ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

UNIVERSITY OF PIRAEUS

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGIES
DEPARTMENT OF DIGITAL SYSTEMS

POSTGRADUATE PROGRAM
”INFORMATION SYSTEMS AND SERVICES”

Extending Hypernetwork-Based Recommender Systems for the Cold-Start Problem Using Side Information

By
Konstantinos Panis

Submitted
in partial fulfillment of the requirements for the degree of
Master of Science in
”Information Systems and Services”
with a specialization in ”Big Data and Analytics”
at the
UNIVERSITY OF PIRAEUS
March 2026

Thesis Supervisor: Dr. Maria Halkidi
Title: Professor

Abstract

The cold-start problem constitutes a fundamental challenge in recommender systems, as it requires generating accurate personalized recommendations for users with limited or even no prior interaction history. Although the integration of side information has been widely adopted in traditional recommender systems, its effectiveness within meta-learning hypernetwork-based framework remains insufficiently explored.

This thesis investigates the capabilities of the HyperRS meta-learning model when heterogeneous side information is incorporated under different user cold-start settings. Specifically, the integration of two types of side information is examined: NLP-based item representations derived from SBERT-encoded movie plots and user demographic data. Three extensions of the original HyperRS model are proposed, integrating SBERT-derived movie-plot embeddings, user demographic information and their combination, respectively.

The experimental results indicate that semantically enriched item side information provides marginal performance improvements under moderate cold-start conditions but becomes less effective as cold-start severity increases. In contrast, user demographic information appears to contribute weak personalization signal and does not consistently improve performance. Overall, the results demonstrate that the effectiveness of additional side information in hypernetwork based recommender systems is highly dependent on interaction data sufficiency and that interaction-based signals remain fundamental even within meta-learning frameworks.

The main contribution of this study lies in the systematic extension and evaluation of hypernetwork-based reccomender systems through the integration of different types of side information under varying cold-start conditions, highlighting important considerations for the integration of heterogeneous side information in practical recommendation scenarios.

Περίληψη

Το πρόβλημα της κρύας έναρξης (cold-start) αποτελεί θεμελιώδες πρόβλημα των συστηματών συστάσεων, καθότι απαιτεί την παροχή αποτελεσματικών προσωποποποιημένων συστάσεων σε χρήστες με περιορισμένό ή ακόμη μηδαμινό ιστορικό αλληλεπιδράσεων με το σύστημα. Παρότι η ενσωμάτωση βοηθητικών δεδομένων (side information) έχει ευρέως μελετηθεί στα παραδοσιακά συστήματα συστάσεων, η αποτελεσματικότητα σε συστήματα συστάσεων που ακολουθούν την μετά-εκπαίδευση (meta-learning) και βασιζόνται σε υπερ-δικτύα (hypernetwork) παραμένει ανεπαρκώς εξερευνημένη.

Η παρούσα διπλωματική εργασία εξετάζει τις δυνατότητες του μετά-εκπαιδευομένου μοντέλου HyperRS με την ενσωμάτωση ετερογενών βοηθητικών πληροφοριών. Συγκεκριμένα, μελετάται η ενσωμάτωση δύο τύπων βοηθητικών πληροφοριών: αναπαράστασεις αντικειμένων βασισμένες σε τεχνικές επεξεργασίας φυσικής γλώσσας (Natural Language Processing) οι οποίες έχουν αντληθεί από πλοκές ταινιών κωδικοποιημένες από μοντέλο SBERT, δημογραφικά δεδομένα του χρήστη και ο συνδυασμός των δύο πληροφοριών.

Τα αποτελέσματα της έρευνας υποδεικνύουν ότι οι εννοιολογικά εμπλουτισμένες αναπαραστάσεις αντικειμένων αποδίδουν ελάχιστη βελτίωση στην απόδοση του συστήματος ύπο ήπιες συνθήκες κρύας έναρξης, ενώ η αποτελεσματικότητα τους μειώνεται όσο η σοβαρότητα της κρύας έναρξης αυξάνεται. Εν αντιθέσει, τα δημογραφικά στοιχεία του χρήστη δεν φαίνεται να βελτιώνουν την απόδοση. Γενικά, τα αποτελέσματα υποδηλώνουν ότι η αποδοτικότητα της επιπρόσθετης βοηθητικής πληροφορίας επηρεάζεται σημαντικά από το μέγεθος των διαθέσιμων δεδομένων αλληλεπίδρασης, και ότι τα μοτίβα που αποκαλύπτουν οι αλληλεπιδράσεις παραμένουν θεμελιώδη ακόμα και στα μοντέλα μετά-εκπαίδευσης.

Η κύρια συνεισφορά της παρούσης εργασίας έγκειται στην συστημική επέκταση των συστημάτων συστάσεων βασισμένα σε υπέρ-δίκτυα μέσω της ενσωμάτωσης διαφορετικών τύπων βοηθητικής πληροφορίας υπό διαφορετικές συνθήκες κρύας έναρξης, τονίζοντας σημαντικούς ενδοιασμούς στην ενσωμάτωση τέτοιων πληροφοριών σε πραγματικά σενάρια συστημάτων συστάσεων.

Acknowledgements

I would like to express my gratitude to everyone who contributed to the completion of this Master's thesis.

First, I would like to warmly thank my thesis supervisor, Professor Maria Halkidi, for her insightful advices, constant availability, and fruitful cooperation. Her guidance was vital for completing this thesis and developing my research skills.

I am also thankful to all the faculty staff for introducing me to diverse areas of Data Science and encouraging me to move beyond my comfort zone.

Last but not least, words cannot express my gratitude to my family and friends. This endeavor would not have been possible without their emotional support and unwavering encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we describe in detail the thesis topic, its academic relevance, motivation, objectives and the research questions to be addressed. Moreover, an overview of the thesis structure and a brief introduction to the following chapters are provided.

1.1 Background and Motivation

In recent years, vast amounts of data have been generated every day overwhelming Internet users and consumers with unprecedented volumes of information. Recommender systems play a pivotal role in personalizing the user experience by surfacing the most suitable data based on each user's preferences. Common use cases of recommender systems include e-commerce platforms, social media applications and entertainment platforms such as Amazon, Facebook and Netflix [1]. Their objective is to identify patterns in user behavior and recommend items that users are most likely to interact positively with, thereby offering a personalized experience.

A positive interaction can vary depending on the domain in which the recommender system is used and the specific use case. For example, on an e-commerce website, a positive interaction may involve either a user actually purchasing a recommended item or simply adding it to their wishlist. In the context of an entertainment platform, such as Netflix, watching a movie or watching a movie and giving a high rating can both indicate a positive interaction.

Although recommender systems have proven crucial for both users and companies, several challenges remain of interest to the research community. A well-known challenge faced by these systems is the cold-start problem, which occurs when a new user (user cold-start) or a new item (item cold-start) is introduced into the system. In that scenario, limited or even nonexistent user-item interactions can

pose a significant hurdle to making effective recommendations. As described in [7], many researchers examine either data-driven techniques or approach-driven techniques. Data-driven techniques aim to utilize available data effectively, for example by using personal region data or social community data, while approach-driven techniques focus on improving existing algorithms or developing new ones.

The methodology proposed in this study builds upon the recently introduced HyperRS model [6], a hypernetwork-based recommender designed to handle user cold-start situations through meta-learning. By leveraging a hypernetwork to generate user-specific parameters conditioned on limited user-item interactions, HyperRS enables rapid adaptation to new users with only a few observed interactions. This meta-learning formulation allows the model to generalize across users and transfer learned preference structures, thereby mitigating the data sparsity inherent in cold-start scenarios. Specifically, the proposed methodology extends this framework by leveraging user demographic information and semantic text embeddings from movie descriptions. These additional information are expected to provide complementary signals that can guide the adaptation process when interaction data are scarce. To systematically evaluate the impact of heterogeneous side information, three model variants are introduced: P-HyperRS, which incorporates semantic plot embeddings; UD-HyperRS, which integrates user demographic attributes; and PUD-HyperRS, which combines both information. These variants enable controlled comparisons across different cold-start settings.

The main contribution of this study lies in the systematic extension and evaluation of the HyperRS meta-learning framework through the integration of heterogeneous side information, including user demographics and semantically enriched item representations, under varying cold-start conditions. To the best of our knowledge, this is the first study that jointly investigates these information sources within a hypernetwork-based recommender in a cold-start setting.

1.2 Research Objectives and Questions

It is well established in the research community that leveraging side information in recommender systems and NLP-powered embeddings can improve performance and even alleviate the cold-start problem ([8], [9], [10], [11], [12], [13]). Although this is a common practice, meta-learning-based recommenders often lack effective integration of comprehensive user context, as highlighted in the literature review on meta-learning methods for the cold-start problem conducted by Zawia et al. [14].

Motivated by these findings, the aim of this research is to explore the capa-

bilites of the HyperRS model when user data are available and to provide insights into its behavior across different types of side information and different cold-start settings. Specifically, user demographic information is utilized to improve user personalization, while NLP-powered item attributes are employed to enhance item representations. The performance of HyperRS extensions is evaluated under different cold-start settings; in the first setting, we simulate a moderate user cold-start and in the second setting we simulate a stricter user cold-start with limited known user-item interactions. From a theoretical perspective, integrating user and item side information can provide additional contextual priors that complement sparse interaction signals during meta-learning adaptation. In cold-start scenarios, where the initial user interactions provided are limited, such auxiliary data may guide the hypernetwork toward more stable and informative user-specific parameter generation. Therefore, examining how different types of side information interact with varying levels of interaction scarcity is central to understanding the robustness of the HyperRS framework. The methodology is evaluated on *MovieLens-1M* dataset [15], which contains one million movie ratings from thousands of MovieLens users, along with user demographics and item attributes. To further enrich the item attributes, movie plots were retrieved using the OMDB API [16].

To better formulate the problem and the objectives of this research, the main research questions addressed are as follows:

- **RQ1:** *How do different types of side information affect HyperRS performance?*
- **RQ2:** *How do different cold-start settings affect HyperRS performance?*

By addressing these research questions, this study aims to provide empirical insights into the robustness and adaptability of HyperRS under varying cold-start scenarios, while systematically examining the impact of different types of side information.

1.3 Overview

The remainder of the study is organized as follows: Chapter 2 presents a detailed literature review along with a brief description of the preliminaries. Chapter 3 introduces the conceptual overview of the proposed approach, its mathematical formulation and the proposed system architecture. Chapter 4 describes the research methodology including the dataset description, data collection procedure and the evaluation setup. Chapter 5 presents the experimental results organized and supported by tables and figures. Chapter 6 provides an interpretation of the experi-

mental results, addressess the research questions and discusses the study's limitations. Finally, Chapter 7 concludes the study and examines future research directions.

Chapter 2

Literature Review and Preliminaries

In this chapter a detailed literature review on recommender systems and the cold-start problem in recommender systems is displayed. Specifically, traditional recommender system architectures are presented first in Section 2.1. Then, we review approaches on cold-start problem in recommender systems (Section 2.2) and mainly focus on meta-learning approaches (Section 2.2.1) and HyperRS model (Section 2.2.1). A review on side information and Natural Language Processing (NLP) techniques utilization in recommender systems is presented in Section 2.3. Finally, we briefly describe the BERT and SBERT language models in Section 2.4.

2.1 Traditional Recommender Systems

Recommender systems emerged in the early 1990s, evolving from information retrieval and information filtering as a response to information overload caused by the rapid growth of the Internet. In particular, Tapestry is often considered the first recommender system. It was an experimental mail system designed to enable users to subscribe to mailing lists of their interest by employing content-based and collaborative filtering techniques [17]. Generally, traditional recommender systems can be partitioned into two foundational categories: content-based filtering and collaborative filtering (also refered to as neighborhood-based).

Content-based recommender systems aim to identify items similar to those users have liked in the past [1]. In this setting, recommendations are not based on the target user's rating history or the rating histories of similar users, but rather on the attributes of items previously liked by the target user. A common approach in content-based recommender systems is k-nearest neighbors. Suppose we wish to predict the rating that a target user *U* will give to an item *I*. A similarity metric, such as cosine similarity, is computed for each (*I*, I_{rated}) pair, where I_{rated} denotes

each item the target user has rated. These similarities are then sorted in descending order, so that the first items are the most similar to the target item. The predicted rating for target item *I* is computed as the average rating of the top-k most similar items weighted by their similarity.

An important advantage of content-based methods is that they can be particularly effective in cold-start settings, as they do not depend on ratings from other users ratings when sufficient information about the target user is available. However, relying solely on attributes of items liked by the target user may lead to recommending only highly similar items, thereby reducing the diversity and novelty of the recommendations-a phenomenon commonly referred to as *"filter bubbles"* [18].

