. For the SOTA models, we observe that the interview can be shortened by ~ 2.5 questions under random sampling when aiming for NDCG, though in all other cases, RE interviews can be conducted at similar or shorter lengths than DE interviews.

One of the primary objectives of cold-start interviews is to minimise the user effort, as described in section 2. Being able to reduce the interview length with 4 questions while maintaining performance is therefore very desirable. Hence, finding that DE interviews are better by multiple questions in our models clearly demonstrates DEs allow effectiveness for eliciting user preferences in a cold-start interview, which is one of the primary hypotheses in our current and related work.

The effectiveness of DEs in interviews further supported by our observations in Table 4 where we see that, in a DE-based interview, the interviewer will start asking towards very broad entities that are highly connected in the KG, but also start asking towards much more specific and less connected entities sooner than the RE-based interviewer. Intuitively, from an interviewing standpoint, it does not make sense to ask specific questions without a clear understanding of a user's broader preferences. Since the interviewer can more quickly elicit a user's broader preferences by means of DEs, it can also shift to eliciting specific preferences sooner that a RE-based interviewer is realistically capable of since all REs represent largely specific preferences, and a larger number of questions is therefore required to elicit broader preferences. We posit that this is the reason why SOTA models are not capable of shortening the interviews as much as our knowledge-aware models, as they do not explicitly model the KG and thus have no access to auxiliary information regarding the connectivity of the entities.

Although DEs intuitively allow for broader preference elicitation, we observe that PPR-COLLAB reaches its highest NDCG@10 at 10 interview questions asking towards REs. We argue that this difference is due to the amount of information the model is able to elicit. When only a few questions are allowed, it makes sense to ask broad questions. However, when an interviewer is allowed to ask 10 questions, asking towards more specific preferences may be beneficial because a user's preferences on many REs will reflect both their broad and specific preferences.

For the k-NN model, we observe an abnormal decrease in diversity under random sampling in Figure 19. Under random sampling, negative samples are generally drawn from further down the long-tail as shown in subsection 6.4. With this, more negative samples are expected to have very few ratings and thus the k-NN model will not be able to find co-raters for some of these unpopular entities. Consequently, those negative samples will score lower than the entities with more co-raters, of which there are naturally fewer, which leads to less diverse ranking lists as the more popular entities will comprise the majority of the top-ranked entities.

Interestingly, the best performing model under random sampling in terms of NDCG is LRMF, though the same model also achieves among the worst diversity scores under the same setting. We attribute this phenomenon to LRMF effectively approximating a slightly personalised popularity-based recommender. While emulating a popularity-based recommender may seem to be an effective strategy in this setting, we also observe that PPR-LINEAR is indeed able to generate personalised recommendations, achieving similar NDCG to LRMF while increasing in diversity over longer interviews.

7.4 Impact of recommenders

We now address our final research question:

"How can the explicit modelling of a knowledge graph over the entities affect the quality of recommendations made?"

A clear disadvantage for most embedding-based recommenders is their inability to represent new users, which is a problem for both MeLU and MF. For MeLU, user metadata is utilised to overcome this problem, which is unavailable for the MindReader dataset. We therefore propose alternative representations of new users for both MeLU and MF, though the effectiveness of our proposed representations does not appear optimally expressive. Specifically, neither MF or MeLUN are able to outperform the other models in terms of diversity as shown in figures 14-17. In order to produce diverse and high-quality recommendations, the model must be able to represent a cold-start user and their observed interactions in a manner that allows the model to determine how to rank entities according to that user's preferences. If users are all represented equally, the model has no means of diversifying recommendations according to user preferences besides random exploration of entities. While the rankings generated by MF and MeLUN are not completely homogeneous, low diversity is an indication that the user embeddings cannot capture diverse preferences.

We also observe that the KG-based recommenders are able to reach higher values in both of these metrics as compared to the otherwise purely CF-based methods. For example, PPR-JOINT is able to score significantly higher than PPR-COLLAB in both NDCG and DIV. PPR-JOINT navigates a joint KG and collaborative graph in making recommendations. Thus, given a user's interview answers, the recommender is able to determine the importance of entities by virtue of their situation in the KG (i.e., not related to popularity) allowing the recommendations to diversify, as well as capturing the collaborative filtering effect for the same entities, allowing the recommendations to follow the common rating tendencies among users.

In subsection 3.2, we allude to the notion that incremental recommenders, where user-entity observations can be added incrementally, are able to more effectively represent cold-start users, and that the graph-based models express this quality. Indeed, the PPR-models clearly outperform both MF and MeLUN in DIV@10 for both sampling strategies. Furthermore, we observe that PPR-JOINT and PPR-LINEAR are the best performing models of the ones tested with relatively high DIV, NDCG, and SER.

While PPR-LINEAR can outperform PPR-JOINT in some cases, there is reason to discuss whether the increased complexity is justified by the observed increase in performance. We note, however, that PPR-JOINT is, in its nature, a static model of recommendations, and that shifting the paradigm from what graphs should be navigated to how graphs can be combined and merged with auxiliary inputs is a worthwhile paradigm to explore. Under any circumstance, our observations show that there is great room for improvement of PPR-LINEAR, and we encourage future research to take advantage of this.

In Figure 20, we also observe that the knowledge-aware models PPR-KG, PPR-JOINT, and PPR-LINEAR are especially effective in generating serendipitous recommendations, and display the largest difference in performance when comparing performance under DE- and RE-based interviews. We posit that PPR-JOINT, PPR-KG, and PPR-LINEAR are able to generate such recommendations better than the other models due to their ability to reason about the importance of entities as situated in the KG rather than simply popularity among users, as popular entities are not likely to lead to surprising recommendations. This effect is especially evident for PPR-KG, which consistently outperforms the MF model, which contrasts the results seen in Figure 18 and Figure 19 where MF clearly outperforms PPR-KG in terms of NDCG in interviews of lengths $1-5$. This is in addition to the PPR-based models' ability to accurately represent cold-start users by virtue of the models being incremental recommenders. Nevertheless, collaborative filtering remains an important property of the models in this setting, as we see that all models significantly outperform the non-personalised, popularity-based model in terms of serendipity in Figure 20. Furthermore, while PPR-KG performs well, it is outperformed by PPR-JOINT and PPR-LINEAR, both of which incorporate users' ratings in generating rankings.

We do, however, note that all PPR models presented do not fully utilise the information available in our KG as they do not take edge labels into account. As metadata can improve recommendation quality as shown in prior works [8, 21, 52, 53, 54], we motivate further research into using edge labels to further increase the model's ability to generate recommendations of both high ranking quality, diversity, and serendipity.

Furthermore, we note that diversity and serendipity are not universal indicators of high-quality, useful recommendations. For some systems, accurate recommendations from the short-head may be the main priority rather than diversifying along the long-tail or conducting cold-start user interviews as these incur the risk of losing the user's interest. In such cases, there is little need for incremental representation of users, and as seen in Figure 19, MF and LRMF can especially perform well in this setting with only one observed rating from the cold-start users.

The results presented here motivate the incorporation of KGs in generating recommendations of high quality, diversity, and serendipity in the user cold-start setting. While more research remains to be done in most effectively selecting questions that optimise all desired metrics specifically, we show that graph-based models capable of navigating both collaborative and knowledge-based graphs, as well as representing cold-start users incrementally, can aid in generating high quality, diverse, and serendipitous recommendations. However, we also note that these metrics of recommendation quality are not universal, and that an embedding-based model such as MF shows great promise in generating recommendations from the short-head and without the need for observations from cold-start users that may be difficult to obtain.

7.5 Impact of negative sampling strategies

In conducting our experiments, we have introduced two different strategies for negative sampling in order to better interpret model performances in tasks where popularity bias is an inherent problem. We observe an increase in diversity as the interview length increases for most models on both equal-popularity and random sampling, indicating improved personalisation, as seen in Figure 18. Specifically, for the best performing models in the experiments under random sampling, ranking quality is maintained while diversity increases. We observe that there is little difference in the performance of the best performing models, since these models learn to recommend the most popular REs. However, we expected these results given the popularity bias observed even in the warm-start setting of MindReader [4], which is reasonably expected to be exaggerated when we limit the number of ratings per user to the length of the conducted interviews. Yet, as the diversification of the best models is far higher than that of the non-personalised popularity-based recommender, we argue that the models are indeed capable of personalisation despite the apparent popularity bias. Similarly, we observe an increase in diversity for longer interviews using equal-popularity sampling, though the NDCG also increases steadily. We presume equal-popularity sampling to present a harder recommendation problem, as a set of equally-popular entities cannot be ranked trivially by popularity in generating recommendations. This presumption is substantiated by the greatly reduced NDCG performance of all models.

