Bachelor's thesis

Batchelor's Programme in Computer Science

Active learning strategies to combat the new user cold start problem in collaborative filtering

Mikael Hokkanen

January 8, 2024

Faculty of Science

University of Helsinki

Contact information

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki,Finland

Email address: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Study programme
Faculty of Science	Batchelor's Programme in Computer Science

Tekijä — Författare — Author

Mikael Hokkanen

Työn nimi — Arbetets titel — Title

Active learning strategies to combat the new user cold start problem in collaborative filtering

Ohjaajat — Handledare — Supervisors

Williams Moreno, Bernardo; Mustonen, Ville

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Bachelor's thesis	January 8, 2024	22 pages

Tiivistelmä — Referat — Abstract

Recommender systems play a crucial role in helping users navigate the vast array of options offered by online services, offering personalized recommendations based on individual preferences. However, these systems often struggle to provide accurate recommendations for new users who lack sufficient historical data. This challenge, known as the new user cold start problem, is particularly prominent in collaborative filtering, which heavily relies on historical user ratings.

To address this issue, active learning strategies have emerged as a solution, wherein users are directly asked to rate items in order to acquire initial ratings. The gist of active learning is identifying criteria for obtaining data that better reflect users' preferences. However, selecting the most informative items while minimizing user effort remains a non-trivial task.

This work explores the use of active learning strategies to combat the new user cold start problem in collaborative filtering. It first establishes the necessary background on collaborative filtering, the new user cold start problem, and active learning. Then, personalized and non-personalized active learning methods are compared through a comprehensive analysis of various experiments. Finally, the findings from these experiments provide practical guidelines for evaluating and designing effective active learning strategies to address the new user cold start problem, accounting for potential biases. Most notably, the significance of personalization is demonstrated by incorporating matrix factorization into a decision tree-based interview process, effectively tailoring each question and integrating the recommendation model.

ACM Computing Classification System (CCS)
Information systems → Information retrieval → Retrieval tasks and goals → Recommender systems
Computing methodologies → Machine learning → Learning settings → Active learning settings

Avainsanat — Nyckelord — Keywords

recommender systems, collaborative filtering, active learning, new user cold start

Säilytyspaikka — Förvaringsställe — Where deposited

Helsinki University Library

Muita tietoja — övriga uppgifter — Additional information

Uuden käyttäjän kylmäkäynnistysongelman torjunta aktiivisen oppimisen menetelmillä yhteistoiminnallisessa suodatuksessa

Tämä tutkielma käsittelee aktiivisen oppimisen (engl. *active learning*) strategioita uuden käyttäjän kylmäkäynnistysongelman (engl. *new user cold start problem*) ratkaisemiseksi yhteistoiminnallisessa suodatuksessa (eng *collaborative filtering*). Luvussa 2 tutustutaan suosittelujärjestelmiin ja niiden perinteiseen luokitteluun. Tämän jälkeen luvussa 3 läpikäydään yhteistoiminnallisen suodatuksen perusteet, ja esitellään matrix factorization pääsuosittelumetodina. Luvuissa 4 ja 5 keskitytään uuden käyttäjän "kylmäkäynnistysongelmaan" ja aktiivisen oppimisen yhteyteen yhteistoiminnalliseen suodatukseen. Luvuissa 6 ja 7 arvioidaan ei-personoituja ja personoituja aktiivisen oppimisen strategioita kokeellisen analyysin kautta. Luku 8 tiivistää tärkeimmät havainnot ja antaa käytännön suuntaviivoja tehokkaiden aktiivisen oppimisen strategioiden suunnitteluun uuden käyttäjän ongelman torjumiseksi.