Despite being a classic approach, content-based recommenders remain an active area of research with numerous extensions and improvements proposed in recent work. In [19], Shapley Value-guided Embedding Reduction (Shaver)-an embedding pruning method for compressing on-device content-based recommender systems to any target size in one shot-was introduced. Shaver addresses the problem of compressing embedding tables used in content-based recommenders by modeling it from a cooperative game perspective and enabling contribution-based parameter pruning through the estimation of Shapley values. Liu et al. proposed a Large Language Model (LLM)-boosted content-based recommender system that leverages both open- and closed-source LLMs [20]. The proposed framework, termed ONCE, employs open-source LLMs as content encoders to enrich item representation and closed-source LLMs with prompting techniques to enrich training data at the token level. Finally, Tuval et al. [21] further demonstrated the effectiveness of their earlier approach in [22], which introduced a content-based model that employs hypercube graphs. In particular, their model is used to infer user preferences with limited rating history while preserving user privacy, thereby helping to alleviate the cold-start problem.

On the other hand, collaborative filtering (CF) methods predict ratings based on similar user rating history or similar item rating behavior. In other words, collaborative filtering assumes that similar users display similar rating behaviors and similar items receive similar ratings [1]. This method can be categorized into two primary types: user-based collaborative filtering and item-based collaborative filtering.

User-based collaborative filtering considers the setting in which similar users rating history is utilized to make predictions for a target user U. The first step in user-based collaborative filtering is to identify the neighborhood of the target user U, i.e., to find the top-K most similar users. To achieve this, the similarity of the target user U to all other users is calculated using a similarity function such as the

Pearson correlation coefficient (Equation 2.1).

$$\text{Sim}(u,v) = \text{Pearson}(u,v) = \frac{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u) \cdot (r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u)^2} \sqrt{\sum_{k \in I_u \cap I_v} (r_{vk} - \mu_v)^2}} \quad (2.1)$$

where u, v are users, $k \in I_u \cap I_v$ denotes the set of commonly rated items, r_{uk} and r_{vk} are ratings, and μ_u, μ_v denote users' mean ratings.

To predict the rating that the target user U will give to a target item I, the closest k users who have rated item I are identified. The final predicted rating is the weighted average of those users' ratings for item I.

Item-based collaborative filtering follows a similar process, except that the similarities are computed between items rather than users. Before computing item similarities, the average rating of each item is subtracted from each rating. A commonly used item similarity metric is the adjusted cosine similarity, defined in Equation 2.2.

$$\text{AdjustedCosine}(i,j) = \frac{\sum\limits_{u \in U_i \cap U_j} s_{ui} \cdot s_{uj}}{\sqrt{\sum\limits_{u \in U_i \cap U_j} s_{ui}^2} \sqrt{\sum\limits_{u \in U_i \cap U_j} s_{uj}^2}} \quad (2.2)$$

where i, j are items, $u \in U_i \cap U_v$ denotes the set of users who rated both items i, j and s_{ui}, s_{uj} are the mean-centered item ratings.

The top-k most similar items to a target item i are determined using Equation 2.2. The predicted rating for the target item i is calculated as the weighted average of those top-k similar items, for which the user has specified ratings. In this way, a user's ratings are essentialy predicted based on their own rating behavior for similar items.

Collaborative filtering recommender systems are widely used across different business domains and applications due to several advantages. One key advantage is that they employ a very simple and intuitive approach: similar users display similar rating behavior and similar items are rated similarly. This also allows better explainability of recommendations. For example, in item-based CF, a recommendation can be justified by showing the item(s) that led to the recommendations with wording such as: "*Because you liked Item I:* ". Moreover, CF recommenders are less prone to system changes due to the addition of new users or items, rendering them relatively stable. Another important advantage is that CF recommenders are capable of producing more diverse recommendations, particularly in user-based CF, where combining rating histories from similar user can surface items reflecting each user's unique taste.

Despite these advantages, there are several challenges faced by CF recommenders. Specifically, it may be hard to find similar users if they have not rated the same items, a problem also referred to as *neighbor transitivity* in [23]. Additionally, the computational complexity of calculating user-to-user or item-to-item similarities, also called the offline phase, can become prohibitive in large-scale settings, posing a significant hurdle to scalability. Finally, CF recommenders are storngly impacted by the long-tail distribution of ratings. In such distributions, only a few items are rated frequently by the users, resulting in a high concentration of ratings on popular items and long tail of rarely rated items (Figure 2.1). Consequently, CF recommenders tend to recommend primarily popular items, or, in other words, items that are rated more frequently by users [1].

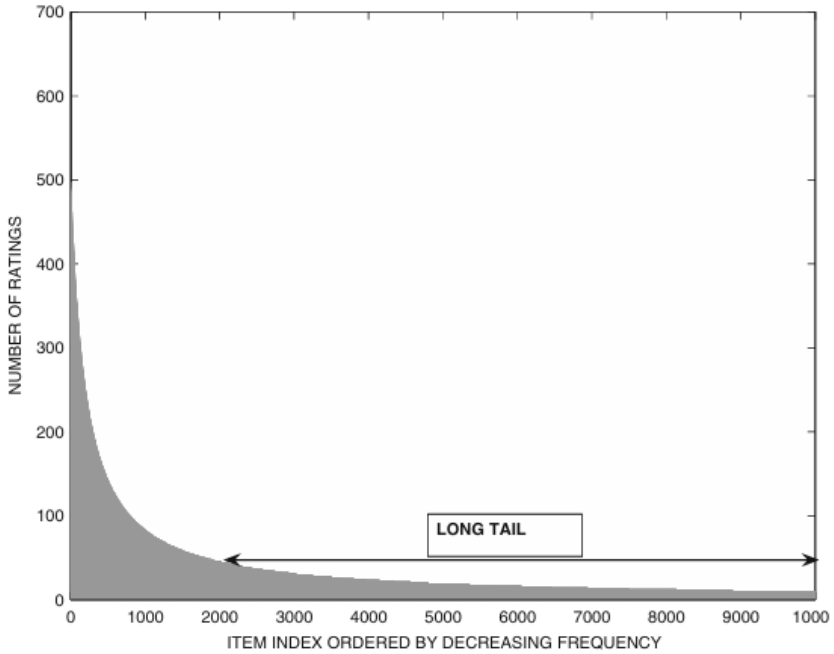

Figure 2.1: *The long-tail distribution of ratings (Reproduced from [1])*

The intuitiveness and simplicity of CF approach is also a main reason why it remains an active research area with plenty of recent work focus to overcome its challenges and improve its performance by integrating modern approaches such as LLMs. In [24], an LLM-enhanced CF framework was proposed by finetuning LLaMA2 [25] to adapt to recommendation tasks. In particular, the fine-tuned LLaMA2, namely *RecGen-LLaMA*, was utilized to generate Chain of Thought (CoT) reasoning with CF information and construct in-context CoT dataset. The latter was then retrieved in order to learn the world knowledge and reasoning guided CF feature, and use this feature to enhance the recommender system. Xia et al. focused on leveraging the relational dependencies of user heterogeneous interactions and capturing the fine-grained latent factors and proposed a heteroge-

neous graph collaborative filtering, termed *MixRec* [26]. *MixRec* was developed to disentangle users' multi-behavior interaction patterns and uncover the latent intent factors of each behavior. Moreover, in [27], the challenges introduced by the long-tail distribution of the ratings was attempted to be tackled by proposing a fair sampling method for collaborative filtering recommenders. The aim of the method developed was to avoid overrepresentation of popular items as positive samples by ensuring each user and item has an equal likelihood of being selected as both positive and negative samples in training. In [28], a Graph Convolution Network-based recommender, namely *Cluster-based Collaborative Filtering* (*ClusterGCF*), was proposed. *ClusterGCF* captures the multiple interest of users and identify the common interests among them in order to perform high-order graph convolution on cluster-specific graphs.

Another well-established type of recommender system which has been widely used is Latent Factor Models. Assuming a rating matrix R_{nXm} where each element r_{ij} denotes the user rating i for the item j, the basic idea is to uncover the evident correlation of the data between the rows and columns of the matrix. This correlation suggests that the rating matrix can be approximated efficiently by a low-rank matrix. Sarwal et al. examined the application of *Singular Value Decomposition* (*SVD*), which is essentially a matrix factorization technique formulated by Equation 2.3, in recommender systems [29].

$$R = U \cdot S \cdot V^T \tag{2.3}$$

where R denotes the ratings matrix of size $n \times m$, U denotes the user latent factor of size $n \times r$, S denotes the singular values diagonal matrix of size $r \times r$ and V^T denotes the item latent factor of size $r \times m$. Matrix R can be quite well approximated by its rank-k reconstructed matrix if k (k < r) singular values be retained and matrices U, V be reconstructed accordingly (Equation 2.4).

$$R_k \approx U_k \cdot S_k \cdot V_k^T \tag{2.4}$$

2.2 Cold-Start Problem in Recommender Systems

It is undeniable that recommender systems play a vital role in offering users a tailored experience while enhancing companies' revenue. However, the cold-start problem poses a significant hurdle to deploying efficient recommenders early on. The cold-start problem arises when a new user (user cold-start) or a new-item (item cold-start) enters the system and there is limited, or even non-existent, user-item interaction history. In such cases, providing meaningful recommendations

becomes a challenging task that demands the employment of non-traditional approaches. The approaches utilized to mitigate the problem can be classified as data-driven or approach-driven, as described in [7]. It is worth noting that even content-based approaches can, to some extent, alleviate the user cold-start problem, as they rely solely on item attributes; however, they still require sufficient user interest information [1] and cannot handle the item cold-start scenario.

Data-driven techniques aim to provide meaningful recommendations by efficiently utilizing any available data. Such data may include personal user information, such as age or country of residence, or cross-domain data, such as user's social network activity. Essentially, data-driven techniques target the root of the problem, which is data insufficiency, and endeavor to capitalize fully on any available information, thus tackling the cold-start problem. Conversely, approach-driven techniques focus on improving existing algorithms or developing new ones. These methods aim to mitigate the cold-start problem by dealing efficiently with limited user-item interaction data rather than directly addressing data insufficiency. A wider variety of methodologies is applicable on such techniques, ranging from traditional techniques, such as content-based, collaborative filtering and latent factor models, to deep learning and meta-learning. Meta-learning approaches are discussed in detail in Section 2.2.1.

In [30], a review aware cross-domain recommender, termed *RACRec*, was proposed to address the user cold-start problem. *RACRec* incorporates valuable product reviews into a cross-domain migration model to infer user preference vectors. In other words, reviews from different product domains (source domain) are used to infer user preferences and enhance product recommendations in the target domain. Lai et. al introduced an end-to-end Generative Adversarial Network (*GAN*), namely *ColdGAN*, which leverages the rating distributions of experienced users to predict ratings for cold-start users [2]. Specifically, they proposed a rejuvenation function that converts a user's warm-state into a cold-state, which is then input to a generative network *G*. *G* attempts to generate warm-states and deceive a discriminative network *D*, which receives the actual warm states (Figure 2.2(a)).Once the *GAN* is trained, it can be employed to generate warm-state predictions given users' cold-state distributions (Figure 2.2(b)). In [31], a social-aware recommendation system, Social Importance and Category Enhanced Recommendation System (*SICERec*), was proposed to alleviate the user cold-start problem by efficiently integrating user social network information. *SICERec* consists of three main components. The first extracts user preferences from user-item historical interactions. The second incorporates the centrality attributes of users in the social network graph in order to enrich the semantic representation of users. Finally, item representations are enhanced by utilizing the category information of

items with which users have interacted.

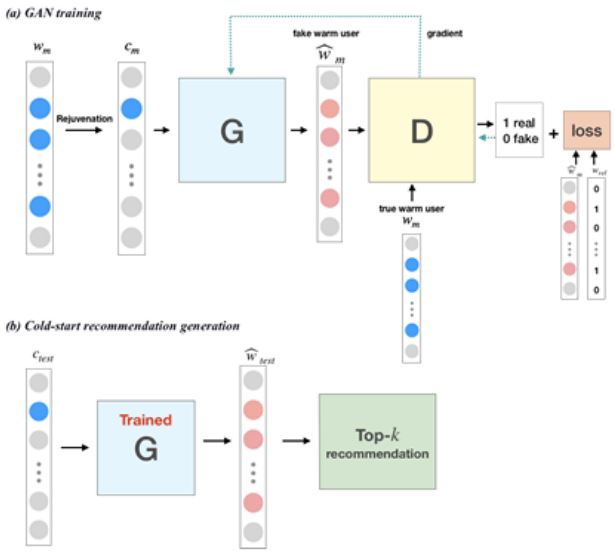

Figure 2.2: *ColdGAN architecture. (a) GAN training phase: Learn the rating distributions given their cold-start distributions. (b) Recommendation inference: Employ trained GAN to infer recommendations (Reproduced from [2])*

The aforementioned approaches constitute a subset of approaches utilized to alleviate the cold-start problem and are referenced for demonstration purposes, although a considerably broader range of methods exist in the literature. Nonetheless, most of the available approaches exhibit substantial limitations. In the systematic review conducted in [32], the approaches examined regarded *GANs*, Associative Rule Mining, Active Learning and Data Imputation. Although these methods efficiently address data sparsity, generate better initial recommendations and enhance data accuracy, practical considerations such as computational complexity and adaptability in real-world scenarios limit their scalability and adoption in typical business cases. Furthermore, a comprehensive review on meta-learning approaches to address the cold-start problem highlighted the importance of designing scalable and computationally efficient recommender systems [14]. The review also emphasized fine-tuning recommender systems for the e-learning domain, improving cross-domain systems' robustness, integrating comprehensive user context and addressing fairness and privacy issues.