7.6 Separation performance

While the separation task is not the primary concern of the evaluation in this work, we conducted an additional study on the performance on a variety of models on that task. We observe that the embedding-based models outperform the graph-based models, which matches our intuition as embedding-based models are known to perform better on RMSE minimisation [10], which is more similar to the separation task than the top-K recommendation task. Specifically, we observe that fMF performs well on this task, whereas it was outperformed by much simpler methods on the recommendation task. This is also in compliance with our expectations, since fMF is designed to minimise a RMSE objective [5]. Unlike the neighbourhood- and graph-based models, the embedding-based models can learn how liked and well-known entities are.

We further observe that k-NN sees a significant increase in performance as the interview increases in length. This is due to the k-NN model not being able to determine an appropriately representative set of co-raters when the embedding vector of the interviewee is very sparse. However, as more rating entries are supplied to the embedding vector, the user embedding is moved towards users of greater similarity to their preferences, and the model thus obtains greater support for correctly ranking the entities.

We also note a potential correlation between modelling popularity and performance in the separation task. While performance on the separation task is not directly correlated to entity popularity (a non-personalised, popularity-based recommender is outperformed by embedding models on this task), we observe that the graph-based models are ordered according to how much influence the collaborative graph has on the final ranking, with PPR-COLLAB being ranked over PPR-JOINT, which is ranked over PPR-KG which does not take user ratings into account in ranking. As observed in our prior work, the MF objective is designed to update embeddings in favour of the most popular entities as the ratings for these comprise the majority of the MF loss [4]. A simple explanation for this is that less popular entities generally receive more “Don’t know” ratings than popular entities, and are thus ranked below the popular entities more often than not in the ground truth ranking. As such, ranking entities exclusively by their popularity can perform appropriately well in this task. Nevertheless, the embedding models display the ability to appropriately personalise the user- and entity embeddings in order to outperform this trivial ranking.

In optimising the training objective of PPR-LINEAR, the predicted ranking scores of liked and disliked entities are separated. However, during preliminary testing we found that PPR-LINEAR performs better when unknown and disliked entities are grouped together and separated equally from liked entities. Thus, the model does not learn how to separate unknown and disliked entities, leading to decreased performance on the separation task.

7.7 Comparison to state-of-the-art methods

The SOTA interviewing recommenders do not perform well when compared to combinations of simpler interviewers and recommenders. For fMF and LRMF it may be due to their training objective, though the lack of material for reproducibility may have an equal impact on performance. We also observe that MeLUN does not utilise the information gained from an interview. As the user embedding is merely the entities a user has rated, we posit that it simply learns to recommend purely based on the entity embeddings as the user embedding is too noisy. As a result the model is not highly personalised, but recommends based on the average user. A better representation of a user might help, but as of now, we deem that MeLU requires user metadata in order to perform well.

7.8 Instability of RL models

As can be observed from the results, both the DQN and DDPG interviewers fail to perform consistently across all recommenders. Although we have observed these interviewers occasionally outperforming their naive and greedy counterparts in prior experiments, the ranking quality achieved varies wildly between interview lengths, with typically no statistical significance nor continuous increase in information as the number of questions increase.

There are several reasons for this. First, the amount of exploration required in order to reliably discover optimal transition sequences increases exponentially with the interview length. Although a neural network is capable of inferring complex and non-linear dependencies between input neurons, the model cannot construct an optimal interview including questions (and answers) that have not been observed before. We attempt to alleviate this by decreasing the rate at with ϵ decays, but saw little change in performance.

Second, cold-start interviews generate sparsely rewarded transitions. This means that in most transitions describing a state-action pair, the reward is 0. Only in transitions where the interview finished can the model observe a reward > 0, as the reward is given as the ranking metric of a ranking generated by an arbitrary recommender. This can lead to a pitfall of RL systems, as the objective of both DQN and DDPG can be minimised trivially by predicting a reward of

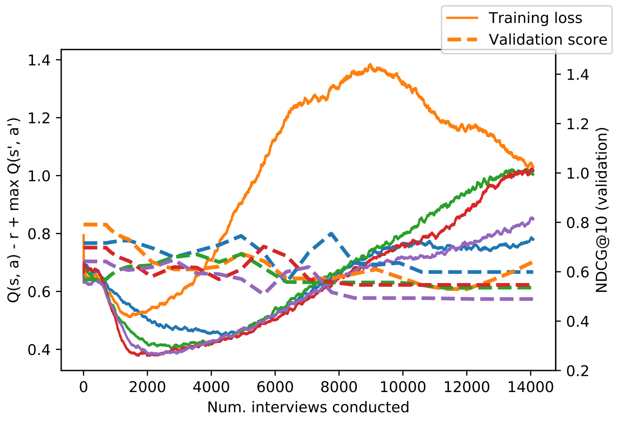

Figure 22: Training losses (left) and validation scores (right) for five different runs of the same DQN model on the same dataset conducting 3-length interviews.

0 if most training samples reflect this situation. This issue should be theoretically alleviated by the recursive nature of Equation 10 where the predicted reward of an action is the cumulative reward of all following, per-policy optimal actions. Yet, if no such near-optimal transition sequences are explored, there is no empirical evidence for the existence of such high-reward interviews, and the model optimises for a set of memories where the reward is always ~ 0.

To alleviate this issue, we conduct a weighted sampling of transitions from the replay memory $\mathcal{M}$, sampling transitions with a high reward (i.e., important memories) with a higher probability than those with reward 0. While we witnessed some increased stability in training, the performance remained highly varying between training sessions.

Finally, in standard supervised learning, the training samples are fixed and remain unchanging throughout the training session. Conversely, in RL, the targets we are training towards are defined by the very model currently being optimised. Due to this, training sessions including both DQN and DDPG incur the problem of descending the gradient of a constantly changing loss function. This introduces a large amount of instability in training, though we alleviate some by introducing both target networks and soft parameter updates [29, 30]. Nevertheless, when the targets are defined by the model itself, the overall trajectory of the training session is dependent mostly on memories generated in the initial exploration phase where $\epsilon \sim 1$. While we can extend the length of the exploration period by decreasing the decay rate of ϵ, the only way to guarantee consistent training is to exhaustively explore all possible transitions, which is an intractable task even for relatively short interviews. Another issue of chasing a moving target is that, depending on the minimum value of ϵ, it is possible for a model to discover new local optima during training, making it difficult to predict when to stop training.

We demonstrate this exact problem in Figure 22, where one of the five trained models starts decreasing in loss after $\sim$ 9000 interviews, long after the model has seemingly converged and started to overfit. This model, represented by the orange lines, ended with a NDCG@10 of $\sim$ 10.3% on interviews of length 3 on the test set under equal-popularity sampling. While this score is unparalleled by any other interviewer, this is a demonstration of both the potential power and significant instability of RL models in a reward-sparse setting such as cold-start interviews.

While we remain confident that RL can allow for better question selection in cold-start interviews, it is worth discussing whether the effort required for building and training these models is warranted given the performance of much simpler approaches.

7.9 Group sizes and overfitting

Several of our experiments did not meet expectations due to overfitting, particularly for the adaptive methods that split users into groups according to their interview answers. As determined in [4], having co-rated entities between users is fundamental for CF methods, and this issue is further exaggerated when users are partitioned not only by co-rated entities, but specifically entities that they have co-liked, co-disliked, etc. As shown in Figure 23, when users are split according to their interview answers, the user groups become too small for the collaborative filtering effect to be captured effectively even after a small number of questions.

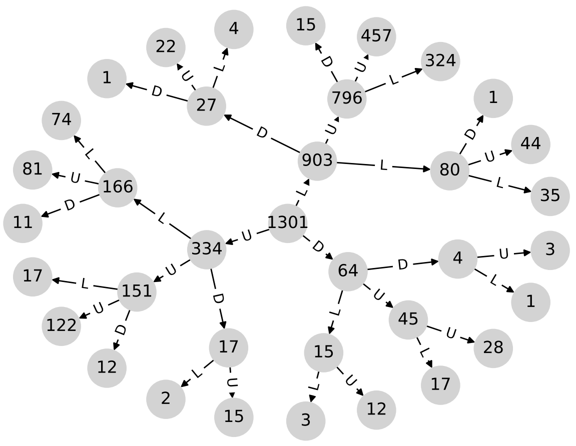

Figure 23: Group sizes when splitting on the most informative entities with a PPR-JOINT recommender (AG interviewer) in a 4-length interview. The decision tree branches by like (L), dislike (D), and unknown (U) ratings.

In adaptive models such as fMF or LRMF that optimise latent feature representations for individual user groups, this means that the feature representations are fit for a very small number of users that are not necessarily representative of the general preferences of all users who would belong to that group.

As seen in Figure 24, the fMF model is able to more accurately predict user ratings when splitting users into smaller groups according to their interview answers as long as the groups do not become too small. However, even after two questions, the groups become so small that the model loses the ability to effectively generalise over user preferences.

These issues may well be caused by the sparsity of our data, however while the experiments are not documented here, we observed the same issues with MovieLens-100K [24]. Another obvious possibility is that we have chosen ineffective hyperparameters, namely the number of latent features and regularisation parameters, for training fMF on our dataset.