Suosittelujärjestelmät (engl. *recommender systems*) auttavat ja ohjaavat käyttäjiä navigoimaan lukuisia vaihtoehtoja online-palveluja selatessaan. Niiden alkuperäinen tarve nousi 1900-luvun lopussa, kun sähköisen kaupankäynnin palvelut halusivat luoda yksilöllisen kokemuksen käyttäjille, jotka kärsivät tiedon ylikuormituksesta (Linden et al., 2003). Suosittelujärjestelmien päätehtävä on tuottaa yksilöllisiä ja merkityksillisiä tarjouksia käyttäjille tuotteista jotka vastaavat heidän mieltymyksiään (Melville and Sindhwani, 2017). Ne siis suodattavat tietoa rajaamalla ja korostamalla kohteita, jotka se löytää sopiviksi. Näitä järjestelmiä käytetään laajalti erilaisissa verkkopalveluissa, kuten verkkokaupoissa, suoratoistopalveluissa ja sosiaalisessa mediassa (Linden et al., 2003).

Suosittelujärjestelmien käyttämä data voi perustua eksplisiittiseen dataan, kuten kohteiden ominaisuuksiin, tai implisiittiseen dataan, kuten käyttäjän ja kohteen väliseen vuorovaikutukseen. Esimerkiksi musiikkikappaletta voidaan luonnehtia eksplisiittisesti sen genren, artistin ja musikaalisten ominaisuuksien perusteella numeraalisesti. Toisaalta kappaletta

voidaan myös luonnehtia ja mitata implisiittisesti sen käyttäjäarvioiden perusteella asteikolla 1-5, tai vaikka laskemalla kuinka monta kertaa käyttäjä on kuunnellut kyseistä kappaletta. Suosittelujärjestelmät käyttävät näitä tiedonlähteitä identifioimaan samankaltaisuuksia käyttäjä- ja kohdeprofiileiden välillä tarjotakseen sopivia suosituksia (Melville and Sindhwani, 2017). Itse suositustekniikka valitaan datan tyypin perusteella. Yleisiin suositustekniikkoihin lukeutuvat koneoppiminen, tietojen louhinta ja tiedonhaku (Elngar, 2020).

Perinteisesti suosittelujärjestelmät ovat jaoteltu kahteen luokkaan: sisältöpohjainen suodatus (engl. *content-based filtering*), ja yhteistoiminnallinen suodatus (Ricci et al., 2010; Elngar, 2020). Sen lisäksi sekamuotoisella suodatuksella viitataan näiden kahden lähestymistavan yhdistelmään (Melville and Sindhwani, 2017). Sisältöpohjainen suodatus oppii suosittelemaan etsimällä samankaltaisia kohteita joista käyttäjä on pitänyt aiemmin, tai sovittamalla kohteita käytäjäprofiilin ominaisuuksiin (Ricci et al., 2010). Sisältöpohjaisessa suodatuksessa jokaiselle käyttäjälle ja kohteelle on asetettu ennalta-määritettyjä attribuutteja, joita suosittelujärjestelmä vertailee keskenään. Yhteistoiminnallinen suodatus taas oppii suosittelemaan kohteita etsimällä käyttäjiä jotka ovat pitäneet samankaltaisista kohteista kuin kohdekäyttäjä (Ricci et al., 2010). Toisin kuin sisältöpohjaisessa suodatuksessa, yhteistoiminnalliset suodatusmenetelmät eivät perustu käyttäjien ja tuotteiden yhdistämiseen; sen sijaan ne havaitsevat merkityksellisiä suhteita käyttäjien välillä, mikä mahdollistaa aktiivisen muutosten tunnistamisen käyttäjien mieltymyksissä. (Aggarwal, 2016). Yhteistoiminnallinen suodatus kärsii huomattavasti enemmän kylmäkäynnistysongelmasta, jossa uusille käyttäjille on vaikea suositella kohteita, tai uusia kohteita on vaikea suositella käyttäjille vähäisen datan takia. Sisältöpohjainen suodatus pystyy huomattavasti helpommin kategorisoimaan uusia kohteita ja käyttäjiä ennalta-määritettyjen attribuuttien ansiosta, kun taas yhteistoiminnallinen suodatus vaatii käyttäjän arvioivan tarpeeksi kohteita ennen kuin sen voi tehdä tarkkoja suosituksia (Elahi et al., 2016). Tästä syystä tämän työn painopiste on ensisijaisesti yhteistoiminnallisessa suodatuksessa.