2.2.1 Meta-Learning Approaches

In the last decade, deep learning models have gained significant popularity, with applications in a wide range of domains such as biology, finance and sports, due to their ability to capture non-linear relationships efficiently. However, it is well known that one major limitation is their requirement for vast amounts of data in

order to capture complex patterns, thus notably increasing training time [33]. Meta-learning, or ”learning how to learn”, enables models to adapt quickly to new tasks and therefore reduce training time by learning from a limited set of examples. Essentially, such models are trained on a limited number of tasks and learn to adapt to new ones by leveraging prior knowledge.

Meta-learning models can be distinguished into three main categories, namely *metric-based*, *model-based* and *optimization-based*. *Metric-based* models leverage the notion of measuring the similarity between two data points based on distance metrics. Well-established and influential metric-based techniques include *Siamese Networks* [34], *Matching Networks* [35] and *Prototypical Networks* [36]. Model-based approaches, such as *Memory-Augmented Neural Networks* (*MANNs*), aim to develop models or improve existing ones that are able to rapidly adapt to new tasks by leveraging prior knowledge. Finally, optimization-based models focus on inferring optimal model parameter initialization for new tasks, thereby optimizing the learning process itself (meta-optimization). The model utilized in this study, HyperRS, also belongs to the optimization-based meta-learning category for recommendations. For this reason, we examine related work that mainly focuses on meta-learning approaches.

Model-Agnostic Meta-Learning (*MAML*), proposed in [37], is one of the earliest and most well-established meta-learning frameworks enabling any model trained with the gradient descent process to gain fast learning on new tasks. Specifically, *MAML* infers model parameters that are highly sensitive to task changes, so that making minor adjustments to these parameters, guided by the gradient of the loss function, results in significant performance improvement on new tasks. Lee et al. proposed a *MAML*-based meta-learning model, termed *MeLU*, which uses user and item attributes to estimate user preferences [3]. Particularly, *MeLU* feeds the concatenation of user and item attributes into a multi-layer neural network that is leveraged to estimate user-preferences. As shown in Figure 2.3, the user and item embedding parameters are randomly initialized in the support set (the user's limited interaction history), and the concatenated embeddings are used to make predictions. The gradient of the loss on the support set is utilized to locally update the multi-layer neural network's parameters. Finally, the loss function computed on the query set prediction is used to globally update the model parameters. In [38], a Meta-learning-based Cold-Start Sequential Recommendation Framework, termed *Mecos*, was proposed to alleviate the item cold-start problem. In order to predict a user's next action, in an item cold-start setting, *Mecos* utilizes a matching network to align support sequence pair embeddings with query sequence pair embeddings, thus optimizing model's parameters through meta-optimization. *Mecos* constitutes a framework that can be effortlessly adapted by

other neural-network-based sequential recommenders, with no need for further fine-tuning after the training has been completed. Additionally, a user-relevance-based adaptive meta-learning model, namely *AdaML*, was proposed in [4] to enhance relevance-based recommender systems performance and generalizability in the user cold-start setting. Relevance-based recommender systems aim to address the limitation of assigning the same global parameters (priors) to all users' learning tasks. To achieve this, they infer locally shared parameters for similar users, thereby providing better user-specific parameters initialization (Figure 2.4). In particular, *AdaML* attempts to identify similar users employing clustering-based and meta-path-based approaches and learning different dynamic weights for each type of related users. Pu et al. proposed a different relevance-based meta-learning recommender, termed *AdaMO*, which utilized user features to personalize the global prior [39]. This model also captured user-preference transitions automatically by introducing a *patterns transition miner*, hence preventing task-level overfitting.

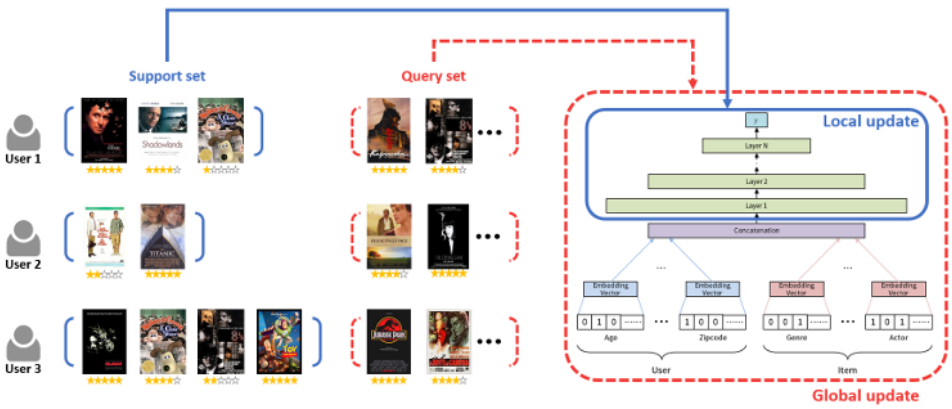

Figure 2.3: *MeLU architecture (Reproduced from [3])*

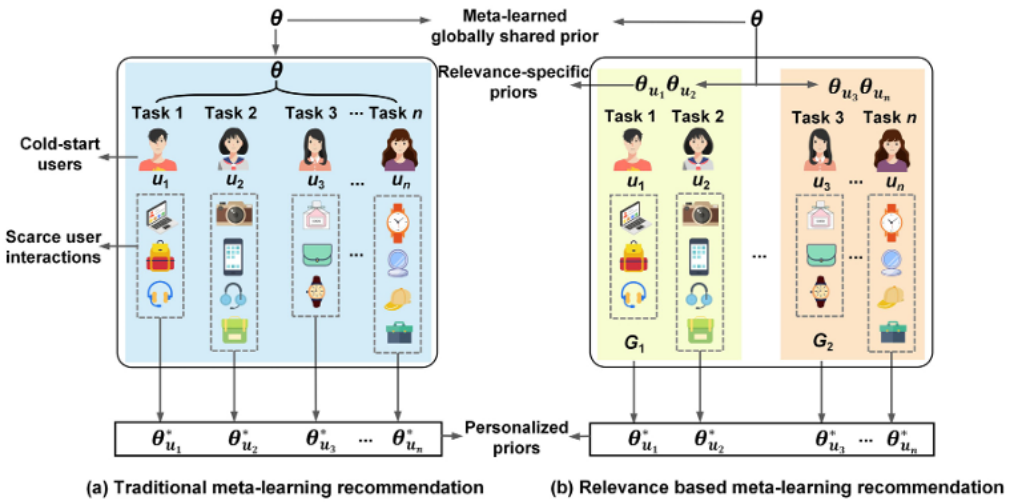

Figure 2.4: *Comparison of traditional and relevance-based recommender systems architecture (Reproduced from [4])*

2.3 Side Information and Natural Language Processing in Recommender Systems

In the context of recommender systems, side information refers to user attributes such as age, gender and occupation as well as item attributes such as genre (e.g. for movies or music tracks), descriptions and even item photographs or videos. Item side information is usually available in most cases and can be utilized without major concerns. On the other hand, user attributes are not always easily available to a system and their utilization may raise crucial concerns. For example, a user signing up for an e-commerce platform is often prompted to provide some basic demographics, such as city of residence, age and occupation. However, if this information is not mandatory for creating an account, the user may choose not to share it and instead provide only the mandatory information, such as an email address and payment details. As a result, a comprehensive user context is often limited. In addition, even when such data are available, they must be utilized carefully to address fairness and privacy concerns that arise. Despite the challenges of integrating side information into recommender systems, a considerable portion of previous work highlights improvement in system efficiency when such information is incorporated, indicating that the performance gains outweigh these challenges.

In particular, in [8] a neural-network-powered *Probabilistic Matrix Factorization* (*PMF*) model was explored in the e-commerce domain. A convolutional neural network (*CNN*) was employed to extract features from item photographs and embedd

them into *PMF*. Strub et al. suggested a hybrid recommender system that employs Autoencoders for Collaborative Filtering (CF) with integrated side information to alleviate the cold-start problem [9]. Although the use of auxiliary data had a limited impact on performance improvement on all users/items, it had a significant impact on cold user/items. In addition, in [10] a new measure of *user-preference dynamics* (*UPD*), which captures the rate at which the current preferences of each user have shifted, is combined with user side data to make recommendations based on a *Coupled Tensor Factorization* (*CTF*) technique. Last but not least, in [11], side data from movie trailers posted on YouTube such as likes, views and comments are utilized to enhance the performance of Matrix Factorization (MF) and Deep Neural Networks (DNN) models.

Although item attributes such as genre can be vital for distinguishing items, more complex item metadata can further enhance item representation by uncovering niche details. For example, a movie's genre is important for categorizing movies; however, movie reviews may be significantly more informative, as they capture specific details about the plot or aesthetics. *Natural Language Processing* (*NLP*) techniques are capable of handling such unstructured data efficiently through understanding the structure and semantics of textual information. As a result, incorporating *NLP* techniques into recommender systems enables the utilization of a wider variety of item attributes, leading to significant performance improvement. In [13] NLP techniques were employed to integrate user review information in a collaborative filtering-based hotel recommendation system, demonstrating considerable performance gains. Moreover, Xu proposed an NLP-based biased Singular Value Decomposition (SVD) model to improve recommender system performance [12]. Specifically, by combining NLP techniques with biased SVD,this methodology achieved more effective textual information representations while addressing inherent biases in user-item interactions.

2.4 Language Models: BERT and SBERT

Language models are a fundamental component of *NLP* which enable us to efficiently understand human language. They have been widely used for tasks such as text classification, machine translation, question answering and text generation.

Bidirectional Encoder Representations from *Transformers* (*BERT*) [40] leverages the encoder component of Transformers [41] to capture contextual information from text data. Specifically, *BERT* is pre-trained in two different tasks: the *masked language model* (*MLM*) and the *next sentence prediction* (*NSP*). During

the MLM task, a random portion of the input tokens is masked and the model is forced to predict the masked tokens using both right and left surrounding context, thereby enabling bidirectional representations. For the NSP task, sentence pairs are concatenated into a single input sequence and provided to the model. A special separator token is used to distinguish between the two sentences, and a learned embedding indicating the sentence to which each token belongs is added. In essence, the NSP task can be perceived as a binary classification problem that aims to predict whether the second sentence in a pair follows the first sentence. After the model has been pre-trained, it can be fine-tuned to any downstream task, such as paraphrasing or question answering, with minimal task-specific parameters.

Although *BERT* demonstrates strong performance on tasks such as text classification, paraphrasing and question answering, its effectiveness is limited in semantic textual similarity tasks. This limitation arises because BERT generates embeddings at token-level rathert than sentence-level.To address this issue, Reimers and Gurevych proposed a BERT-based model, namely *Sentence-BERT* (*SBERT*), which leverages siamese or triplet network architecture to construct semantically rich sentence representations [5]. Specifically, three different tasks can be employed to fine-tune BERT and derive *SBERT*; *Natural Language Inference* (*NLI*), *Semantic Text Similarity* (*STS*) and *Triplet Dataset*. In the NLI task, a siamese BERT network is utilized to classify the relationship between two sentences, given three possible labels, namely *"contradiction"*, *"entailment"* and *neutral*. The concatenation of the two sentence embeddings u and v, along with their difference $|u-v|$ is passed to a softmax classifier to predict the sentence-pair label (Figure 2.5). In the STS task, a siamese BERT network is employed to compute sentence embeddings for a given sentence pair. Afterwards, the cosine similarity of the embeddings is calculated to investigate the similarity of the sentences. Finally, in the triplet dataset setting, BERT is fine-tuned using three sentences defined as follows:

- *anchor sentence a*: the reference sentence used for comparison
- *positive sentence p*: a sentence that is semantically related to the *anchor sentence a*
- *negative sentence n*: a sentence that is not semantically related to the *anchor sentence a*

Sentences a, p, n are provided as input to the triplet network (Figure 2.6). The weights of the network are optimized based on the triplet loss function (Equation

2.5), so that the sentence embedding of a is closer to sentence embedding of p than to the sentence embedding of n in the embedding space.

$$max(||s_a - s_p|| - ||s_a - s_n|| + \epsilon, 0) \tag{2.5}$$

where s_i denotes the sentence embedding of sentence i, $||.||$ represents a distance metric, and e the minimum distance margin.