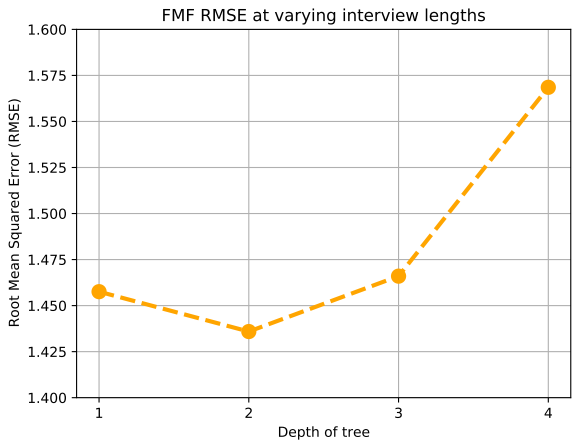

Figure 24: Rating prediction error of fMF at varying interview lengths.

However, due to the computational complexity of the fMF training algorithm, conducting a full grid search for all such hyperparameters is not practically viable. While fMF is able to perform more consistently on larger datasets such as MovieLens-1M [5], it is also worth mentioning that fMF is designed primarily for the rating prediction task on 5-star ratings, and not categorical ratings as are present in MindReader.

7.10 Statistical significance testing

For the purposes of our statistical significance testing, we follow the methodology of fMF [5] and employ a paired t-test over 4-fold cross-validation. While this approach is more powerful than the re-sampled t-test used to evaluate LRMF [27] since the testing sets between folds are disjoint, it has the issue that there are overlaps in the training sets between folds [55]. Consequently, the approach underestimates the variance that would be observed with independent training sets, resulting in a higher number of Type I errors (i.e., we are more likely to determine a statistical significance when there is none).

To reduce the number of type I errors, [55] proposes a 5x2 cross-validation approach, where 5 repetitions of 2-fold cross-validation are performed such that both the training and testing sets on a single repetition are non-overlapping. However, as is also concluded in the paper the choice of approach should depend on the running time of the algorithm evaluated. Empirically, we observed that training and testing a single fold of fMF took several days, which would make it impractical to evaluate 10 folds given the time constraints of this work. As such, we reckon the choice of 4-fold cross-validation is justified in our case, but given more time a better estimate of variance is warranted.

8. RELATED WORK

Our work is primarily concerned with addressing the user cold-start problem. This specific problem is burdened with a host of difficult-to-solve problems, e.g., inherent data sparsity, how to adapt to different cold-start users in making recommendations, and how to elicit a user's preferences effectively.

Preference elicitation. Multiple works have proposed solutions to the preference elicitation problem in the cold-start recommendation setting [5, 6, 12, 14, 21, 22, 23, 27]. In this setting, the typical approach is to conduct interviews, asking users questions in order to elicit their preferences.

At its core, an interview consists simply of a series of questions to ask the interviewee. The key challenge in these interviews is to determine the series of questions to ask. In a simple and early approach to cold-start interviews, [23] proposes to select a set of fixed questions that, when provided with cold-user feedback, best optimise a given recommender against a certain metric. While effective, the fixed question approach lacks adaptability throughout the interview. To address this issue, a popular approach is to represent the interview as a decision tree, guiding the user along branches of the tree according to their answers [5, 7, 22, 26, 27].

The decision tree representation can be a cumbersome data structure to deal with, especially when the question selection procedure is computationally expensive [5]. In [6], users and entities are represented as latent feature vectors, and the cold-start user embedding navigates the latent space according to their answers, selecting questions based on the user's affinity to other entities. While affinity can be an effective estimator of informativeness, [21] uses meta-learning to adapt to new users, using the network gradients as indicators of informativeness.

In addition to presenting a new approach for modelling a user's preferences during an interview, [6] further explores the effects of asking questions specifically considering absolute questions regarding the user's sentiment towards a specific entity and pairwise, relative questions asking for whether a user prefers one entity over another.

Use of auxiliary data. Few works have considered auxiliary data to assist in solving the preference elicitation problem. For example, the MeLU model, as proposed by [21], incorporates user metadata in order to improve personalisation. In another work, [12] implicitly determines user preferences w.r.t. genres from movie ratings and uses this information in question selection. Yet, incorporating auxiliary data to overcome the data sparsity problem is a common approach in recommender systems research [8, 37, 38, 53, 56]. As the aforementioned models are the only ones to make use of such auxiliary information, we find this approach to be surprisingly overlooked in addressing the preference elicitation problem. While user metadata is not always a readily available resource, KGs have been incorporated in good effect in producing more personalised recommendations [56] and to overcome data sparsity [8]. Specifically, [8] find that training a multi-tasking neural network to generate KG embeddings and user-item recommendations jointly is capable of generating useful recommendations with only a fraction of the original ratings. In another example, [53] models a user's preferences as ripples over an embedded KG, propagating the user's observed ratings across the graph embeddings in order to generate recommendations. In [37], recommendations are generated by a combination of KG node embeddings generated from random walks over sub-KGs restricted to certain edge types. Finally, [38] uses a non-uniform teleport distribution over a collaborative KG as in PPR-JOINT, weighing the REs liked by a user and the DEs connected to the liked REs higher than all other nodes.

Specific and broad questions. While the inclusion of auxiliary data has not been used in almost all of the aforementioned preference elicitation models, the interviews constructed have been designed specifically to select the questions that elicit the most useful information from the user. Yet, these works have focused on evaluation in contexts where REs can be posed as interview questions. Despite the apparent limitation to RE-based interviews in the literature, several works have suggested that users express their preferences naturally through broader, more descriptive entities [6, 12, 13]. In cold-start interview research, the prime objectives are to (i) conduct interviews that allow for useful recommendations, and (ii) shorten the required interview length as much as possible, thereby limiting user effort [5, 6, 27]. Keeping an interview brief and rich in information requires that users are familiar with and able to opine on questions posed during the interview. As found in [4], while both DE and RE ratings follow a short-head/long-tail distribution, users are generally far more likely to be familiar with certain types of DEs than REs, making DEs intuitively well-fit sources of information in cold-start interviews where we want to ask as few questions as possible.

Datasets. This focus on RE-based interviews is likely a consequence of lacking datasets supporting such interviews. However, while the two popular MovieLens [24] and Netflix Prize [57] datasets do not contain explicit user ratings on DEs, [58] built a dataset of explicit user ratings on movies and movie-specific tags. The tags are created by the users of the system through a process called social tagging. Because tags are not drawn from an existing, structured knowledge base, the quality of the tags and their semantic relations to entities varies greatly, and some low-quality tags such as "bad movie" carry their own sentiment [59].

Contrary to tag datasets, MindReader collects explicit ratings on DEs and REs drawn from an existing knowledge base, specifically a KG over entities in the movie domain. The entities in the MindReader KG carry no underlying sentiment, and their inter-relations are pre-defined by the KG rather than the users, ensuring higher quality of DEs. Furthermore, the MindReader KG not only models the fact that entities are connected, but how they are connected by means of relation labels such as Tom Hanks *starring in* Forrest Gump, which *has the genre* drama. This provides a solid foundation on which to select broader and more descriptive interview questions as well as means for extrapolating a user's interview answers to a more precise profile of the user's preferences.

Conversational recommender systems. Cold-start interviews have close ties to conversational recommender systems, as both approaches are instances of querying the user and using their input for the recommendations. The general difference in existing literature is that cold-start interviews focus more on what entities to ask about in a structured manner, whereas conversational recommender systems focus on how Natural Language Processing (NLP) can be used to elicit the preferences. In general, conversational recommender systems have been applied in e-commerce, i.e., by asking the user about technical specifications and price on the product they are seeking [60, 61]. E-commerce is a more extreme example of the efficacy of broader questions, since users are even less likely to opine on specific products as opposed to the movie domain.

9. CONCLUSION

In this work, we present a comprehensive analysis of different strategies and models for cold-start user interviews in the top-K recommendation task, addressing the informational value of explicit ratings on descriptive and recommendable entities. We also show how different interviewing systems can model our definition of user preference, separating entities of differing sentiments with respect to a given user. We investigate specifically the relationships between interviewing strategies and recommender paradigms in what, to the best of our knowledge, is the first such study conducted for cold-start interviews. We show how MindReader, the data collection platform responsible for the dataset used, can be further optimised for better recommendation quality and diversity while staying performant under high usage. Using the newest version of the MindReader dataset, we find that asking towards users' opinions of descriptive entities leads to better performance than asking towards recommendable entities, both in terms of ranking quality (i.e., NDCG) as well as the aggregate diversity and serendipity of recommendations. We also show how different sampling strategies in experimentation can help reduce popularity bias and allow for analysis of the relative performance differences between interviewing systems in different settings of the recommendation task.

Though not widely used in state-of-the-art models for cold-start recommendations, we show that iterative, graph-based models, especially when navigating a knowledge graph, can generate recommendations of higher quality than other widely used collaborative filtering models, and encourage that future research takes advantage of this in constructing state-of-the-art models for cold- and warm-start recommendations. We show that a linear combination of such models navigating different graphs can outperform all models considered in this work, as well as how an optimal combination can be inferred through simple gradient descent.