Yhteistoiminnallisen suodatus voidaan jakaa edelleen kahteen osaan: *muistipohjaisiin* (engl. *memory-based*) ja *mallipohjaisiin* (engl. *model-based*) järjestelmiin (Melville and Sindhwani, 2017). Muistipohjaiset järjestelmät pyrkivät löytämään naapuruston käyttäjiä, jotka vastaavat kohdekäyttäjää, tai samankaltaisia kohteita joille käyttäjä on antanut korkeat arviot (Bobadilla et al., 2013; Aggarwal, 2016). Mallipohjaiset järjestelmät taas arvioivat tilastollisten mallien paramametrejä käyttäjän arvostelujen perusteella hyödyntäen koneppimisen ja tiedon louhinnan tekniikoita. Verrattuna muistipohjaisiin järjestelmiin, mallipohjaiset järjestelmät ovat skaalautuvampia, käsittelevät paremmin niukkaa

dataa (engl. *sparse data*), ja ovat usein tarkempia tarjoamaan suosituksia (Ricci et al., 2010).

Tämän työn pääasiallisena suodatusmenetelmänä käytetään matrix factorization-mallia, joka toimii pohjimmaisena suosittelujärjestelmänä kaikissa keskusteltavissa tutkimuksissa. Se on mallipohjaisen yhteistoiminnallisen suodatuksen malli, jonka tavoitteena on ennustaa käyttäjien arviointeja hajoittamalla alkuperäinen arviointimatriisi kahteen käyttäjiä ja kohteita vastaaviin matalamman ulottuvuuden matriisiiseihin (Ricci et al., 2010; Elahi et al., 2016; Koren et al., 2009). Malli pyrkii tunnistamaan käyttäjien ja kohteiden piileviä ominaisuuksia (engl. *latent feature*) aikaisempien arvioiden perusteella. Esimerkiksi elokuvan piileviä ominaisuuksia voisivat olla elokuvan pituus, genre, tai vaikkapa elokuvan väriteema. Matrix factorization nousi suosioon voitettuaan suoratoistopalvelu Netflixin järjestämän “The Netflix Prize"-kilpailun, jonka tavoitteena oli parantaa alustan suosittelujärjestelmän tarkkuutta kymmenellä prosentilla (Koren et al., 2009).

Yhteistoiminnallisen suodatuksen suorituskykyä mitataan useimmiten offline-tilassa, jossa valmiiksi kerätty tietojoukko on jaoteltu harjoitussettiin (engl. *training set*) ja testaussettiin (engl. *testing set*). Suositusten laatua mitataan usein tarkkuudella, joka mittaa erotusta ennustetun arvostelun ja todellisen arvostelun välillä (Ricci et al., 2010; Melville and Sindhwani, 2017). Toisaalta online-tutkimuksissa kohteen käyttöä pidetään joskus tehokkuuden kannalta toivottavampana tekijänä kuin itse tarkkuutta. Esimerkiksi elokuvien suosittelussa suositus katsotaan onnistuneeksi, jos käyttäjä katsoo elokuvan suosituksen jälkeen. Kohteen käyttöä voidaan mitata täsmällisyydellä (engl. *precision*), joka mittaa suosittelulistan lopullisessa järjestyksessä olevien relevanttien kohteiden osuutta. Lisäksi palautuskyky (engl. *recall*) mittaa relevanttien tuotteiden osuutta kaikista relevanteista tuotteista (Ricci et al., 2010).