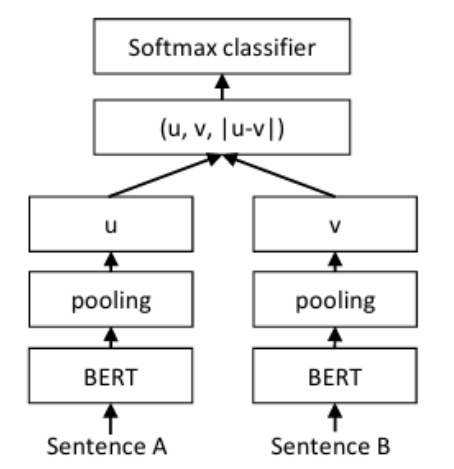

Figure 2.5: *SBERT fine-tuning for Natural Language Inference (Reproduced from [5])*

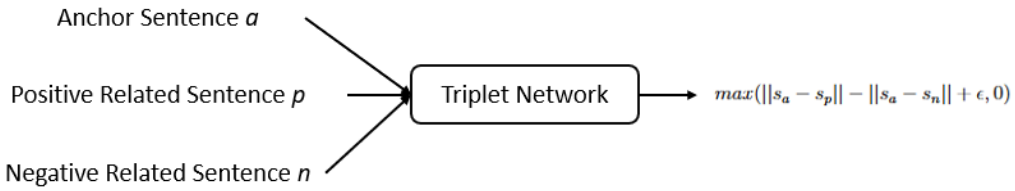

Figure 2.6: *SBERT fine-tuning for Triplet Dataset*

Chapter 3

Proposed Approach

In this chapter, we present the conceptual overview, system architecture, and mathematical formulation of the proposed approach. The chapter is organized as follows: Section 3.1, describes the general idea and structure of the proposed system; Section 3.2 presents the system architecture and the mathematical formulation in detail; and Section 3.3 introduces the three proposed HyperRS extensions, each discussed in its corresponding subsection.

3.1 Conceptual Overview

The proposed methodology constitutes an extension of the meta-learning model termed HyperRS, which was recently introduced in [6]. HyperRS is a hypernetwork-based meta-learning model that follows the Model-Agnostic Meta-Learning (MAML) framework, which was briefly described in Section 2.2.1. Specifically, HyperRS comprises two neural networks: a base-learner neural network (also referred to as the recommender network), that produces the rating predictions, and a hypernetwork, that generates user-specific parameters for the base learner. The objective of HyperRS is to learn the hypernetwork's weights from limited user-item interactions so as to generate improved user-specific parameter initialization for the recommender network in the cold-start setting. The model is provided with user and item side information and leverages their corresponding embeddings to infer a user-interest embedding. This embedding reflects each user's preferences over item content attributes and enables personalized initialization of the recommender network's parameters.

This framework serves as the foundation of the present study, which extends HyperRS by incorporating additional side information, such as user demographics and movie content. Concretely, in this study we examine the extension of HyperRS by *(i)* incorporating NLP-derived movie plot embeddings alongside movie

genres and movie year of release as item side information, *(ii)* incorporating user information such as occupation and age in the user embedding generation component and *(iii)* combining both approaches in the same system architecture.

The motivation of incorporating movie plot embeddings is to capture more niche and contextual movie details that categorical item attributes are unable to provide. Hence, the integration of movie plot embeddings in item features is intended to capture richer contextual item representations within the recommender network that aim to enrich the user interest preferences over item attributes. Additionaly, unlike HyperRS, which relies solely on item features, our approach incorporates user features aiming to enrich the user representation input signal, enabling the generation of more accurate user-specific parameters for the recommender network.

It should be noted that, even though Lu et. al [6] excluded demographic information , that decision was a design choice, rather than a fundamental limitation of the model. Consequently, the proposed approach investigates the model's capabilities when both user demographic data and item content features are available, aiming to provide insights into its behavior across different types of side information.

3.2 HyperRS framework

As mentioned previously, the proposed methodology builds upon the HyperRS framework and incorporates different types of item and user side information. For this reason, this section presents a brief summary of the HyperRS framework as introduced in [6], focusing on the components relevant to our extension. Specifically, in Subsection 3.2.1, we present the system architecture and the model's mathematical formulation, while in Subsection 3.2.2, we present the training procedure.

3.2.1 System Architecture

HyperRS employs two neural networks–the recommender network and the hypernetwork–in order to learn how to generate optimal user-specific parameters given limited user-item interactions. Figure 3.1 shows a visual representation of the HyperRS architecture and its main components.

The recommender network is utilized to predict item scores. It is provided with user and item side information and makes predictions based on their corresponding embeddings. Each item $i \in I$, where I is the set of items, is described by k attributes as follows:

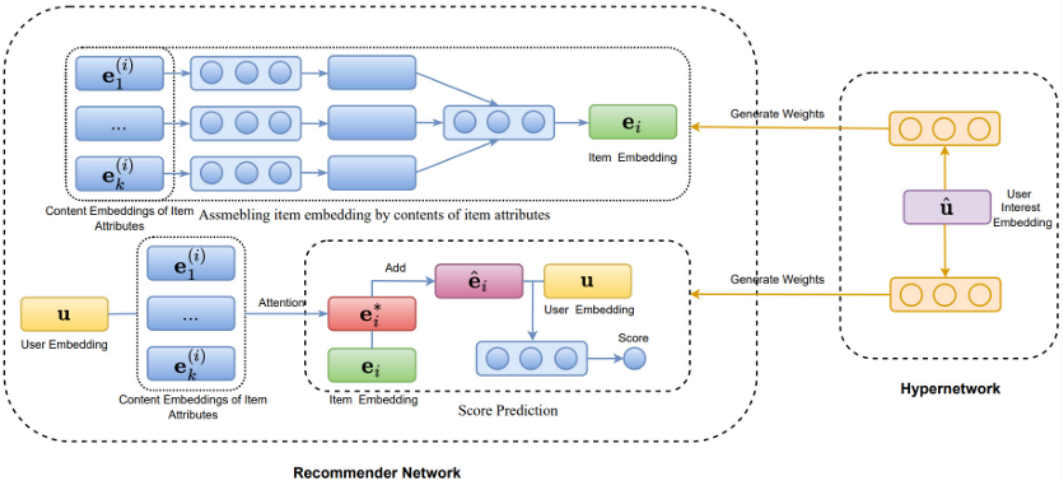

Figure 3.1: *HyperRS architecture (Reproduced from [6])*

$$p(i) = (p_1^{(i)}, p_2^{(i)}, \ldots, p_k^{(i)}) \tag{3.1}$$

The contents of attributes are represented by multi-hot vectors $(\boldsymbol{p_1^{(i)}}, \boldsymbol{p_2^{(i)}}, \ldots, \boldsymbol{p_k^{(i)}})$. Each item attribute content embedding, $\boldsymbol{e_x^{(i)}}$, is formed by the dot product of all attribute x content embeddings $\boldsymbol{M_x}$ and their corresponding multi-hot vectors:

$$\boldsymbol{e_x^{(i)}} = \boldsymbol{M_x p_x^{(i)}}, \quad x = 1, 2, \ldots, k \tag{3.2}$$

The item attribute content embeddings are then transformed by a hidden layer and merged to generate the item embedding $\boldsymbol{e_i}$ as follows:

$$\boldsymbol{\hat{e}_x^{(i)}} = \phi\big(\boldsymbol{W_x e_x^{(i)}} + \boldsymbol{b_x}\big), \quad x = 1, 2, \ldots, k \tag{3.3}$$

$$\boldsymbol{e_i} = \phi\big(\boldsymbol{W_t}[\boldsymbol{\hat{e}_1^{(i)}} \oplus \cdots \oplus \boldsymbol{\hat{e}_k^{(i)}}] + \boldsymbol{b_t}\big) \tag{3.4}$$

where $\boldsymbol{W_x}$, $\boldsymbol{W_t}$, $\boldsymbol{b_x}$, $\boldsymbol{b_t}$ are weights and biases, $\oplus$ denotes the vector concatenation and ϕ is the activation function.

The recommender network learns a user embedding $\boldsymbol{u}$ for each user $u \in U$, where U is the set of users, via an embedding-lookup based on the user's id. To formulate the user interest in item content attributes, the recommender network employs an attention mechanism:

$$\boldsymbol{e_i^*} = attention(\boldsymbol{u}, \boldsymbol{P}) \tag{3.5}$$

where $\boldsymbol{u}$ is the user embedding and $\boldsymbol{P}$ contains item attribute content embeddings $\boldsymbol{e_x^{(i)}}$ as column vectors.

The final item embedding used for prediction is obtained by summing the

attention-based item embedding e_i^* (Equation 3.5) and the naive item embedding e_i (Equation 3.4):

$$\hat{\boldsymbol{e}}_{\boldsymbol{i}} = \boldsymbol{e}_{\boldsymbol{i}}^* + \boldsymbol{e}_{\boldsymbol{i}} \tag{3.6}$$

The item embedding $\hat{\boldsymbol{e}}_{\boldsymbol{i}}$ and the user embedding $\boldsymbol{u}$ are then given as input to a four-layer neural network to produce the final prediction:

$$\hat{s}^{(u,i)} = f_\theta(\hat{\boldsymbol{e}}_{\boldsymbol{i}} \oplus \boldsymbol{u}) \tag{3.7}$$

where $\hat{s}^{(u,i)}$ is the predicted score (rating) that user u will assign to item i and θ represents the parameters of the four-layer neural network. We denote the full parameter set of the recommender network as $\Theta = \{\boldsymbol{W}_{\boldsymbol{x}}, \boldsymbol{b}_{\boldsymbol{x}}, \boldsymbol{W}_{\boldsymbol{t}}, \boldsymbol{b}_{\boldsymbol{t}} | x = 1, 2, \ldots, k\} \cup \theta$.

The hypernetwork's objective is to generate accurate user-specific parameters of the parameter set Θ. To achieve this, it is provided with the user-interest embedding which can be formulated as follows:

$$\hat{\boldsymbol{u}} = \boldsymbol{M}_{\boldsymbol{u}} \boldsymbol{u}^* \tag{3.8}$$

where $\boldsymbol{M}_{\boldsymbol{u}}$ denotes the user interest basis, implemented as a dense layer weight matrix, and $\boldsymbol{u}^*$ is the user embedding computed solely from the user id. $\boldsymbol{M}_{\boldsymbol{u}}$ is a global parameter learned by existing user-item interactions, while $\boldsymbol{u}^*$ is a user-specific parameter used to adapt the global representation to a target user. Specifically, when the model trains on a user's limited interactions (the support set) ,$\boldsymbol{M}_{\boldsymbol{u}}$ remains frozen, forcing the user-specific parameter $\boldsymbol{u}^*$ to adapt by finding the optimal linear combination of the existing basis. $\boldsymbol{M}_{\boldsymbol{u}}$ is instead updated exclusively based on the prediction error in the user's unseen query set. By iteratively aggregating these query-set updates across the entire training population, the optimization process forces M_u to discover and encode generalized, universal latent interests rather than user-specific anomalies. On the other hand, $\boldsymbol{u}^*$ stores the weights of the user interest embeddings which are updated via backpropagation in a target user's limited interactions. This robust global representation ensures that when a new target user arrives, HyperRS only needs to quickly adapt $\boldsymbol{u}^*$ to the target user via backpropagation on their limited interactions, thereby alleviating the cold-start problem.

The hypernetwork consists of a single-layer neural network that takes the user-interest embedding as input and generates user-specific parameters for the recommender network:

$$\theta^* := \phi(\boldsymbol{W}_{\boldsymbol{\theta^*}} \hat{\boldsymbol{u}} + \boldsymbol{b}_{\boldsymbol{\theta^*}}), \quad \theta^* \in \Theta \tag{3.9}$$

3.2.2 Training Procedure

In a cold user setting, the dataset is split in training, validation and test splits with no overlapping users between the splits. During training, each user's ratings are further partitioned into support set S and query set Q, where the support set typically contains a small number k (e.g., 3, 5, 10) of ratings to simulate a user's few interactions in the cold-start scenario. For each user, the hypernetwork initializes the recommender network parameters Θ, based on the user interest embedding derived from the support set. Afterwards, the recommender network generates predictions for each item in the support set, and the user-specific parameters of the recommender network are updated via gradient steps. The updated user-specific parameters are subsequently used by the recommender network to generate predictions for the query set. Finally, the loss between the actual and predicted ratings in the query set is used to update the hypernetwork's parameters Φ. This process trains the hypernetwork to generate accurate user-specific parameters given limited user-item interactions. For completeness, we include the original HyperRS training algorithm (Algorithm 1), as adapted from [6].

In this setup, the hypernetwork functions as the meta-learner, learning how to produce effective user-specific parameters for the recommender network. Each user corresponds to a meta-task (i.e., a few-shot learning episode), with few-shot learning simulated through taking gradient updates on the support set. The calculation of the query set loss and the subsequent update of the hypernetwork's parameters constitute the meta-optimization step.

3.3 HyperRS Extension

Having described the HyperRS framework and training procedure, in the following subsections we introduce the proposed extensions that constitute the main contribution of this work.

3.3.1 P-HyperRS

First, we propose *Plot-HyperRS* (*P-HyperRS*), a HyperRS extension which incorporates movie plot embeddings derived from an SBERT-based model, and consists of two stages; the embeddings generation stage and the model training stage.