We also find that simple and generic methods for selecting interview questions independent of the underlying recommender system can outperform less generic and more complex state-of-the-art models for cold-start interviews. However, we note that these results may be subject to scrutiny, as we did not have access to the original implementation of two out of three models considered and modification was required for the model with accessible implementation, the differences in implementation may have affected the performance.

We also find that, while promising and suggested by prior works, our implementations of reinforcement-learning approaches to cold-start user interviews are largely unstable and require substantial computational resources for training, and that more research is required to make the learning processes more stable and reinstate better guarantees for useful convergence in the cold-start setting which is inherently sparse in terms of observable rewards.

While the informational value of descriptive entities in cold-start interviews is certain, our experiments are limited to a single dataset in a single domain of recommendation. We encourage researchers to consider the use of descriptive entities in other domains, both in cold-start interviews as well as warm-start recommendations. Furthermore, since recommender systems are closely tied to human-computer interaction, additional user-studies should be conducted in order to fully address the usefulness of descriptive entities.

10. FUTURE WORK

Our results provide grounds for a wide range of interesting future research within the application of DE ratings in recommender systems in general and approaches to generating cold-start interviews. We now cover some of these avenues of research with concrete examples of approaches to be considered.

10.1 Improved incorporation of KGs

Though we find PPR models to perform well and efficiently, they do not utilise all the available information of KGs. Specifically, the PPR models do not take the different relationship and entity labels into account. Furthermore, as a linear combination of PPR models seems to outperform existing PPR models, it might be interesting to further split the graphs into their relationship labels, as in [37], to utilise the relation information of the KG.

Moreover, as PPR scores clearly contain valuable information, it could be incorporated as auxiliary information about the cold-start user for other models. More specifically, for models where we are unable to represent new users, users could instead be represented by their PPR scores and translated through a learned translation mechanism, e.g., a transformation matrix as used in LRMF.

10.2 Improving RL approaches

While theoretically well-founded, our RL-based interviewers have failed to produce worthwhile results when compared to much simpler models with stronger guarantees for performance. Nevertheless, we remain firm in the position that RL is a promising avenue to research in conducting cold-start interviews, though RL specifically requires substantial work and additional computational resources before it becomes a practically applicable approach.

In subsection B.2, we cover preliminary testing of a Relational Graph-Convolutional Network (R-GCN) in order to support the DQN decision process with entity embeddings created from convolving user answers across a joint collaborative KG. While our preliminary tests did not show any significant improvements over the simpler DQN model, this can likely be attributed to the limited amount of testing and hyperparameter tuning we were able to conduct. In theory, graph-based embeddings should assist the model in reasoning between which entities to choose for the next questions, though more work needs to be done on the overall model and general approach to predicting rewards. We also motivate further research into ensuring stronger guarantees for useful convergence when training deep learning models for RL, where potential approaches to consider include model architectures such as duelling DQNs [62].

Finally, the observed tendency of deep NNs to overfit to a good average-case prediction and disregard inputs when they are sparse is another pitfall of deep learning methods in environments of sparse information that needs to be addressed in future research, as this is an inherent problem of the cold-start setting in general.

10.3 Optimising greedy question selection

One of the primary drawbacks of our proposed AG interviewer is that building the decision tree for representing the interview is computationally complex, and when the height of the tree increases, nodes with many users become the primary bottleneck.

When receiving a large number of users at a node in the decision tree, it may be the case that we do not have to consider all users in that group. Instead, it may be possible to efficiently determine a subset of the user group that is optimally representative of the rest.

10.4 Improvements to PPR-LINEAR

Currently, we learn a single weight per sentiment-graph combination. When we have no likes or dislikes to use as source nodes, we use a teleportation vector with a uniform weight distribution, i.e., global PageRank. In the training phase, this is not an issue since most users have both likes and dislikes, hence the global PageRank is rarely used. However, this intuitively leads to overfitting during the testing phase, since the input from users is very sparse and thus we are more likely to use global PageRank.

In an AG-based interview, we group dislikes and "don't know" answers together. As the disliked group is the smallest group and "don't know" is the largest, we worry that the negative ratings have little impact in the selection of the next question. With our PPR-LINEAR model, we are able to represent likes and dislikes in different graphs, and it may make sense to group likes and dislikes together instead and let the model itself use the ratings. Doing so would increase the number of users in each node, and thereby help in overfitting.

Acknowledgements

We would like to thank our supervisors Katja Hose, Matteo Lissandrini, and Peter Dolog for their valuable input and contributions to this thesis. Furthermore we would like to thank the users of MindReader for providing preference data.

References

[1] A. Bellogín and A. Said, "Recommender systems evaluation", in *Encyclopedia of Social Network Analysis and Mining*, R. Alhajj and J. Rokne, Eds. New York, NY: Springer New York, 2018, pp. 2095–2112, ISBN: 978-1-4939-7131-2. DOI: 10.1007/978-1-4939-7131-2_110162. [Online]. Available: https://doi.org/10.1007/978-1-4939-7131-2_110162.

[2] P. Pu, L. Chen, and R. Hu, "A user-centric evaluation framework for recommender systems", in *Proceedings of the fifth ACM conference on Recommender systems - RecSys '11*, ACM Press, 2011. DOI: 10.1145/2043932.2043962. [Online]. Available: https://doi.org/10.1145/2043932.2043962.

[3] C. C. Aggarwal, *Recommender systems: the textbook*. Springer, 2016.

[4] A. Brams, A. Jakobsen, and T. Jendal, "Evaluating the effects of non-item ratings inrecommender systems", 2019. [Online]. Available: https://projekter.aau.dk/projekter/files/320764987/MI911_new.pdf.

[5] K. Zhou, S.-H. Yang, and H. Zha, "Functional matrix factorizations for cold-start recommendation", in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information - SIGIR 11*, ACM Press, 2011. DOI: 10.1145/2009916.2009961. [Online]. Available: https://doi.org/10.1145/2009916.2009961.

[6] K. Christakopoulou, F. Radlinski, and K. Hofmann, "Towards conversational recommender systems", in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, San Francisco, California, USA: ACM, 2016, pp. 815–824, ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939746. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939746.

[7] M. Sun, F. Li, J. Lee, K. Zhou, G. Lebanon, and H. Zha, "Learning multiple-question decision trees for cold-start recommendation", in *Proceedings of the sixth ACM international conference on Web search and data mining - WSDM '13*, ACM Press, 2013. DOI: 10.1145/2433396.2433451. [Online]. Available: https://doi.org/10.1145/2433396.2433451.

[8] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, and M. Guo, "Multi-task feature learning for knowledge graph enhanced recommendation", *CoRR*, vol. abs/1901.08907, 2019. arXiv: 1901.08907. [Online]. Available: http://arxiv.org/abs/1901.08907.

[9] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, and T.-S. Chua, "Explainable reasoning over knowledge graphs for recommendation", Nov. 2018.

[10] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks", in *Proceedings of the Fourth ACM Conference on Recommender Systems*, ser. RecSys '10, Barcelona, Spain: ACM, 2010, pp. 39–46, ISBN: 978-1-60558-906-0. DOI: 10.1145/1864708.1864721. [Online]. Available: http://doi.acm.org.zorac.aub.aau.dk/10.1145/1864708.1864721.

[11] Y. Ren, T. Zhu, G. Li, and W. Zhou, "Top-n recommendations by learning user preference dynamics", in *Advances in Knowledge Discovery and Data Mining*, J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 390–401, ISBN: 978-3-642-37456-2.

[12] X. Zhang, J. Cheng, S. Qiu, G. Zhu, and H. Lu, "Dualds: A dual discriminative rating elicitation framework for cold start recommendation", *Knowledge-Based Systems*, vol. 73, pp. 161–172, 2015.

[13] F. Radlinski, K. Balog, B. Byrne, and K. Krishnamoorthi, "Coached conversational preference elicitation: A case study in understanding movie preferences", in *Proceedings of the Annual SIGdial Meeting on Discourse and Dialogue*, 2019.

[14] A. M. Rashid, G. Karypis, and J. Riedl, "Learning preferences of new users in recommender systems: An information theoretic approach", *Acm Sigkdd Explorations Newsletter*, vol. 10, no. 2, pp. 90–100, 2008.

[15] F. M. Harper, X. Li, Y. Chen, and J. A. Konstan, "An economic model of user rating in an online recommender system", in *International conference on user modeling*, Springer, 2005, pp. 307–316.

[16] K. Swearingen and R. Sinha, "Beyond algorithms: An hci perspective on recommender systems", in *ACM SIGIR 2001 workshop on recommender systems*, Citeseer, vol. 13, 2001, pp. 1–11.

[17] J. Gope and S. K. Jain, "A survey on solving cold start problem in recommender systems", in *2017 International Conference on Computing, Communication and Automation (ICCCA)*, IEEE, 2017, pp. 133–138.