Eräs yhteistoiminnaliisen suodatuksen suurimmista ongelmista on aiemmin mainittu uuden käyttäjän kylmäkäynnistysongelma (engl. *the new user cold start problem*), joka ilmenee tilanteissa joissa järjestelmä ei ole kerännyt tarpeeksi dataa tehdäkseen tarkkoja suosituksia (Elahi et al., 2016). Perinteisten yhteistoiminnallisen suodatumenetelmien on vaikea profiloida uusia käyttäjiä tarkasti, johtaen vaillinaisiin suosituksiin. Käyttäjien tulisi itsenäisesti löytää ja arvioida tarpeeksi kohteita, kunnes järjestelmä voisi aloittaa tehokkaan suosittelun. Yleisin tapa ratkaista kylmäkäynnistysongelma on hyödyntää lisätietoja, joita perinteinen suositusmenetelmä ei kykene keräämän (Jesús et al., 2012). Lisätietoja voidaan joko kerätä suoraan kysymällä käyttäjää arvioimaan kohteita, tai epäsuoralla tiedonkeruulla ilman käyttäjän tietämystä. Tämä tutkielma keskittyy jatkossa ni-

menomaan suoriin menetelmiin tiedon hankkimiseksi, eikä epäsuoria menetelmiä käsitellä enää.

Kuten mainittu, yksi tapa selvittää käyttäjän mieltymyksiä varhaisessa vaiheessa on kysyä häntä suoraan arvioimaan listan kohteita. Tämä lähes ilmeinen strategia, jota kutsutaan aktiiviseksi oppimiseksi (engl. *active learning*), on osoittanut parantavan merkittävästi suositusten laatua uusille käyttäjille (Elahi et al., 2016; A. Rashid et al., 2002; Karimi et al., 2013; Zhou et al., 2011; Golbandi et al., 2011; Geurts et al., 2020). Se on koneoppimisen tekniikka, joka aktiivisesti pyrkii keräämään mahdollisimman laadukkaita datapisteitä minimoidakseen häviöfunktion (engl. *loss function*) (Melville and Sindhwani, 2017; Gope and Jain, 2017). Uusien käyttäjien astuessa sisään järjestelmä muodostaa haastatteluprosessin, jossa esitetään huolellisesti valittu joukko kohteita jonkin heuristiikan perusteella, kuten kohteiden suosio (A. Rashid et al., 2002; Elahi et al., 2016). Aktiivisen oppimisen päätavoitteena on vaalia mahdollisimman paljon tietoa käyttäjästä ja samalla minimoida kysymysten määrä. Aktiivisen oppimisen strategiat voidaan luokitella ei-personoituihin (engl. *non-personalized*) ja personoituihin (engl. *personalized*) menetelmiin. Ei-personoidut menetelmät tarjoavat kaikille uusille käyttäjille saman listan kohteita, riippumatta heidän aikaisemmista vastauksistaan. Personoidut menetelmät taas ottavat huomioon käyttäjän aikaisemmat vastaukset ja muokkaavat kyselyä sen mukaisesti (A. Rashid et al., 2002). Kummatkin luokat voidaan edelleen jakaa yksiheuristisiin (engl. single-heuristic) menetelmiin, jotka käyttävät yksittäistä kriteeriä valitakseen kohteita, tai moniheuristisiin (engl. multiple-heuristic) menetelmiin, jotka hyödyntävät monta eri strategiaa (Elahi et al., 2016).

A. Rashid et al. (2002) and Geurts et al. (2020) tutkivat uuden käyttäjän kylmäkäynnistysongelmaa vertaillen yksiheuristisissa ja moniheuristisia ei-personoituja aktiivisen oppimisen strategioita. He huomasivat moniheuristisien menetelmien menestyvän parhaiten, kun heuristiikoiksi valittiin kohteen suosio, ja entropia (engl. *entropy*) tai virhe (engl. *error*). Valitsemalla kohteet suosion perusteella kohteet olivat tuttuja suurimmalle osalle käyttäjistä, jättäen välistä vastaamattomat arviot. Sen lisäksi entropia tai virhe kasvattivat kohteiden tiedonarvoa, sillä ne kutakuinkin valitsevat kohteita, jotka jakavat eniten mielipiteitä, jolloin käyttäjät ovat helpompi erotella. Ei-personoitujen menetelmien arvioimisessa on myös otettava huomioon haastatteluprosessin pituus, eli kohteiden määrä kyselyssä. Tutkimuksissa huomattiin, kuinka kysymysten määrä vaikutti parhaiten suoriutuvaan strategiaan (Geurts et al., 2020). Todellisessa tilanteessa käyttäjä ei usein ole valmis vastaamaan kovin moneen kysymykseen, joten haastatteluprosessin toteutuksessa