In the first stage, we utilized *all-MiniLM-L6-v2* SBERT-based model, available on the HuggingFace platform, to precompute embeddings from the textual summaries of the movie plots. all-MiniLM-L6-v2 is a distilled SBERT language model

Algorithm 1 HyperRS Training Procedure

Require: Support set S and query set Q, where $S, Q \subset \mathcal{D}$
Require: Learning rates α, β
Randomly initialize hypernetwork parameters Φ
for $i = 1$ **to** N **do**
  for each user $u \in \mathcal{U}_{\text{train}}$ **do**
    Initialize recommender parameters Θ using the hypernetwork (Eq. (3.8), (3.9))
    for each $(u, i, s) \in S_u$ **do**
      Predict $\hat{s}(u, i)$ using the recommender network
      $\forall \boldsymbol{\gamma} \in \Theta: \quad \boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma} - \alpha \nabla_{\boldsymbol{\gamma}} \mathcal{L}(s, \hat{s}(u, i))$
      $\boldsymbol{u}^* \leftarrow \boldsymbol{u}^* - \alpha \nabla_{\boldsymbol{u}^*} \mathcal{L}(s, \hat{s}(u, i))$
    end for
    for each $(u, i, s) \in Q_u$ **do**
      Predict $\hat{s}(u, i)$ using the recommender network
      $\forall \gamma \in \Phi: \quad \gamma \leftarrow \gamma - \beta \nabla_{\gamma} \mathcal{L}(s, \hat{s}(u, i))$
      $\boldsymbol{M_x} \leftarrow \boldsymbol{M_x} - \beta \nabla_{\boldsymbol{M_x}} \mathcal{L}(s, \hat{s}(u, i))$
      $\boldsymbol{u} \leftarrow \boldsymbol{u} - \beta \nabla_{\boldsymbol{u}} \mathcal{L}(s, \hat{s}(u, i))$
    end for
  end for
end for=0

[42] fine-tuned on a 1 billion sentence pairs dataset using a contrastive learning objective and consists of 6 transformer blocks (layers) with 12 self-attention heads each and 384 hidden size that maps each sentence to a 384-dimensional vector. Essentially, given a sentence from the pair, contrastive learning aims to predict which sentence out of other sentences was actually paired with the given sentence. To generate the final embedding, the raw text of a movie plot is first tokenized. As this tokens sequence passes through the transformer layers, the model generates a contextualized 384-dimensional vector for each individual token. To collapse these token-level representations into a single, fixed-size vector that represents the entire plot, the model applies a mean pooling operation, which calculates the mathematical average of all the token embeddings. An example of movie plot embedding generation is shown in Figure 3.2. Since the model truncates text inputs with more than 256 tokens, an examination of movie plot tokens was conducted. As shown in Table 3.1, the maximum number of movie plot tokens is 73 with an average of approximately 35 tokens per plot and a minimum of 8 tokens. Additionally, the majority of movie plots (75%) has less than 43 tokens. As a result, there was no movie plot input that was truncated during embedding computation. By precomputing the embeddings on the offline stage, we avoid adding additional computation overhead on the training stage while enriching the recommender's signal when generating the item embeddings.

As shown in Figure 3.3, P-HyperRS builds on HyperRS architecture by adding

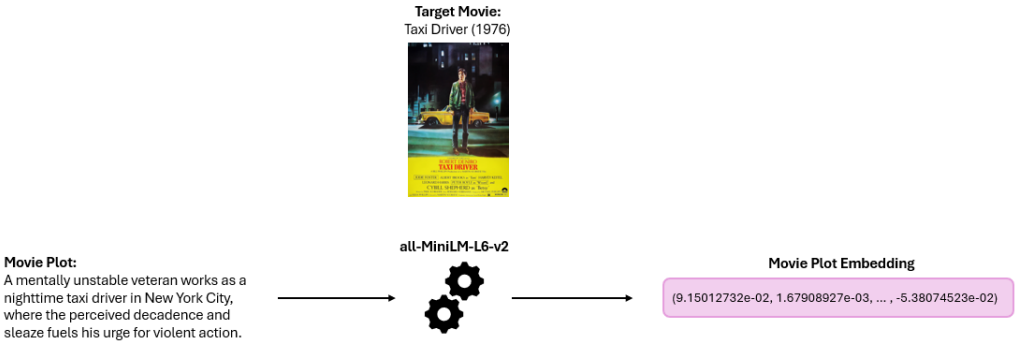

Figure 3.2: *HyperRS architecture (Reproduced from [6])*

	Count of tokens
Minimum	8
Mean	35.0089
75th percentile	43
Maximum	73

Table 3.1: *Statistics of tokens count per plot*

a building block on top of it. Specifically, we gather movie plot embeddings $\boldsymbol{e}_{\boldsymbol{plot}}^{(i)}$ by embedding-lookup of the item's ID and employ a dense layer without bias term to adapt the plot embeddings on the given user-item interactions:

$$\boldsymbol{e}_{\boldsymbol{plot}}^{(i)\prime} = \boldsymbol{e}_{\boldsymbol{plot}}^{(i)} \boldsymbol{W}_{\boldsymbol{a}} \tag{3.10}$$

where $\boldsymbol{W}_{\boldsymbol{a}}$ denote the adapter's layer weights.

A projector layer is employed to project the 384-dimensional concatenation of the precomputed and adapted plot vectors to the model's item embedding size:

$$\boldsymbol{e}_{\boldsymbol{plot}}^{(i)\prime\prime} = \boldsymbol{e}_{\boldsymbol{plot}}^{(i)} + \boldsymbol{e}_{\boldsymbol{plot}}^{(i)\prime} \tag{3.11}$$

$$\boldsymbol{e}_{\boldsymbol{plot}}^{(i)*} = \boldsymbol{e}_{\boldsymbol{plot}}^{(i)\prime\prime} \boldsymbol{W}_{\boldsymbol{p}} \tag{3.12}$$

where $\boldsymbol{W}_{\boldsymbol{p}}$ denotes the projector's layer weights.

The projected plot embeddings $\boldsymbol{e}_{\boldsymbol{plot}}^{(i)*}$ are normalized using the L2-norm in order to prevent embeddings with large magnitudes from exploding gradients during training.

Finally, the normalized plot embeddings $\boldsymbol{e}_{\boldsymbol{plot\_normalized}}^{(i)*}$ is concatenated with $\boldsymbol{e}_{\boldsymbol{i}}$ (Equation 3.4) and $\boldsymbol{P}$ (Equation 3.5). In this way, the plot embeddings are integrated both in the attention mechanism of Equation 3.5 and the final item embedding generation of Equation 3.6, hence enriching the recommender's network item representation signal.

Overall, P-HyperRS extends the HyperRS framework by incorporating SBERT-derived movie plot embeddings into the item representation pipeline, allowing unstructured textual content to be jointly modeled with existing item attributes while maintaining the original meta-learning formulation.

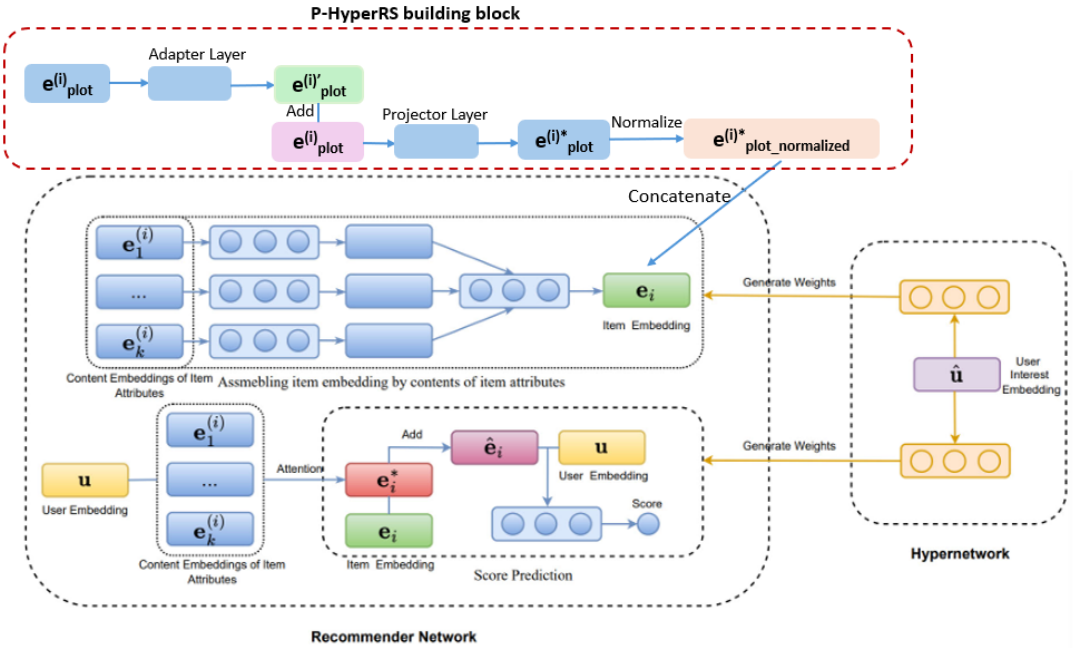

Figure 3.3: *P-HyperRS architecture (Adapted from [6])*

3.3.2 UD-HyperRS

User Demographics-HyperRS (*UD-HyperRS*) constitutes the second HyperRS extension proposed in this study. As its name suggests, it extends HyperRS by integrating user demographic information into the recommender network component. Similarly to P-HyperRS, as shown in Figure, UD-HyperRS builds upon HyperRS existing architecture by incorporating an additional building block on the user embedding generation side.

Particularly, each user attribute is represented by a randomly initialized embedding $\boldsymbol{ue}_y^{(u)}$, which is shared across users with the same attribute values, where u denotes the user id and $y = 1, 2, \ldots, n$ denotes the user attribute. Here, n is the total number of user attributes. The user attribute embeddings are concatenated to form the demographics embedding as follows:

$$\boldsymbol{d}_u = \boldsymbol{ue}_1^{(u)} \oplus \cdots \oplus \boldsymbol{ue}_n^{(u)} \tag{3.13}$$

The demographic embedding is then provided as input to a two-layer neural network, which is used to project them in the model's embedding space. The first

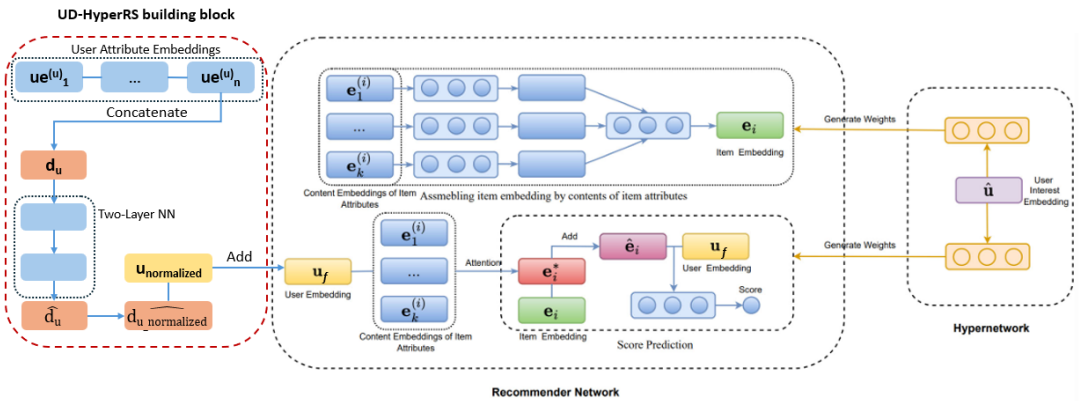

Figure 3.4: *UD-HyperRS architecture (Adapted from [6])*

layer of the neural network introduces non-linearity, thereby enabling the modeling of interactions between user attributes, and is formulated as follows:

$$\boldsymbol{d}_{\boldsymbol{u}}^{*} = \phi(\boldsymbol{W}_{\boldsymbol{p1}}\boldsymbol{d}_{\boldsymbol{u}} + \boldsymbol{b}_{\boldsymbol{p1}}) \tag{3.14}$$

where ϕ denotes the activation function, $\boldsymbol{W}_{\boldsymbol{p1}}$ the layer's weights and $\boldsymbol{b}_{\boldsymbol{p1}}$ the layer's bias.

The second layer of the neural network is responsible for projecting the demographics embedding to the model's embedding size:

$$\hat{\boldsymbol{d}}_{\boldsymbol{u}} = \boldsymbol{W}_{\boldsymbol{p2}}\boldsymbol{d}_{\boldsymbol{u}}^{*} + \boldsymbol{b}_{\boldsymbol{p2}} \tag{3.15}$$

where $\boldsymbol{W}_{\boldsymbol{p2}}$ denotes the weights of the layer and $\boldsymbol{b}_{\boldsymbol{p2}}$ its bias.

The demographic embedding $\hat{\boldsymbol{d}}_{\boldsymbol{u}}$ and the user embedding $\boldsymbol{u}$ are normalized using the L2-norm. The user embedding $\boldsymbol{u}$ is obtained by embedding-lookup of the user ID, following the formulation of the HyperRS architecture.