[18] J. Lee, "Recommendation systems", in *Big Data and Computational Intelligence in Networking*, CRC Press, 2017, pp. 227–264.

[19] I. Fernández-Tobías, M. Braunhofer, M. Elahi, F. Ricci, and I. Cantador, "Alleviating the new user problem in collaborative filtering by exploiting personality information", *User Modeling and User-Adapted Interaction*, vol. 26, no. 2-3, pp. 221–255, 2016.

[20] W. Wang, H. Yin, Z. Huang, X. Sun, and N. Q. V. Hung, "Restricted boltzmann machine based active learning for sparse recommendation", in *International Conference on Database Systems for Advanced Applications*, Springer, 2018, pp. 100–115.

[21] H. Lee, J. Im, S. Jang, H. Cho, and S. Chung, "Melu: Meta-learned user preference estimator for cold-start recommendation", in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1073–1082.

[22] F. Hu and Y. Yu, "Interview process learning for top-n recommendation", in *Proceedings of the 7th ACM conference on Recommender systems*, 2013, pp. 331–334.

[23] N. Golbandi, Y. Koren, and R. Lempel, "On bootstrapping recommender systems", in *Proceedings of the 19th ACM international conference on Information and knowledge management*, 2010, pp. 1805–1808.

[24] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context", *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, Dec. 2015, ISSN: 2160-6455. DOI: 10.1145/2827872. [Online]. Available: https://doi.org/10.1145/2827872.

[25] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl, "Getting to know you: Learning new user preferences in recommender systems", in *Proceedings of the 7th international conference on Intelligent user interfaces*, 2002, pp. 127–134.

[26] N. Golbandi, Y. Koren, and R. Lempel, "Adaptive bootstrapping of recommender systems using decision trees", in *Proceedings of the fourth ACM international conference on Web search and data mining*, 2011, pp. 595–604.

[27] L. Shi, W. X. Zhao, and Y.-D. Shen, "Local representative-based matrix factorization for cold-start recommendation", *ACM Transactions on Information Systems (TOIS)*, vol. 36, no. 2, pp. 1–28, 2017.

[28] M. Ge, C. Delgado-Battenfeld, and D. Jannach, "Beyond accuracy: Evaluating recommender systems by coverage and serendipity", in *Proceedings of the Fourth ACM Conference on Recommender Systems*, ser. RecSys '10, Barcelona, Spain: Association for Computing Machinery, 2010, 257–260, ISBN: 9781605589060. DOI: 10.1145/1864708.1864761. [Online]. Available: https://doi.org/10.1145/1864708.1864761.

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning", *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[30] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning", *arXiv preprint arXiv:1509.02971*, 2015.

[31] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep reinforcement learning in large discrete action spaces", *arXiv preprint arXiv:1512.07679*, 2015.

[32] S. Funk, "Netflix update: Try this at home", 2006.

[33] X. Ning, C. Desrosiers, and G. Karypis, "A comprehensive survey of neighborhood-based recommendation methods", in *Recommender systems handbook*, Springer, 2015, pp. 37–76.

[34] D. Gallo, M. Lissandrini, and Y. Velegrakis, "Personalized page rank on knowledge graphs: Particle filtering is all you need!",

[35] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh, "Wtf: The who to follow service at twitter", in *Proceedings of the 22nd International Conference on World Wide Web*, ser. WWW '13, Rio de Janeiro, Brazil: Association for Computing Machinery, 2013, 505–514, ISBN: 9781450320351. DOI: 10.1145/2488388.2488433. [Online]. Available: https://doi.org/10.1145/2488388.2488433.

[36] S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, "How to find a good submatrix", in *Matrix Methods: Theory, Algorithms And Applications: Dedicated to the Memory of Gene Golub*, World Scientific, 2010, pp. 247–256.

[37] E. Palumbo, D. Monti, G. Rizzo, R. Troncy, and E. Baralis, "Entity2rec: Property-specific knowledge graph embeddings for item recommendation", *Expert Systems with Applications*, p. 113 235, 2020.

[38] C. Musto, G. Semeraro, M. de Gemmis, and P. Lops, "Tuning personalized pagerank for semantics-aware recommendations based on linked open data", in *European Semantic Web Conference*, Springer, 2017, pp. 169–183.

[39] T. H. Haveliwala, "Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search", *IEEE transactions on knowledge and data engineering*, vol. 15, no. 4, pp. 784–796, 2003.

[40] W. Chen, T.-Y. Liu, Y. Lan, Z.-M. Ma, and H. Li, "Ranking measures and loss functions in learning to rank", in *Advances in Neural Information Processing Systems*, 2009, pp. 315–323.

[41] S. Rendle and C. Freudenthaler, "Improving pairwise learning for item recommendation from implicit feedback", in *Proceedings of the 7th ACM international conference on Web search and data mining*, 2014, pp. 273–282.

[42] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[43] A. Taha, Y.-T. Chen, T. Misu, A. Shrivastava, and L. Davis, "Boosting standard classification architectures through a ranking regularizer", in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 758–766.

[44] X. He, T. Chen, M.-Y. Kan, and X. Chen, "Trirank: Review-aware explainable recommendation by modeling aspects", in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ser. CIKM '15, Melbourne, Australia: ACM, 2015, pp. 1661–1670, ISBN: 978-1-4503-3794-6. DOI: 10.1145/2806416.2806504. [Online]. Available: http://doi.acm.org/10.1145/2806416.2806504.

[45] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering", in *Proceedings of the 26th international conference on world wide web*, International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[46] L. Chen, Y. Yang, N. Wang, K. Yang, and Q. Yuan, "How serendipity improves user satisfaction with recommendations? a large-scale user evaluation", in *The World Wide Web Conference on - WWW '19*, ACM Press, 2019. DOI: 10.1145/3308558.3313469. [Online]. Available: https://doi.org/10.1145/3308558.3313469.

[47] T. Murakami, K. Mori, and R. Orihara, "Metrics for evaluating the serendipity of recommendation lists", in *New Frontiers in Artificial Intelligence*, K. Satoh, A. Inokuchi, K. Nagao, and T. Kawamura, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 40–46.

[48] G. Adomavicius and Y. Kwon, "Improving aggregate recommendation diversity using ranking-based techniques", *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 896–911, 2011.

[49] M. G. Kendall, "A new measure of rank correlation", *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[50] Z. Zolaktaf, R. Babanezhad, and R. Pottinger, "A generic top-n recommendation framework for trading-off accuracy, novelty, and coverage", in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, IEEE, 2018, pp. 149–160.

[51] A. Bellogín, P. Castells, and I. Cantador, "Statistical biases in information retrieval metrics for recommender systems", *Information Retrieval Journal*, vol. 20, no. 6, pp. 606–634, 2017.

[52] D. Liang, J. Altosaar, L. Charlin, and D. M. Blei, "Factorization meets the item embedding", in *Proceedings of the 10th ACM Conference on Recommender Systems - RecSys '16*, ACM Press, 2016. DOI: 10.1145/2959100.2959182. [Online]. Available: https://doi.org/10.1145/2959100.2959182.

[53] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo, "Ripplenet: Propagating user preferences on the knowledge graph for recommender systems", in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 417–426.

[54] E. Palumbo, G. Rizzo, R. Troncy, E. Baralis, M. Osella, and E. Ferro, "Translational models for item recommendation", in *ESWC'18*, 2018, pp. 478–490.

[55] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms", *Neural computation*, vol. 10, no. 7, pp. 1895–1923, 1998.

[56] Y. Cao, X. Wang, X. He, Z. Hu, and T.-S. Chua, "Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences", in *The World Wide Web Conference on - WWW*, ACM Press, 2019. DOI: 10.1145/3308558.3313705. [Online]. Available: https://doi.org/10.1145/3308558.3313705.

[57] J. Bennett, S. Lanning, *et al.*, "The netflix prize", in *Proceedings of KDD cup and workshop*, Citeseer, vol. 2007, 2007, p. 35.

[58] S. Sen, J. Vig, and J. Riedl, "Tagommenders: Connecting users to items through tags", in *WWW'09*, 2009, pp. 671–680.

[59] F. Gedikli and D. Jannach, "Improving recommendation accuracy based on item-specific tag preferences", *ACM Transactions on Intelligent Systems and Technology*, vol. 4, no. 1, pp. 1–19, Jan. 2013. DOI: 10.1145/2414425.2414436. [Online]. Available: https://doi.org/10.1145/2414425.2414436.

[60] Y. Zhang, X. Chen, Q. Ai, L. Yang, and W. B. Croft, "Towards conversational search and recommendation: System ask, user respond", in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 177–186.

[61] Y. Sun and Y. Zhang, "Conversational recommender system", in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 235–244.

[62] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning", *arXiv preprint arXiv:1511.06581*, 2015.

[63] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning", *arXiv preprint arXiv:1506.02142*, 2015.

[64] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate", *arXiv preprint arXiv:1409.0473*, 2014.