on otettava huomioon tasapaino käyttäjän panoksen ja suositusten tarkkuuden välillä.

Personoituja menetelmiä tutkittin rakentamalla haastatteluprosessi päätöspuilla (engl. *decision-trees*, joka valitsevat seuraavat kysymykset käyttäjän aikaisempien vastausten perusteella Golbandi et al. (2011). Lisäksi, Zhou et al. (2011) and Karimi et al. (2013) yhdistivät päätöspuu-algoritmiin matrix factorization-mallin kahdella eri tavalla. Tulosten mukaan päätöspuu-pohjaiset personoidut menetelmät onnistuvat keräämän paremmin arviointeja kuin ei-personoidut menetelmät, ja samalla valitsemaan informatiivisempiä kohteita, johtaen parempiin suosituksiin. Toisaalta ne ovat laskennallisesti vaativampia ja toteutuksellisesti vaikeita. Yhdistämällä matrix factorization -malli päätöspuihin voidaan tehokkaasti integroida suositusmalli alkuhaastatteluun ja tunnistaa piileviä ominaisuuksia käyttäjistä jo varhaisessa vaiheessa. Tämä mahdollistaa käyttäjien tehokkaamman luokittelun ja suositusten tarjoamisen alusta asti.

Tulokset osittavat, että aktiivisen oppimisen strategiat toimivat tehokkaana ratkaisuna uuden käyttäjän kylmäkäynnistysongelmaan. Tutkimukset myös korostavat käyttäjien tyypillistä subjektiivisuutta. Ei-personoidut menetelmät eivät ota huomioon käyttäjän kohdemakua, päätöksentekoa, tai kohdetietoisuutta, kun taas personoidut menetelmät pystyvät kategorisoimaan käyttäjän jo muutaman vastauksen jälkeen (Koren et al., 2009; Geurts et al., 2020). Ei-personoidut metodit ovat helppo toteuttaa, mutta vaativat huomattavasti enemmän vastauksia saavuttaakseen personoitujen menetelmien tarkkuuden (A. Rashid et al., 2002). Toisaalta ei-personoidut menetelmät valitsevat kohteet jo ennen haastattelua, jolloin järjestelmän ei tarvitse ottaa huomioon uusien kohteiden laskemista. Jokaisen tutkimuksen yksi olennaisista ongelmista oli strategioiden online-testauksen puute. Offline-kokeet voivat johtaa helposti vinoumiin tuloksissa, koska ne olettavat, että kaikki järjestelmän valitsemat ja ehdottamat kohteet ovat käyttäjälle jo ennalta tiedossa (Karimi et al., 2013; A. Rashid et al., 2002). Strategioiden testaaminen vakiintuneessa suosittelujärjestelmässä varmistaisi niiden todellisen tehokkuuden.

Contents

1 Introduction

In today's digital landscape, users often find themselves overwhelmed by the number of options offered by online services. The vast magnitude of choices poses a challenge for both users and providers, as presenting all suitable options to users becomes an impractical task. Recommender systems have emerged to tackle these issues, aiming to assist users in navigating the overwhelming range of options by providing personalized recommendations based on their preferences (Ricci et al., 2010; Elngar, 2020).