The final user embedding provided to the recommender network is formed by the element-wise summation of the normalized user embedding $\boldsymbol{u}_{\boldsymbol{normalized}}$ and the normalized demographics embedding $\hat{\boldsymbol{d}}_{\boldsymbol{u\_normalized}}$:

$$\boldsymbol{u}_{\boldsymbol{f}} = \boldsymbol{u}_{\boldsymbol{normalized}} + \hat{\boldsymbol{d}}_{\boldsymbol{u\_normalized}} \tag{3.16}$$

The final user embedding $\boldsymbol{u}_{\boldsymbol{f}}$ is integrated into the recommender network through the attention mechanism (Equation 3.5) and the final prediction provided by the recommender network's four-layer network (Equation 4.2). The corresponding equations are updated accordingly as follows:

$$\boldsymbol{e}_{\boldsymbol{i}}^{*} = attention(\boldsymbol{u}_{\boldsymbol{f}}, \boldsymbol{P}) \tag{3.17}$$

$$\hat{s}^{(u,i)} = f_\theta(\hat{\boldsymbol{e}}_{\boldsymbol{i}} \oplus \boldsymbol{u}_{\boldsymbol{f}}) \tag{3.18}$$

In general, UD-HyperRS enhances the user signal in the recommender network by incorporating demographic embeddings, rather than relying solely on embedding lookup of the user ID to form the user embedding. User attribute embeddings are randomly initialized and used to provide a structured initialization that serves as a starting point for generating more personalized user embeddings. Furthermore, the utilization of the two-layer neural network allows the model to refine these initial representation's projection based on user-item interactions during backpropagation.

3.3.3 PUD-HyperRS

Plot and User Demographics-HyperRS (*PUD-HyperRS*) extends HyperRS model by integrating both SBERT-derived plot embeddings in the item embedding generation procedure and user demographic information in the user embedding generation procedure. Essentially, as shown in Figure 3.5, PUD-HyperRS combines the building blocks of *P-HyperRS* and *UD-HyperRS* in order to simultaneously enhance both user and item representation signals. Since the two building blocks are architecturally separable but jointly optimized within the same meta-learning framework, PUD-HyperRS is built by simply integrating P-HyperRS and UD-HyperRS components, as described in sections 3.3.1 and 3.3.2, on top of HyperRS architecture.

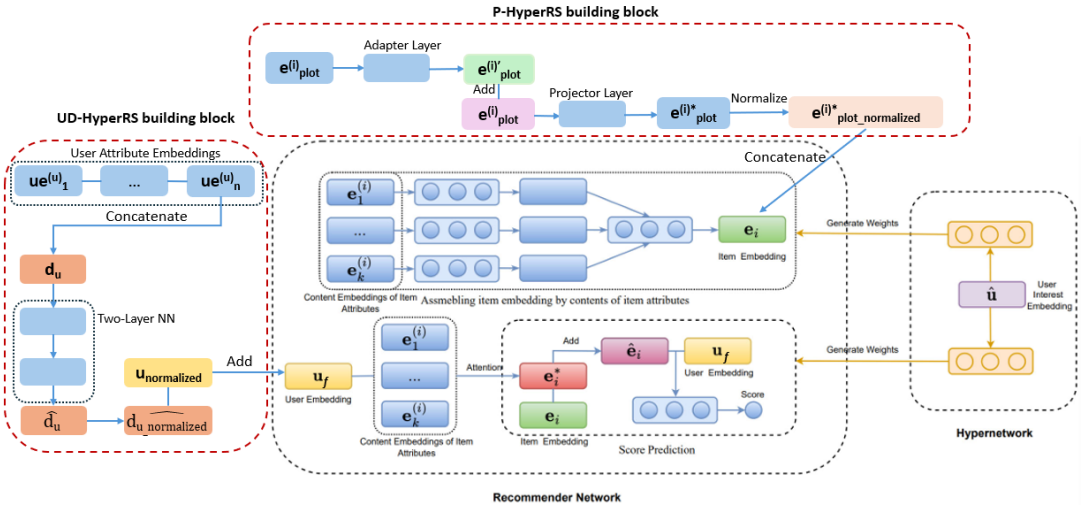

Figure 3.5: *PUD-HyperRS architecture (Adapted from [6])*

Chapter 4

Research Methodology

This chapter presents the research methodology including the dataset description, data collection procedure, data analysis, the evaluation setup and the working environment utilized for running the experiments.

4.1 Research Design

To evaluate the performance of the proposed HyperRS extension, we conducted experiments in the MovieLens-1M dataset [15], which contains approximately 1 million ratings of 3800 movies from thousands of MovieLens users. Because MovieLens-1M does not contain all the necessary item features, we utilized OMDB API [16] to extract additional movie features. A detailed description of the existing dataset and the data collection procedure is presented in the following section. All experiments were conducted in a Virtual Machine (VM) running on Amazon Web Services Elastic Compute Cloud environment (AWS EC2). The VM ran on Ubuntu 20.04 LTS operating system with 4 virtual CPUs (vCPUs), 16 GB memory, 1 NVIDIA T4 GPU with 16 GB video memory (VRAM), and 50 GB disk storage.

4.2 Data Collection and Analysis

The MovieLens-1M dataset contains 1,000,209 anonymous ratings of 3,883 movies made by 6,040 MovieLens users. Ratings are integers on a scale of 1-5 with no intervals. The dataset includes the following item and user features:

- *Item features*:
  - *Title*: Movie title, including year of release
  - *Year of release*: Year of release extracted from the movie title

- *Genres*: At least one genre out of eighteen possible genres.

- *User features*:
  - *Gender*: User gender (Female or Male)
  - *Age*: Age group
  - *Occupation*: User's occupation id, selected from a list of twenty encoded occupations
  - *Zip-code*: City of residence zip-code

In order to properly prepare the dataset for the model, preprocessing techniques were applied. In particular, the *year of release* feature was discretized into fourteen groups, *genres* were multi-hot encoded, and *gender* and *age* were assigned unique ids.

Furthermore, the dataset was enriched with movie plots obtained via the OMDb API [16], which retrieves data from IMDb using IMDb movie IDs. Since MovieLens-1M does not provide IMDb IDs, we utilized MovieLens-20M dataset, which contains movie identifiers for various external sources, including IMDb website. Mutual movies between the two datasets were identified and their IMDb IDs were used to retrieve their plots through the API (Figure 4.1). As shown in Table 4.1, the IMDb ID and plot were successfully retrieved for the majority of 3,883 movies, with only 26 movies having no IMDb ID match. Additionally, only 18 identified movies had null plots on the IMDb website. For both unmatched and null-plot movies, we treated the plot as missing.

Figure 4.1: *Data collection workflow*

	Count of Movies	% of Total Movies
Valid Plots	3839	98.87%
NA Plots	18	0.47%
Unmatched IDs	26	0.66%
Total Movies	**3883**	

Table 4.1: *Movie plots retrieval results*

The MovieLens-1M dataset provides users with sufficient interactions, thus enabling experimentation with different support and query set splits to simulate

cold-start settings. Specifically, each user has rated at least 20 movies, with an average of approximately 192 ratings per user and a maximum number of 2,314 movies (Table 4.2) . As shown in Figure 4.2, the vast majority of users (approximately 4,700 users) has rated between 20 and 249 movies.

	Total ratings by users
Minimum	20
Mean	192.47
Maximum	2314

Table 4.2: *Total ratings by users statistics*

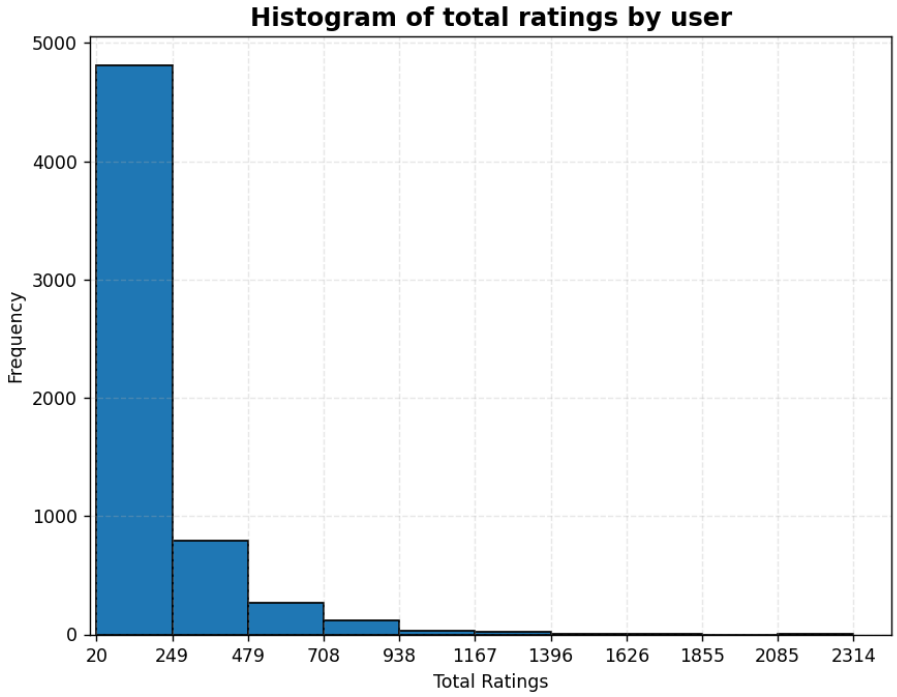

Figure 4.2: *Histogram of total ratings by user*

4.3 Evaluation Setup

To evaluate the performance of the proposed HyperRS extensions, the following models were trained[1]:

- *vanilla HyperRS*: Baseline HyperRS model utilizing only basic item features.

[1]The implementation of the baseline HyperRS model and the foundational architecture for our proposed variants (P-HyperRS, UD-HyperRS, PUD-HyperRS) are based on the original source code provided by the authors of the HyperRS paper [6]. The original source code is publicly available on GitHub [43].

- *P-HyperRS*: HyperRS extension that incorporates SBERT-derived movie plot embeddings to enrich item features.
- *UD-HyperRS*: HyperRS extension that incorporates user demographics information to enrich user features.
- *PUD-HyperRS*: HyperRS extension that combines both item and user feature enrichment.

Each model was provided with different types of side information as input data, as summarized in Table 4.3, in order to evaluate HyperRS performance under each condition. Moreover, we fine-tuned each model's hyperparameters by testing combinations in a non-exhaustive search due to training time constraints. The hyperparameters tuned were as follows:

- *Embedding Size*: Embedding size of user and item embeddings used in the recommender network. Values tested: {64, 96, 100, 128}.
- *User Interest Basis Embedding Size*: Embedding size of user interest basis used in the hypernetwork. Values tested: {64, 128}.
- *User Interest Embedding Size*: Embedding size of user interest used in the hypernetwork. Values tested: {128, 256}
- *α*: Learning rate of the recommender network. Values tested: {1×10^{-3}, 3×10^{-3}, 5×10^{-3}, 5×10^{-4}}
- *β*: Learning rate of the hypernetwork. Values tested: {1×10^{-4}, 5×10^{-4} }
- *Epochs*: Number of training epochs. Values tested: {5, 10, 15, 20}

The hyperparameter tuning for each model is shown in Table 4.4. All experiments were conducted under a cold user-warm item setting. The dataset was split into training, validation and test sets in a 60%-20%-20% ratio. In order to ensure the cold user state, the splits were made with no overlapping users between them. Furthermore, we established the warm item state by including in the validation and test sets only the items which also appeared in the training set. For each user in each split, we randomly retain only 20 user-item interactions (ratings), which were then divided into the support set S and the query set Q, as described in Section 3.2.2. The performance of HyperRS extensions was also evaluated under different user cold-start settings by changing the support set size $|S|$, which simulates the user's few known interactions, and the remaining query set size $|Q|$, which simulates the first new user interactions in the system. Specifically, experiments were conducted with $|S| = |Q| = 10$ and $|S| = 5, |Q| = 15$. In addition, we

randomly selected 20 items that each user had not interacted with and assigned them a zero rating to form negative samples. Negative samples were added to S and Q during training.