APPENDIX

A. EXPERIMENTAL FRAMEWORK

From working with SOTA models, we found that the reproducibility of these is rather limited in the sense that both fMF and LRMF do not provide code for their model implementation nor experimental setup [5, 27]. Given the issues we encountered we want to allow other researchers to easily reproduce our findings, including both model implementations and experimental setup. With our cold-start framework, researchers can easily access implementations of the SOTA and novel methods covered in this work, and arbitrarily combine recommenders with interviewers as we have done. The cold-start framework in its entirety is available on GitHub[7]. In the remainder of this section, we outline key features of the framework.

A.1 Experiment definitions

In this work, we have demonstrated how issues such as popularity bias can be alleviated by means of different sampling strategies. With a few different sampling strategies and experiment settings, managing and maintaining all experiments can become cumbersome. In our cold-start framework, we define experiment configurations by means of `ExperimentOptions` objects as shown in Listing 1.

```
equal_popularity = ExperimentOptions(
  seed=123,
  name='equal',
  cold_start_ratio=0.25,
  include_unknown=False,
  evaluation_samples=1
  count_filters=[
    CountFilter(
      lambda count: count >= 1,
      entity_type=EntityType.RECOMMENDABLE,
      sentiment=Sentiment.POSITIVE)
  ],
  ranking_options=RankingOptions(
    unseen_sampling=
        UnseenSampling.EQUAL_POPULARITY,
    num_positive=1,
    num_unseen=100
  )
)
```

Listing 1: Configuration of our experiment using equal-popularity sampling on one positive sample against 100 negative samples, requiring that all users have at least one positive rating on a recommendable entity.

Regardless of the experiment configuration, an experiment can be run where interview questions are restricted to REs by including the `recommendable` flag when running the experiment (see Listing 4).

A.2 Model definitions

As we have mentioned, interviewers can arbitrarily be combined with recommenders under our framework. Consider for example Listing 2 in which we define a model using the AG interviewer with PPR-JOINT as the underlying recommender. As is also shown in the example, model definitions can include custom arguments for both the interviewer and recommender. In this case, the `GreedyInterviewer` class supports both fixed and adaptive interviews, so we explicitly specify an adaptive approach should be used.

```
'adaptive-greedy-ppr-joint': {
  'interviewer': GreedyInterviewer,
  'recommender': JointPageRankRecommender,
  'interviewer_kwargs': {
    'adaptive': True
  }
}
```

Listing 2: Model definition of AG with PPR-JOINT.

The `GreedyInterviewer` and `JointPageRankRecommender` classes both must inherit from the abstract `InterviewerBase` and `RecommenderBase` classes, respectively, that define methods required for training and evaluation. The class signatures are shown in Listing 3.

```
class InterviewerBase:
    def warmup(training, interview_length)
    def interview(answers, max_qs)
    def predict(items, answers)
    def get_parameters()
    def load_parameters(params)

class RecommenderBase:
    def fit(training)
    def predict(items, answers)
    def clear_cache()
```

Listing 3: Class signature of the `InterviewerBase` and `RecommenderBase` abstract classes.

A.3 Distributed experiments

Running all experiments on a single machine becomes impractical and time-consuming with a high number of model combinations, splits, experiment settings, and models requiring substantial computing resources.

To distribute the experimentation effort while maintaining an overview of experimentation status, we enable automatic uploading of experiment results to a main server that aggregates the data automatically, presenting a statistical overview for each model and metric at any interview length.

This functionality is enabled with the `upload` flag when running an experiment as seen in Listing 4. The address to which results should be pushed is configured in the top-matter of the `interview.py` entrypoint, which must run the Spectate API found in our GitHub repositories. The Spectate API is a centralised, optional part of the framework to which results can be uploaded and retrieved amongst distributed workers.

```
python3.8 entrypoints/interview.py
          --experiments equal
          --recommendable
          --upload
```

Listing 4: Running the equal-popularity experiment with interview questions restricted to recommendable entities and automatic results uploading enbaled.

Using the Spectate API, the results are visualised as seen in Figure 25.

[7]`https://github.com/MI-911/cold-start-framework`

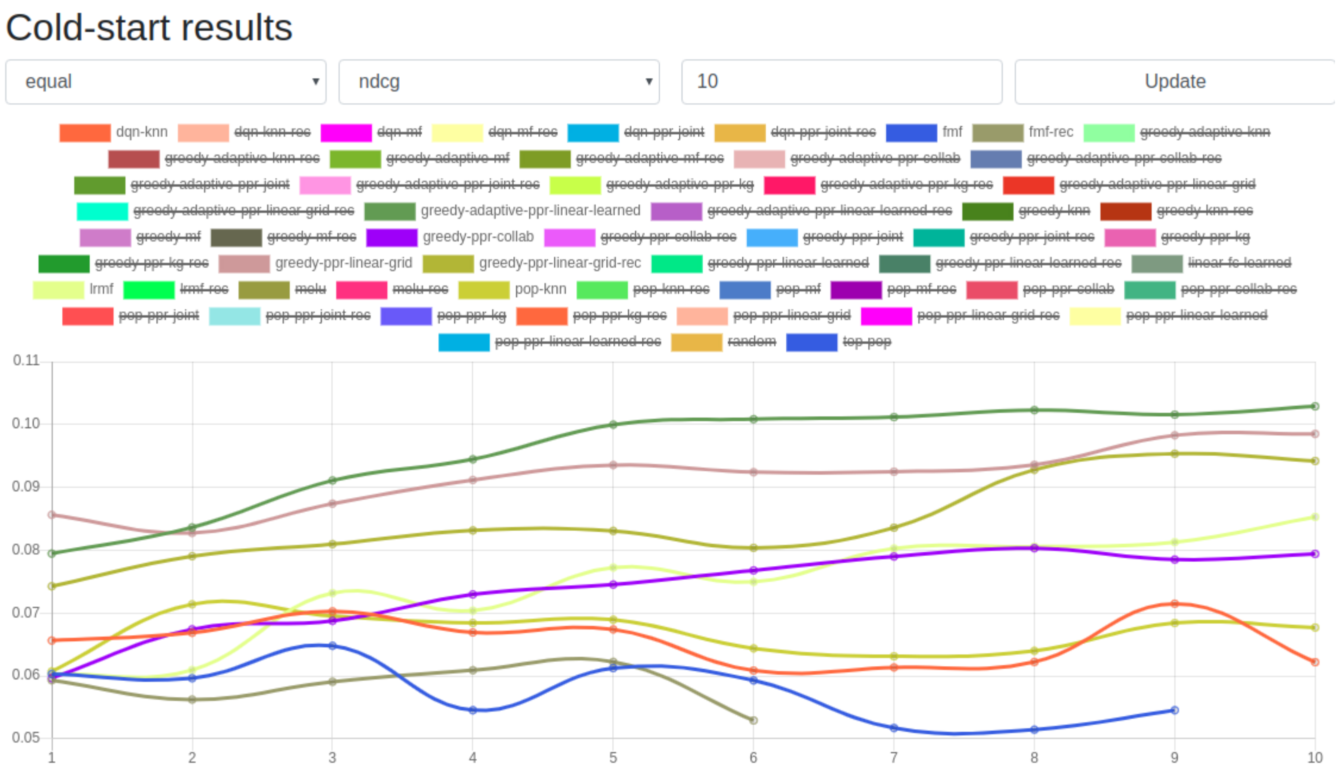

Figure 25: Results visualisation in the Spectate API.

B. ADDITIONAL MODELS

In addition to our proposed models for interviewing and recommendation, we have conducted preliminary testing of other models and approaches to question selection. In the following, we briefly cover those approaches, why we have tested them, and the observations we made.

B.1 Uncertainty-based models

In this work, we have primarily been focused on selecting questions based in their informativeness, namely, how much useful information will a question provide the model with in making better recommendations. However, informativeness can be difficult to quantify, as asking the question that allows an underlying recommender to perform the best for some group of users is not necessarily the most informative question, but rather the question that supports recommendations for that group the best.

Another way of thinking about informativeness is instead lack of information. Indeed, a natural approach to selecting interview questions is to select the questions that the interviewer is most unsure about. In deep learning, one way of representing uncertainty is to run the same input through the same model using random dropout in every forward propagation, measuring the prediction variance for every output neuron [63]. This approach of course requires the model to be a deep learning interviewing recommender model, as it both has to generate recommendations (in order to measure uncertainty in recommendations) and select questions based on that uncertainty.

We constructed a simple feed-forward neural network taking in an interview state representation vector as described in subsubsection 4.2.4, and produces a vector of ranking scores for all entities. In selecting questions, we run the same state vector through the network 100 times, applying 50% dropout on the hidden layers in the network at every forward pass. The question selected corresponds to the entity with the highest variance in predicted ranking score weighted by the popularity of the entity as in [21].

While we observed that the model was capable of choosing good questions that largely corresponded to those selected by the AG interviewer, we also observed that the model was not able to consistently improve as it was provided with more questions. This leads into a broader discussion of the limitations of deep learning models and sparse input vectors, as we see the same issue in both the MeLUN, DQN, and DDPG models.