However, recommender systems, especially collaborative filtering, faces a significant challenge when a new user joins the system with little to no historical data. Without sufficient information, the system struggles to make meaningful inferences to make accurate recommendations. This problem, known as the new user cold-start problem, has been extensively studied in collaborative filtering (Bobadilla et al., 2013; A. M. Rashid et al., 2008).

One way to elicit user preferences early on is to directly ask the user to rate items. This, almost an obvious strategy called *active learning*, has been shown to significantly improve recommendations for new users (Elahi et al., 2016). By querying the user a series of questions through an initial interview, the missing item ratings can be collected. However, choosing items that return most information but require minimal user effort is not a trivial task. The current state-of-art in active learning has demonstrated the effectiveness of personalization through the use of decision trees and incorporation of matrix factorization in the interview process (Golbandi et al., 2011; Karimi et al., 2013; Zhou et al., 2011). On the other hand, if one chooses to go with a cheaper non-personalized strategy, they should combine multiple heuristics that aim to collect many ratings while simultaneously preserving their relative informativeness (A. Rashid et al., 2002; Geurts et al., 2020).

The structure of this work is as follows: Chapter 2 introduces recommender systems and their traditional classification. Chapter 3 provides an overview of collaborative filtering and matrix factorization as foundational concepts, to discuss their connection to the new user problem and active learning in chapters 4 and 5, respectively. In chapters 6 and 7, non-personalized and personalized active learning strategies are evaluated through experimental analysis. Chapter 8 summarizes the main findings and suggests practical guidelines in the design of effective active learning strategies to combat the new user problem.

2 Background

This chapter introduces recommender systems generally, and briefly evaluate the most common recommendation methods.

2.1 Recommender systems

In today's digital world, customers are represented with an increasing range of products, while at the same time, the service might find it hard to make personalized offers for each customer (Melville and Sindhwani, 2017). Making a decision becomes harder as the number of options increases. To prevent information overload, the service must serve an efficient but personalized experience. Recommender systems have become the main tool for helping individuals who lack competence in evaluating a large number of options, and at the same time help the service provider to create a personalized experience for each user (Ricci et al., 2010). Originally started by the demand for e-commerce services around the end of the20$^{\text{th}}$, they have become a vital component of almost any online service (Linden et al., 2003).

"Recommender systems are information processing systems that actively gather and process data from various sources" (Ricci et al., 2010). Their main goal is to generate relevant and meaningful recommendations to a collection of users for items and products that match their preference (Melville and Sindhwani, 2017). Efficient recommender systems help users to make deciding actions faster so they do not leave the platform. Therefore, recommender systems' main function is to choose whether an item is worth recommending. The value provided by the recommender system to the user can be in the form of diversifying taste, extracting the most relevant items, finding a specific item, or gaining a deeper insight into a specific domain of items, etc. (Ricci et al., 2010).

In simple terms, recommendations are provided as a ranked list of items, starting with the most suitable item as determined by the system. For example, the items offered could include movies from a streaming service or research papers from a digital library. The data used by recommender systems can be explicit, implicit, or a combination of both (Ricci et al., 2010). Explicit data refers to quantifiable properties of users and items, such as the genre, artist, or musical properties of a song. Implicit data, on the other hand, is gathered

from user-item interactions, such as the song's rating on a scale of 1 to 5 or the number of times a user has listened to it. By utilizing these data sources, recommender systems identify similarities between user and item profiles to offer well-matched recommendations (Melville and Sindhwani, 2017). The choice of a suitable recommendation method depends on the type of data, and commonly employed methods include machine learning, data mining, and information retrieval (Elngar, 2020).

The item ratings are stored in matrices where each row corresponds to the ratings of a single user. Unrated items are commonly undefined or set to some number outside the regular rating scale, such as $\emptyset$ when the scale is from 1 to 5 (Bobadilla et al., 2013).

	Spiderman	Life of Brian	Taxi Driver	Inception
Bob	4	3	5	$\emptyset$
Liisa	$\emptyset$	$\emptyset$	5	4
Pekka	5	5	$\emptyset$	2
Cindy	2	$\emptyset$	4	$\emptyset$

Table 2.1: An example of a movie rating matrix where unrated items are undefined.