Since our objective is to provide users with a ranked list of Top-N recommendations rather than predicting their exact rating scores, we evaluated the recommendation quality using the *binary normalized discounted cumulative gain* (*NDCG*), as presented in [6], as the evaluation metric. $NDCG@N$ is a widely used metric in information retrieval and recommender systems because it measures the quality of a ranking by considering both the relevance of the recommended items and their ranking. It operates on the premise that highly relevant items should appear earlier in the recommendation list, applying a logarithmic discount to penalize relevant items that are placed lower in the ranking. Its values range from 0 to 1, with 1 indicating the ideal ranking–where the top-N recommendations contain only relevant items–and 0 indicating the worst possible ranking, where no relevant items appear in the top-N recommendations. Specifically, the binary $NDCG@N$ is computed as:

$$DCG@N(\boldsymbol{p}) = \sum_{i=1}^{N} \frac{2^{\mathbf{1}(\boldsymbol{p}_i \in \boldsymbol{t})} - 1}{log_2(i+1)} \tag{4.1}$$

$$NDCG@N(\boldsymbol{p}) = \frac{DCG@N(\boldsymbol{p})}{DCG@N(\boldsymbol{t})} \tag{4.2}$$

Here, $\boldsymbol{p}$ is set that defines the top-N item recommendations based on the model's highest predicted scores for a specific user, while $\boldsymbol{t}$ comprises the set of the actual items the user has historically interacted with. The term $\mathbf{1}(s)$ represents a binary indicator function, yielding 1 when the statement s is valid and 0 when it is false. The final $NDCG@N$ for a dataset D is computed as the average $NDCG@N$ of all users $U(D)$ in the dataset:

$$NDCG@N(D) = \frac{1}{|U(D)|} \sum_{u \in U(D)} NDCG@N(\boldsymbol{p}(u), \boldsymbol{t}(u)) \tag{4.3}$$

where $\boldsymbol{p}(u)$ is the items with the top-N highest predicted rating to user U, and $\boldsymbol{t}(u)$ is the set of items interacted by the user.

	Item Features			User Features			
Model	**Genres**	**Year**	**Movie Plot**	**Gender**	**Age**	**Occupation**	**Zip-Code**
Vanilla HyperRS	✓	✓	×	×	×	×	×
P-HyperRS	✓	✓	✓	×	×	×	×
UD-HyperRS	✓	✓	×	✓	✓	✓	✓
PUD-HyperRS	✓	✓	✓	✓	✓	✓	✓

Table 4.3: *Side-information configuration of evaluated models*

Model	**Embedding Size**	**User Inter. Basis Embed. Size**	**User Inter. Embed. Size**	**Alpha**	**Beta**	**Epochs**
vanilla HyperRS	64	64	128	0.003	0.0005	10
P-HyperRS	64	64	128	0.003	0.0005	15
UD-HyperRS	64	64	128	0.003	0.0005	15
PUD-HyperRS	64	64	128	0.003	0.0005	15

Table 4.4: *Hyperparameters configuration of evaluated models*

Chapter 5

Results

This chapter presents the experimental results derived from the research methodology described in Chapter 4. The results are organized and supported by detailed figures and tables. It should be noted that the interpretation and justification of the results are not included in this chapter.

5.1 Experimental Results

During the experimental procedure, the models' hyperparameters were fine-tuned with respect to the binary NDCG@N performance metric. The selected hyperparameter configuration is as follows: user/item embedding size of 64, user interest basis embedding size of 64, user interest embedding size of 128, recommender network learning rate α equal to 1×10^{-3} and hypernetwork learning rate β equal to 5×10^{-4}. The vanilla HyperRS was trained for 10 epochs, while the HyperRS extensions P-HyperRS, UD-HyperRS, and PUD-Hypers were trained for 15 epochs.

A summary of the models' evaluation results with a configured support set size $|S| = 10$ and query set size $|Q| = 10$ is presented in Table 5.1. The values of $\Delta(NDCG@10)_{vanilla}$% represent the percentage change in $NDCG@10$ for each model with respect to the $NDCG@10$ score of the vanilla HyperRS model. A visual representation of the evaluation results is shown in Figure 5.1. Similarly, the evaluation results for support set size $|S| = 5$ and query set size $|Q| = 15$ are presented in Table 5.2, while their corresponding visual representation is shown in Figure 5.2. Finally, a comparison between the HyperRS models and related work models is presented in Table 5.3.

[1] AdaML [4] uses a different evaluation protocol and is not directly comparable to HyperRS results.

[2] ColdGAN [2] constitutes a GAN-based model and utilizes a different training procedure compared to meta-learning methods.

Model	NDCG@5	NDCG@10	$\Delta(NDCG@10)_{vanilla}$%
Vanilla HyperRS	0.1234	0.1871	0.00
P-HyperRS	**0.1249**	**0.1922**	**0.51**
UD-HyperRS	0.1199	0.1842	-0.29
PUD-HyperRS	0.1170	0.1793	-0.78

Table 5.1: *Models evaluation results for* $|S| = 10$, $|Q| = 10$

Model	NDCG@5	NDCG@10	$\Delta(NDCG@10)_{vanilla}$%
Vanilla HyperRS	**0.1049**	**0.1620**	**0.00**
P-HyperRS	0.0871	0.1350	-2.70
UD-HyperRS	0.0976	0.1510	-1.10
PUD-HyperRS	0.0951	0.1398	-2.22

Table 5.2: *Models evaluation results for support size* $|S| = 5$, $|Q| = 15$

Model	NDCG@5
Vanilla HyperRS	0.1234
P-HyperRS	0.1249
UD-HyperRS	0.1199
PUD-HyperRS	0.1170
AdaML[1]	**0.8491**
ColdGAN [2]	0.1124

Table 5.3: *HyperRS comparison with related work's models*

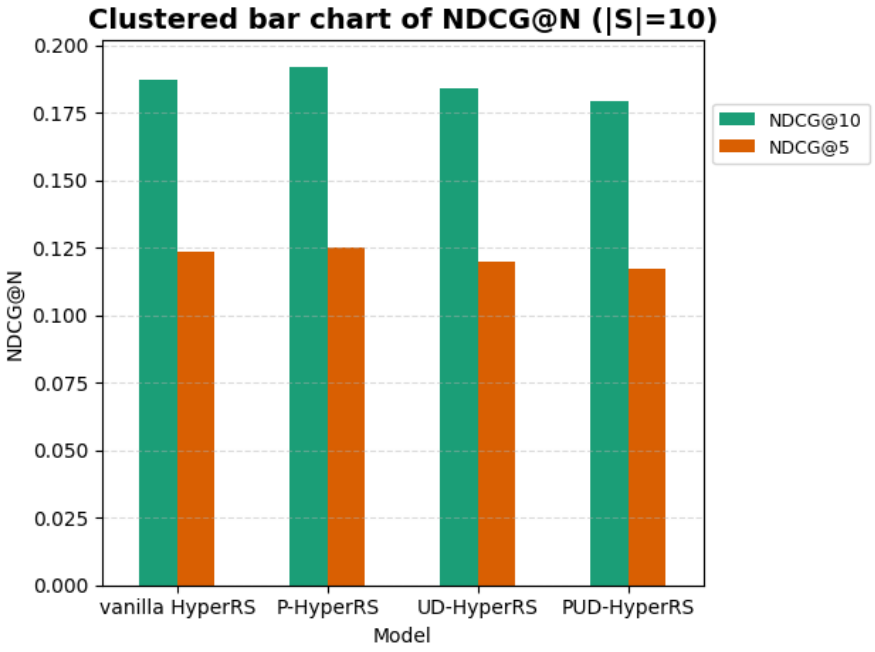

Figure 5.1: *Clustered bar chart of NDCG@N for support size* $|S| = 10$, $|Q| = 10$

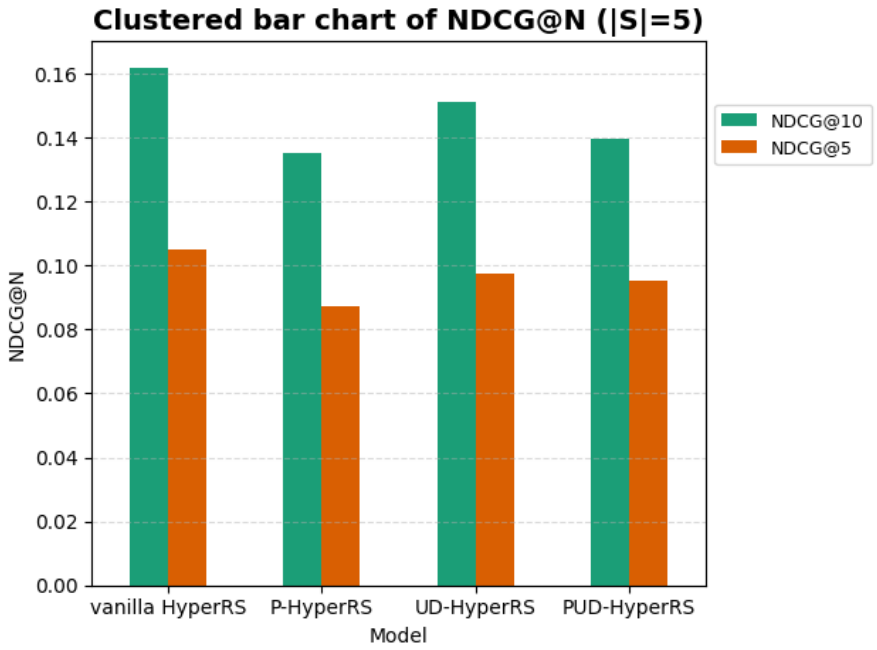

Figure 5.2: *Clustered bar chart of NDCG@N for support size* $|S| = 5$, $|Q| = 15$

Chapter 6

Discussion

This chapter presents the interpretation of the experimental results presented in the previous chapter, addresses the research questions of this study, and investigates the main limitations of this study.

6.1 Interpretation of Findings

The experimental results reveal several noteworthy findings regarding the capabilities of the HyperRS framework when multimodal side information is available. As thoroughly described in the previous chapters, the HyperRS framework was extended in three different ways: (i) through integrating SBERT-derived movie plot embeddings, (ii) through integrating user demographic data and (iii) by combining both types of side information in the same architecture. To provide a structured and well-justified interpretation of the findings, the discussion is organized into four main directions: evaluation under a moderate user cold-start setting, evaluation under a strict user cold-start setting, comparison with models from the existing literature and interpretation with respect to the study's research questions.

Under a moderate user cold-start setting, where the support set size simulating the user's known interactions was set to $|S| = 10$, the P-HyperRS model achieved the best performance. Specifically, as shown in Table 5.1, P-HyperRS achieved an $NDCG@10$ score of 0.1922, compared to 0.1871 for vanilla HyperRS, 0.1842 for UD-HyperRS and 0.1793 for PUD-HyperRS. This result indicates that incorporating precomputed SBERT-derived movie plot embeddings into the item representation pipeline allows the model to capture more fine-grained item characteristics, thereby improving the modeling of user preference over item attributes. Although movie plot embeddings are semantically rich and encode contextual movie information, the observed performance improvement is marginal ($+0.51\%$) compared to the vanilla HyperRS model. Since movie plot embeddings are concate-

nated with item atrribute embeddings, which are optimized through user-item interactions, their additional contribution may be partially redundant. In addition, the marginal performance improvement may indicate that the information contained in movie plot embeddings overlaps with features already captured by structured attributes such as genres and year of release. In other words, movie plots may implicitly reflect movie genres and temporal information. In contrast, integrating user demographic data slightly decreases performance. Specifically, UD-HyperRS exhibits a -0.29% decrease in $NDCG@10$ compared to vanilla HyperRS, suggesting that incorporating additional user features may introduce noise or redundancy in the learned user representation when sufficient interaction data are available. Finally, the integration of movie plot embeddings and user demographics in the same architecture in PUD-HyperRS has the least positive impact in model performance, with a -0.78% decrease (0.1793 compared to 0.1871) in $NDCG@10$ score relative to the baseline model.

The evaluation of HyperRS extensions under a stricter user cold-start setting yielded slightly different results, as shown in Table 5.2. In particular, when the support set size $|S|$ was reduced to 5, the model that achieved the highest $NDCG@10$ was the baseline model, vanilla HyperRS, with a score of 0.1620. In that case, P-HyperRS demonstrated the lowest performance (0.1350) corresponding to a percentage change of $NDCG@10$ equal to -2.70%. This result suggests that when HyperRS is provided with limited user-item interaction, it cannot effectively leverage movie plot embeddings. As previously discussed, item attribute embeddings (such as genres and year of release) are optimized through user-item interactions and movie plot embeddings acts as an auxiliary item representation signal. When limited user-item interactions are available, integrating movie plot embeddings may possibly introduce noise or misalignment, consequently reducing the ranking quality. Furthermore, UD-HyperRS achieved the second-best performance with a $NDCG@10$ score of 0.1510 and the smallest percentage decrease (-1.10%) relative to the baseline, compared to PUD-HyperRS(-2.22%) and P-HyperRS (-2.70%). This observation implies that the integration of user side information may be more beneficial than contextual item side information when user-item interactions are extremely limited. Nevertheless, UD-HyperRS still underperforms vanilla HyperRS, suggesting that while demographics may provide slight regularization when interaction data are scarce, they lack the expressive capacity to model nuanced user-item affinity. In other words, this result may imply that demographic attributes provide low-resolution preference signals that may not align with fine-grained item representations learned from interactions.