The interview state vector $s \in \mathbb{R}^{|\mathcal{E}| \cdot 2}$ is a huge and very sparse vector, and neural networks are trained to best approximate the desired function over all inputs. When the input neurons almost always receive a 0, the network is likely to simply disregard the input, and update the weights and biases in the hidden layers in order to generate an optimal average prediction.

Related works have proposed attention mechanisms in order to let the network (or a completely separate network) learn what features should be paid attention to, and what features can be disregarded from the input vector [64]. However, this specific issue is less one of paying too much attention to the wrong features, and more that each feature must be assumed to be equally important, but only a very small number of features are actually observed.

B.2 Graph convolutional networks

While we have little information to append to the sparse interview-state vector, we can provide some additional information to the RL models by processing the input by means of the KG the entities are situated in. Graph-Convolutional Networks (GCNs) are specific types of NNs that allow a model to convolve inputs across neighbouring nodes in a graph, tying convolution weights to each node and relation in the graph. In order to take advantage of the full expressive power of KGs, we use an R-GCN to model a combined KG and collaborative graph as used in PPR-JOINT.

The goal of including this additional model is to allow a DQN to learn latent representations of the entities in the action space that are supported by the graph structure. The input is convolved across the graph by means of message passing, where the weights of a node are passed and aggregated in the neighbouring nodes. For R-GCN specifically, we use separate weights for different relation types. Each layer of the network propagates the features to neighbouring nodes, resulting in aggregations in the l-hop neighbours at the l^{th} layer of the network.

Formally, the representation of a node h_i at layer $l+1$ is given by the message passing formula:

$$\mathbf{h}_i^{l+1} = \sigma\Big(\sum_{r\in\mathcal{R}}\sum_{n\in\mathcal{N}_r}\mathbf{W}_r^{(l)}\mathbf{h}_n^{(l)}\Big) \tag{31}$$

where $r \in \mathcal{R}$ denote the relation types, $n \in \mathcal{N}_r$ denote the neighbours of h_i reachable through a single relation of type r, $\mathbf{W}_r^{(l)}$ are the relation-specific weights at layer l, and σ is a non-linear activation function.

We generate a distribution of Q values by taking the inner product between node entity embeddings and a user embedding, much like the procedure of MF. In order to construct a user embedding, we extract the embeddings for each of the entities the user has provided answers for from the interview state vector.

Let M denote the layers of the R-GCN, and let $\mathcal{E}'$ denote the entities the user has provided answers for in the interview. We first extract entity embeddings by passing the entities through the network, yielding the embeddings $\mathbf{E} = [M(e_i)$ for $e_i \in \mathcal{E}']$.

From $\mathbf{E}$, we construct a unified embedding for each rating category in $\{\text{Like}, \text{Dislike}, \text{Don't know}\}$ by summing embeddings within each category, resulting in the embeddings $\mathbf{e}_L$, $\mathbf{e}_D$, and $\mathbf{e}_U$, respectively. In order to capture the semantics of different ratings, we use a fully connected layer L to construct the final user embedding $\mathbf{u} = L([\mathbf{e}_L; \mathbf{e}_D; \mathbf{e}_U]) = \sigma(\mathbf{W}_L \cdot [\mathbf{e}_L; \mathbf{e}_D; \mathbf{e}_U] + \mathbf{b}_L)$ where $\mathbf{W}_L$ and $\mathbf{b}_L$ are the weights and biases in L.

The final Q values are then predicted as $Q(s, e) = \mathbf{u}\cdot M(e)$ where s is the interview state, e is an action (i.e., an entity to ask about), and $\mathbf{u}$ is given by our user embedding procedure $\mathbf{u} = L(M(s))$. The network parameters are updated similarly to the update strategy used in Algorithm 2, descending the gradient of the temporal difference loss from predicted and observed rewards incurred by the learned policy.

Unfortunately, our R-GCN model did not seem to improve significantly upon the results of our simpler DQN model. However, it should be noted that we did not have the time to fully test and tune the model, and that the increased computational complexity of the R-GCN DQN did not allow for exhaustive testing sessions. In the end, we observed the same issues as with our uncertainty-based NN and MeLUN, where the network simply approximates a good average prediction and more or less disregards different inputs.

Model	NDCG@10	DIV@10	SER@10	HR@10
AG	0.0974	0.2296	0.1498	0.1953
AG$_{\text{DIV}}$	0.0859	**0.2597***	0.1348	0.1809
AG$_{\text{SER}}$	0.0890	0.2380	0.1458	0.1867
AG$_{\text{HR}}$	0.0950	0.2371	0.1510	0.1943
AG$_{\text{MIX}}$	**0.0978**	0.2465*	**0.1568**	**0.2023**

Table 6: Results after a 5-length interview. Subscript indicates the metric to reorder top-5% candidate questions on. Star indicates statistical significance from non-reordering model ($\alpha = 0.05$).

C. ADDITIONAL EXPERIMENTS

The results covered in the main body of this work are primarily oriented with optimising models for cold-start interviews and recommendations. However, since MindReader is a new dataset and opens up for much interesting new research, we deem it worthwhile to conduct a series of additional experiments specifically addressing what kinds of entities are best to ask about.

C.1 Reordering candidate questions

In our experiments with the AG interviewer, we observed that the difference in NDCG between the top candidate questions is often quite small. As such, we posit that a secondary ordering on another metric, e.g., DIV or SER, could greatly improve the other qualities of the recommendations while largely maintaining ranking quality.

We show the results of reordering the top-5% candidate questions in Table 6 using the equal-popularity sampling dataset. We use PPR-JOINT as the underlying recommender due to its efficiency and high performance in the main experiments. As can be seen, reordering on DIV results in a statistically significant increase in DIV, yet the NDCG does not decrease with statistical significance. A small decrease in ranking quality is expected, as the trade-off between ranking quality and diversity has been documented in existing works [48]. On the other hand, reordering on other metrics, specifically HR and SER, does not appear to cause a significant difference in any of the metrics.

We also consider a combination of all additional parameters in the reordering where the candidate questions are ordered by the mean of all metrics, denoted AG$_{\text{MIX}}$. This approach achieves the best performance across all metrics except on DIV, however a statistically significant increase in DIV is afforded without loss of NDCG.

C.2 Limiting entity types

In this additional experiment, we consider the effect of limiting an interviewer to the different broader entity types in the KG, i.e., people, categories, companies, and decades. We compare against the two settings used in the main body (i.e., only REs and DEs, respectively) and a scenario where all entities can be asked about. Due to time constraints, we consider only the AG interviewer with a PPR-JOINT recommender. Furthermore, we allow the interviewer to interview each user with up to 50 questions. We set the maximum adaptive depth to 10 questions and use a fixed-question approach for the remaining 40 questions. The results from this experiment are shown in Figure 26, where we plot the NDCG@10 at different interview lengths when limited to different entity types.

We observe that asking about decades yields little information, and its performance flattens after only a few questions. This is expected, since there are only 10 decades in the KG and few ratings on older decades. Among the different entity types, asking about categories (i.e., genres and subjects) seems to yield the most information. From Figure 27, which shows the rating variance and how well-known different entity types are, we observe that subjects have high variance and are relatively well-known. Similarly, genres are among the most well-known entities, with relatively high rating variance. On the other hand, movies have high variance, but users are less likely to be familiar with these, hence the performance on REs-only interviews is relatively low.

Finally, we observe that asking towards both REs and DEs becomes the optimal strategy ones the interview becomes long (> 20 questions). By means of adaptive interviews, we can elicit coarser preferences with DEs first and then finer preferences with REs. Indeed, inspections of the generated decision trees show that the interviewer tends to first ask about DEs, then a mix of REs and lesser known DEs in later parts of the interview.

D. TIME COMPLEXITY

We now analyse the complexity of our AG interviewer, as it is the most complex interviewer we have proposed. We denote $\mathcal{E}_{cand}$ as the entities we can ask about, i.e., DEs and REs, and $\mathcal{O}(S)$ as the complexity of the recommender. We can then define complexity of the selection process at node N as $\mathcal{O}(n^{(N)} * |\mathcal{E}_{cand}| * \mathcal{O}(S))$, where $n^{(N)}$ is the number of users in node N. The overall complexity is then $\mathcal{O}\big(|\mathcal{E}_{cand}| * \sum_{N \in \mathcal{N}}[n^{(N)} * \mathcal{O}(S)]\big)$, where $\mathcal{N}$ is the set of nodes in the tree, s.t. $|\mathcal{N}| = \dfrac{d^{m+1}-1}{d-1}$, where d is the maximal degree of a node and m is the interview length. The AG interviewer is therefore exponential in the depth of the interview as it is dependent upon the tree structure, similar to existing works that rely on decision trees for adaptive interviews [5, 27].

While the time complexity is exponential, we found the biggest bottleneck to be the recommender systems as the AG interviewer is directly dependent on the complexity of these. In the following, we explain how the practical runtime can be reduced drastically.