2.2 Recommendation methods

Traditional recommender system taxonomy distinguishes recommendation methods into two broader classes: content-based filtering and collaborative filtering (Ricci et al., 2010; Elngar, 2020). In addition, hybrid filtering will be used to refer to the combination of these two approaches (Melville and Sindhwani, 2017). Additional filtering classes, such as demographic and social filtering have also been introduced as the research in the field has developed further (Bobadilla et al., 2013). However, this paper will maintain adherence towards the traditional definition of classes.

Content-based filtering learns to recommend by finding similar items that the user has liked in the past, or by matching items to the user profile's attributes (Ricci et al., 2010; Elngar, 2020). The recommendation process involves calculating similarities between items' predefined attributes to find the most suitable matches against the user profile. The attributes can include any content possessed by the item such as text, sound, and images.

Collaborative filtering methods have remained "as the most popular and widely implemented recommender systems over the years" (Ricci et al., 2010). Instead of comparing

items, they learn to recommend by finding users that have liked similar content as the target user (Ricci et al., 2010). Based on the idea that users with highly correlated item ratings share similar tastes, unrated items are recommended to the target user based on the rating history of similar users (Aggarwal, 2016). Unlike content-based filtering, collaborative filtering methods do not require detailed item- and user profiles; instead, they detect meaningful relationships between users, allowing them to actively identify changes in users' preferences. Collaborative filtering suffers from the new user cold start problem, as it requires initial actions from the user before making accurate recommendations, leading to poor suggestions for new users. On the other hand, content-based filtering suffers significantly less from the new user problem as users can be immediately described by their attributes upon entering the system (Ricci et al., 2010). Therefore, the focus of this work will primarily lay on collaborative filtering.

Lastly, *hybrid recommender* systems use a combination of content-based filtering and collaborative filtering to overcome the disadvantages of one method alone (Ricci et al., 2010; Elngar, 2020; Aggarwal, 2016). Hybrid recommendation is especially useful when the original method suffers from the cold start problem. For example, item-based models in collaborative filtering can be equipped with content-based filtering to make initial classifications on the user domain for an item (Ricci et al., 2010). However, using multiple recommendation methods comes with an increased computational cost.

3 Collaborative filtering

This chapter discusses collaborative filtering in more detail, and how the Netflix Prize competition played a significant role in the development of model-based approaches. Then, matrix factorization is introduced as the primary method for collaborative filtering in this work.

As mentioned, a collaborative filtering recommender system produces user-specific recommendations of items based on patterns of ratings without the need for pre-defined attributes about either items or users (Ricci et al., 2010). The ratings are stored in a $m \times n$ matrix where m is the number of users and n is the number of items. The rows of the matrix represent users' ratings and the columns the items' ratings. The rating r_{ui} of user u of item i is stored in the row u and column i.

$$R = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{21} & r_{22} & \dots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \dots & r_{mn} \end{bmatrix} \tag{3.1}$$

3.1 Collaborative filtering methods

Collaborative filtering methods can be further divided into *memory-based* and *model-based* approaches (Melville and Sindhwani, 2017). *Memory-based* approaches rely on user-item ratings stored in the system to make predictions. They aim to find a neighborhood of users similar to the target user or items similar to the target user's previously rated items in order to make recommendations (Bobadilla et al., 2013; Aggarwal, 2016). These methods are simple and computationally efficient, but they heavily rely on the ratings provided by users and struggle with sparse data. On the other hand, *model-based* approaches estimate parameters of statistical models for user ratings using machine learning and data mining techniques (Melville and Sindhwani, 2017). They are more scalable, handle sparse data better, and often provide higher prediction accuracy compared to memory-based methods (Ricci et al., 2010). Regardless, model-based approaches are still prone to the cold start problem. Examples of model-based collaborative filtering include decision trees, matrix

factorization, and neural networks. However, model-based approaches require a more expensive training phase and involve a trade-off between scalability and efficiency.