The comparison between the HyperRS-based models and related approaches from the literature, summarized in Table 5.3, provides additional insights. The

AdaML meta-learning model [4] achieved the highest $NDCG@5$ score (0.8491) significantly outperforming the other models. However, AdaML follows a different evaluation protocol, in which selected cold users have between 15 and 100 user-item interactions with a query set size $|Q| = 10$ and the remaining interactions forming the support set. As described in Section 4.2, users in the MovieLens-1M dataset have at least 20 ratings, with the majority of having up to 200 ratings. Therefore, AdaML evaluation results are not directly comparable to those of HyperRS. Moreover, all HyperRS models outperform ColdGAN [2] by margins ranging from 0.46% (PUD-HyperRS) to 1.25% (P-HyperRS). Although, it should be noted that ColdGAN constitutes a GAN-based model rather than a meta-learning model and follows a fundamentally different training procedure than meta-learning frameworks.

6.2 Addressing the Research Questions

After interpreting the experimental results and deriving the key findings, this section explicitly addresses the research questions of this study.

RQ1: *How do different types of side information affect HyperRS performance?* The motivation for integrating SBERT-derived movie plot embeddings was to enable the model to capture structured textual content in conjunction with existing categorical item attributes, such as genres, and to improve the item representation signal. However, movie plot embeddings provide only marginal performance improvements under moderate cold-start setting, while they degrade model's performance when provided with limited user-item interactions. Regarding the integration of user demographic data, the experimental results show that demographics-enhanced HyperRS consistently underperforms the baseline HyperRS model. These results suggest that available user representation data provide a relatively weak signal. However, under the strict cold-start setting, user demographics perform better than movie plot embeddings, implying that they are slightly more robust than item side information in such setting. Finally, combining both movie plots and user demographics results in a performance decrease in both cold-start settings. Overall, SBERT-enhanced item-side information provides limited improvement and is sensitive to the availability of interaction data, while user side information appears relatively redundant, introducing noise and misalignment during training. These findings suggest that the HyperRS framework is highly optimized to efficiently model interaction-based item attributes, thus limiting the effect of additional signals.

RQ2: *How do different cold-start settings affect HyperRS performance?* As

expected, configuring different support set sizes yields different HyperRS performance levels. Specifically, all HyperRS models' performance decreases when evaluated under the strict cold-start setting ($|S| = 5$), compared to the moderate cold-start setting ($|S| = 10$). This result suggests that the effectiveness of additional side-information decreases as cold-start severity increases. In addition, the vanilla HyperRS model proves to be the most stable under the strict cold-start setting, showing the smallest performance loss relative to its moderate cold-start performance. This observation may imply that the HyperRS model is capable of efficiently modeling user-item interactions when limited item attributes and interaction data are available. In general, the findings demonstrate that HyperRS is highly sensitive to cold-start settings. As user interaction become scarce, the framework struggles to exploit additional side information, suggesting that interaction-based signals remain fundamental even in meta-learning frameworks.

6.3 Study Limitations

Despite the main contributions of this study, several important limitations should be clearly stated. First, the proposed approach evaluation is limited to a single dataset (MovieLens-1M), rendering the study specific to the movie recommendation domain. Second, the integration of user side information relies on limited demographic data rather than richer behavioral or temporal data, such as viewing history sequences and clicks behavior. Third, even though different user cold-start scenarios were evaluated (5 and 10 cold-start user known interactions), it is common in real-world applications that the number of known item interactions for a new user is often even more limited (e.g., 3 or even 1). Finally, the performance evaluation was conducted solely with respect to the $NDCG@N$ metric, without incorporating additional metrics that could provide better model interpretability, such as attention scores over item attributes within the attention mechanism.

Chapter 7

Conclusion and Future Work

The last chapter of this study concludes on the problem this study investigates. In particular, the chapter briefly restates the research problem, the motivation for this study, the proposed approach, the data collection procedure, the key findings of the experimental results and the addressing of the research questions formulated. Additionally, future research directions are proposed to address the limitations of the study based on research outcomes.

7.1 Conclusion

This study examines the cold-start problem in recommender systems, where the system is required to generate effective personalized recommendations for a new user whose interaction history is limited or even non-existent.

Specifically, this work examines the extension of the meta-learning hypernetwork-based model, termed HyperRS, through the incorporation of user and item side information. Three different HyperRS extensions are proposed. The first model, namely P-HyperRS, constitutes an extension of the original HyperRS model that incorporates SBERT-derived movie plot embedding through an additional building block added on top of HyperRS meta-learning framework. The second model, UD-HyperRS, integrates user demographic information through an embedding layer followed by a two-layer neural network built on top of HyperRS architecture. Finally, a combination of P-HyperRS and UD-HyperRS building blocks, termed PUD-HyperRS, is proposed in order to simultaneously integrate both movie plot embeddings and user demographic information.

The main contribution of this study lies in the systematic extension and evaluation of the HyperRS meta-learning framework through the integration of heterogeneous side information under different user cold-start settings. Consequently, the research questions addressed are the following: *"How do different types of side*

information affect HyperRS performance?" (*RQ1*) and *"How do different cold-start settings affect HyperRS performance?"* (*RQ2*).

The proposed HyperRS extensions are evaluated under varying user cold-start settings and compared to the baseline model, vanilla HyperRS, which utilizes only item attributes (genres and year of release) as side information. The experimental results indicate that SBERT-enhance item side information provides marginal performance gains and is sensitive to user cold-start state, while user side-information appear relatively redundant and introduce noise during training. Furthermore, the findings suggest that the cold-start severity significantly affects model performance, since when user-interaction data are limited, the framework struggles to exploit additional side information.

Despite the study's contribution and insights, the findings are subject to certain limitations. The evaluation is conducted exclusively on a movie-domain dataset and the available user side information is limited to basic demographic attributes. As a consequence, the genealizability of the results to other domains or richer user contexts remain uncertain. Moreover, the evaluation methodology is restricted to cold-start setting that may not reflect real-world cold-start settings. Also, the performance evaluation solely relies on the NDCG@N metric and does not incorporate additional metrics towards better model explainability.

Based on the experimental findings and identified limitations, several directions for future work emerge. Future research could evaluate the proposed extensions performance in other domains, such as music or e-commerce, explore the integration of user behavioral data such as clicks behavior and viewing history sequences, and evaluate performance on stricter cold-start settings to provide deeper insights into model adaptability. Finally, incorporating additional metrics, such as attention scores over item attributes within the attention mechanism, could provide insights into model's internal mechanisms and enhance its explainability.

Bibliography

[1] Charu C. Aggarwal. *Recommender Systems: The Textbook*. Springer Cham, 1 edition, 2016. eBook, XXI + 498 pages, 61 b/w illustrations, 18 color illustrations.

[2] Po-Lin Lai, Chih-Yun Chen, Liang-Wei Lo, and Chien-Chin Chen. Coldgan: Resolving cold start user recommendation by using generative adversarial networks, 2020.

[3] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. Melu: Meta-learned user preference estimator for cold-start recommendation, 2019.

[4] Jia Xu, Hongming Zhang, Xin Wang, and Pin Lv. Adaml: An adaptive meta-learning model based on user relevance for user cold-start recommendation. *Knowledge-Based Systems*, 279:110925, 2023.

[5] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[6] Yuxun Lu, Kosuke Nakamura, and Ryutaro Ichise. Hyperrs: Hypernetwork-based recommender system for the user cold-start problem. *IEEE Access*, PP:1–1, 01 2023.

[7] Hongli Yuan and Alexander Hernandez. User cold start problem in recommendation systems: A systematic review. *IEEE Access*, 12 2023.

[8] Yufeng Duan and Ryosuke Saga. Apparel goods recommender system based on image shape features extracted by a cnn. pages 2365–2369, 10 2018.

[9] Florian Strub, Jérémie Mary, and Romaric Gaudel. Hybrid recommender system based on autoencoders. 06 2016.

[10] Dimitrios Rafailidis and Alexandros Nanopoulos. Modeling users preference dynamics and side information in recommender systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(6):782–792, 2016.

[11] Debashish Roy and Chen Ding. Movie recommendation using youtube movie trailer data as the side information. In *Proceedings of the 12th IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '20, page 275–279. IEEE Press, 2021.

[12] Hong Xu. Enhancing recommender systems with nlp-based biased singular value decomposition. In *2023 3rd International Symposium on Computer Technology and Information Science (ISCTIS)*, pages 808–811, 2023.

[13] S. Sindhu Priya, A. Deepa Madhuri Reddy, G. Aagnesh, S. Srinivasreddy, K. Shalini, and Shanmugasundaram Hariharan. Collaborative filtering based hotel recommendation system using nlp. In *2025 5th International Conference on Pervasive Computing and Social Networking (ICPCSN)*, pages 1323–1327, 2025.

[14] Jamallah M. Zawia, Maizatul Akmar Binti Ismail, Mohammad Imran, Buce Trias Hanggara, Diva Kurnianingtyas, Silvi Asna, and Quang Tran Minh. Comprehensive review of meta-learning methods for cold-start issue in recommendation systems. *IEEE Access*, 13:24622–24641, 2025.

[15] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015.

[16] OMDB API. OMDB API: The open movie database. https://www.omdbapi.com/, 2025. Accessed: 02-12-2025.

[17] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, December 1992.

[18] Qazi Mohammad Areeb, Mohammad Nadeem, Shahab Saquib Sohail, Raza Imam, Faiyaz Doctor, Yassine Himeur, Amir Hussain, and Abbes Amira. Filter bubbles in recommender systems: Fact or fallacy – a systematic review, 2023.

[19] Hung Vinh Tran, Tong Chen, Guanhua Ye, Quoc Viet Hung Nguyen, Kai Zheng, and Hongzhi Yin. On-device content-based recommendation with single-shot embedding pruning: A cooperative game perspective. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, page 772–785, New York, NY, USA, 2025. Association for Computing Machinery.

[20] Qijiong Liu, Nuo Chen, Tetsuya Sakai, and Xiao-Ming Wu. Once: Boosting content-based recommendation with both open- and closed-source large language models. In *Proceedings of the 17th ACM International Conference on*

Web Search and Data Mining, WSDM '24, page 452–461, New York, NY, USA, 2024. Association for Computing Machinery.

[21] Noa Tuval, Alain Hertz, and Tsvi Kuflik. Addressing the cold start problem in privacy preserving content-based recommender systems using hypercube graphs, 2023.

[22] Alain Hertz, Tsvi Kuflik, and Noa Tuval. Resolving sets and integer programs for recommender systems. *Journal of Global Optimization*, 81:1–26, 09 2021.

[23] Xiaoyuan Su and Taghi Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. Artificial Intellegence*, 2009, 10 2009.

[24] Zhongxiang Sun, Zihua Si, Xiaoxue Zang, Kai Zheng, Yang Song, Xiao Zhang, and Jun Xu. Large language models enhanced collaborative filtering. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, page 2178–2188, New York, NY, USA, 2024. Association for Computing Machinery.

[25] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[26] Lianghao Xia, Meiyan Xie, Yong Xu, and Chao Huang. Mixrec: Heterogeneous graph collaborative filtering. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, WSDM '25, page 136–145, New York, NY, USA, 2025. Association for Computing Machinery.

[27] Jiahao Liu, Dongsheng Li, Hansu Gu, Peng Zhang, Tun Lu, Li Shang, and Ning Gu. Unbiased collaborative filtering with fair sampling. In *Proceedings*

of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '25, page 2555–2559, New York, NY, USA, 2025. Association for Computing Machinery.

[28] Fan Liu, Shuai Zhao, Zhiyong Cheng, Liqiang Nie, and Mohan Kankanhalli. Cluster-based graph collaborative filtering. *ACM Trans. Inf. Syst.*, 42(6), October 2024.

[29] Badrul Sarwar, George Karypis, and John Riedl. Application of dimensionality reduction in recommender system – a case study. 08 2000.

[30] Yaru Jin, Shoubin Dong, Yong Cai, and Jinlong Hu. Racrec: Review aware cross-domain recommendation for fully-cold-start user. *IEEE Access*, 8:55032–55041, 2020.

[31] Bin Hu, Yinghong Ma, Zhiyuan Liu, and Hong Wang. A social importance and category enhanced cold-start user recommendation system. *Expert Systems with Applications*, 277:127130, 2025.

[32] Hisham Bawa and Achala Aponso. Addressing the cold start problem of recommendation systems: A comprehensive review of traditional and emerging solutions. In *2025 10th International Conference on Information and Network Technologies (ICINT)*, pages 41–46, 2025.

[33] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys*, 52(1):1–38, February 2019.

[34] Gregory R. Koch. Siamese neural networks for one-shot image recognition. 2015.

[35] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning, 2017.

[36] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning, 2017.

[37] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

[38] Yujia Zheng, Siyi Liu, Zekun Li, and Shu Wu. Cold-start sequential recommendation via meta learner, 2020.

[39] Juhua Pu, Yuanhong Wang, Fang Nan, and Xingwu Liu. Adamo: Adaptive meta-optimization for cold-start recommendation. *Neurocomputing*, 580:127417, 2024.

[40] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[42] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

[43] Yuxun Lu, Kosuke Nakamura, and Ryutaro Ichise. Hyperrs: Hypernetwork-based recommender system for the user cold-start problem [source code]. https://github.com/ichise-laboratory/hyperrs, 2023.