D.1 Caching

As our proposed interviewing strategies support arbitrary recommenders, exhaustively testing candidate questions on a large subset of users can become intractable for complex underlying recommenders on large datasets. To overcome this issue, we have implemented caching on PPR-based models, k-NN, and MF. In our implementation, we cache the scores over all REs given a question-answer state (i.e., finite mapping from entities to ratings).

Caching is particularly useful in preference elicitation for several reasons: First, since the recommenders have no access to learned representation of cold-start users, they are effectively stateless components that always output the same RE scores given the same question-answer state. This is unlike learned models in the warm-start recommendation setting, where each user is represented with an individual latent representation. Second, even though users have individual preferences, many preferences will be similar when evaluating candidate questions. For example, consider the case of determining an initial question for the AG interviewer. For a single candidate question, users can either like, dislike, or state that they don't know it. To reduce the state space of the cache, we can treat a "don't know" answer as a non-answer. With $|\mathcal{E}_{cand}|$ candidate questions, the recommender is then invoked a maximum of $|\mathcal{E}_{cand}| * 2 + 1$ times assuming there is no cache limit.

The number of possible question-answer states will increase with longer interview lengths, resulting in fewer cache hits and therefore more invocations of the underlying recommender. However, for the AG interviewer the number of users will decrease for nodes that are deeper in the tree, resulting in a smaller per-node cost. Furthermore, the users at an arbitrary node share sentiment on the predecessor nodes. In Figure 28, we illustrate this effect by showing the percentage of cache hits at different levels of an AG-generated decision tree. As expected, we observe that deeper parts of the tree generally incur more care misses. In addition, we observe that the percentage of cache hits is lower for the right part of the tree, since red branches group dislike and "don't know" answers. Even with a relatively larger number of cache misses in the lower nodes of the tree, caching drastically reduces the runtime of AG models. We note that such a form of caching is not a possibility for the embedding-based fMF and LRMF models, since these models construct nodes by means of optimising embeddings for all candidate questions and users at a given node.

D.2 Performance on a large dataset

To consider the runtime performance of different interviewers, we measure runtime on 5-length interview construction using a dataset larger than MR-170K and compare against SOTA methods. Specifically, we consider the largest available MovieLens dataset [24], containing 25M movie ratings from 162K users. While larger datasets exist, our KG-based methods are designed specifically for the movies used in the MovieLens datasets, hence they can be used in our cold-start framework without construction of a new KG. For non-SOTA models, we consider PPR-JOINT as the underlying recommender. Runtime for all models is measured using the same machine with 64 GB RAM and an 8-core AMD Opteron 6376 CPU running at 2.3 GHz. Due to time constraints, we allow the models to run for two days.

Model	**Runtime**
FixedPop (FP)	3s
FixedGreedy (FG)	1h 11m
AdaptiveGreedy (AG)	1h 15m
LRMF	$>$ 5h 13m
fMF	DNF

Table 7: 5-length interview construction time on different models on the MovieLens 25M dataset. DNF indicates that the model did not finish within the time window.

From Table 7 we observe that the runtime of AG and FG is quite similar, despite that FG only has to select questions five times. However, this is in compliance with our expectations given the effect of using caching in AG interviews, as described in subsection D.1. In terms of the SOTA models, unfortunately none of them managed to finish interview

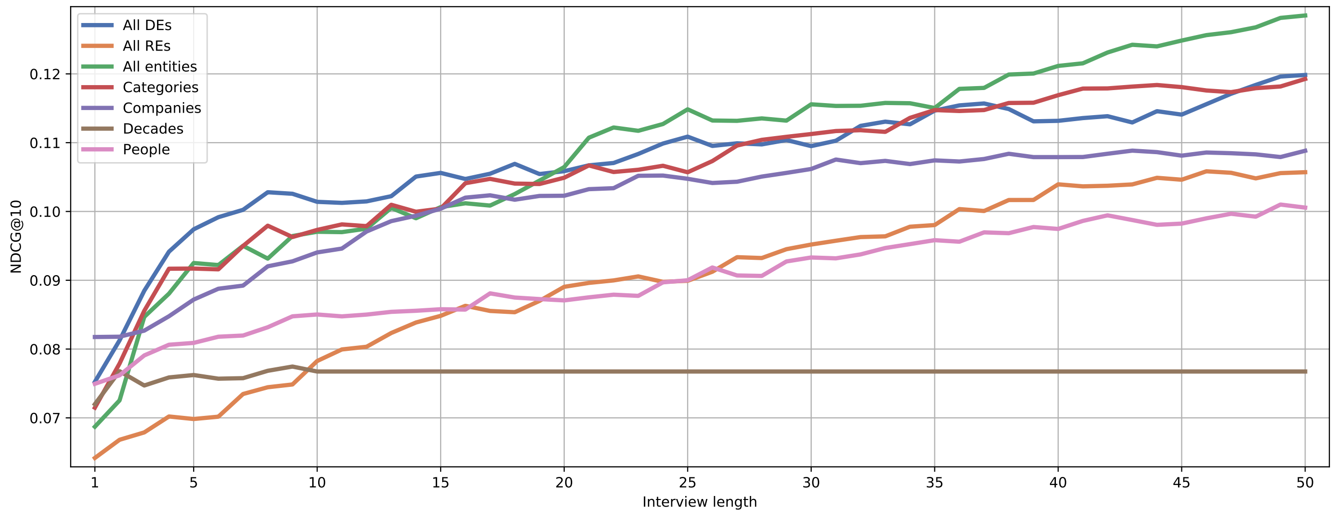

Figure 26: NDCG@10 at different interview lengths where an AG interviewer using a PPR-JOINT recommender is limited to different entity types.

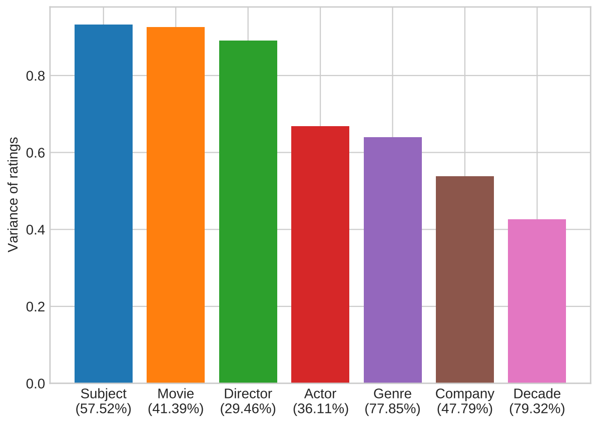

Figure 27: Variance and how well-known different entity types are. Percentages denote the fraction of all ratings for an entity type that are binary ratings, i.e., not "don't know" ratings.

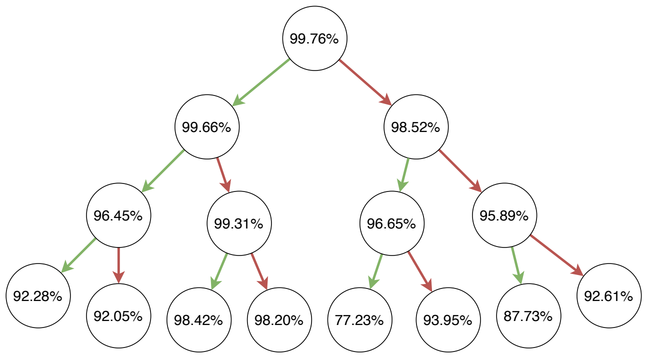

Figure 28: Percentage of cache accesses being cache hits during question selection in the nodes of an AG-generated decision tree. Green branches indicate likes, while red branches indicate dislikes and "don't know" answers.

construction. LRMF managed to construct a 1-length interview but ran out of memory. In the case of fMF, it did not manage to finish within the time window of two days. We note that even on the much smaller MR-170K dataset, construction of a 10-length interview took several days. We expect that fMF is even slower on the MovieLens dataset as its time complexity depends on both the number of entities and number of ratings per user [5].

More generally, we note that few works have focused on the runtime performance of models for preference elicitation. Our implementations of the DQN and DDPG interviewers were an attempt to alleviate this long runtime, yet more research is required to make these methods viable. While fMF in particular appears to be very slow, this is to be expected from how the greedy question selection strategy is implemented in the model. At every node, fMF learns an optimal user embedding for representing all users at the node and records the loss using this representation to predict the users' ratings for a candidate question. This procedure must take place for all remaining candidate questions (limited to the top-100 most popular entities in our experiments) so the candidate question leading to the lowest loss can be selected, which is inherently a very complex procedure even with a small number of candidate questions.

LRMF alleviates part of this issue with fMF by effectively cutting the decision tree in half and replacing the remaining questions with a fixed set of questions identified by means of the Maxvol algorithm, which is much more efficient than building individual nodes by means of testing all candidate questions individually. In our AG approach, we have drawn inspiration from LRMF in only splitting users in two groups and allowing the specification of a maximum adaptive depth of the decision tree.