3.2 Evaluation of collaborative filtering methods

Collaborative filtering methods are popularly evaluated offline as conducting online experiments is often impractical as they require implementing all the proposed strategies on an established online platform (Karimi et al., 2013). In offline evaluation, a pre-collected dataset is split into a training set to train the model, and the remaining testing set is used to make the predictions. The most common metric for evaluating offline performance is accuracy, which measures the difference between the predicted rating and the actual rating (Ricci et al., 2010; Melville and Sindhwani, 2017). Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are commonly used to measure accuracy. To calculate the accuracy, the model generates predicted ratings $\hat{r}_{ui}$ for a test set $\mathcal{T}$ of user-item pairs (u, i) for which the true ratings r_{ui} are known. The actual ratings for the test set are known because they are hidden in an offline experiment, or they are gathered through a user study or online experiment. The RMSE between predicted and actual ratings is given by:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} (\hat{r}_{ui} - r_{ui})^2}$$

Another popular alternative measure is the Mean Absolute Error (MAE):

$$\text{MAE} = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} |\hat{r}_{ui} - r_{ui}|}$$

MAE favors systems that by magnitude make larger errors but are more consistent, whereas RMSE disproportionately penalizes larger errors (Ricci et al., 2010).

	Recommended	Not Recommended
Used	True-positive (tp)	False-Negative (fn)
Not used	False-Positive (fp)	True-Negative (tn)

Table 3.1: Recommendation outcome classification table.

In an online setting, the usage of the item can be a more desired factor in measuring the effectiveness of the model (Ricci et al., 2010). For example, in movie recommendation,

the recommendation is considered successful if the user will watch the movie after the recommendation. Offline experiments predict the usage by hiding some of the items and asking the recommender system to predict a set of items that the user will use.

The most common ranking metrics are precision and recall, and can be calculated by the count of each item classified by the table 3.1:

$$\begin{aligned} \text{Precision} &= \frac{\#tp}{\#tp + \#fp} \\ \text{Recall (True Positive Rate)} &= \frac{\#tp}{\#tp + \#fn} \end{aligned}$$

Precision measures the fraction of relevant items retrieved in the final ranking. Furthermore, recall measures the fraction of relevant items retrieved out of all relevant items (Elngar, 2020). There is typically a trade-off between precision and recall. A larger number of recommendations typically improves recall while reducing precision (Ricci et al., 2010).

3.3 The Netflix Prize competition

In 2006, the streaming platform Netflix launched a competition called "The Netflix Prize" to improve their recommendation system algorithm's accuracy by 10% with a prize pool of $1 million (Ricci et al., 2010). They released a huge dataset of over 100 million ratings from their users on around 17,000 movies. For the first time, researchers were able to train their models on a large-scale real-world dataset and evaluate the performance on the same given metrics. The competition played a crucial role in advancing the field of collaborative filtering, especially matrix factorization, which eventually won the grand prize in 2009 (Koren et al., 2009).

Netflix Prize competition did not deal with the cold start situation that matrix factorization suffers from (Koren et al., 2009). The competition aimed to improve the overall recommendation accuracy, and it did not specifically consider the predictions for users who had only given a few ratings. In the real world, new users constantly enter the system, giving possibly a skewed impression of the actual performance of the model.

3.4 Matrix factorization

This section presents the basics of matrix factorization that will be used as the base recommendation method for the experiments regarding the new user cold start problem.

Matrix factorization is a model-based collaborative filtering method that learns a vector of latent factors for each user and item from item rating patterns (Koren et al., 2009; Elahi et al., 2016). The goal is to decompose the original rating matrix $R \in \mathbb{R}^{\text{users} \times \text{items}}$ into two lower dimension matrices $Q \in \mathbb{R}^{users \times latentfactors}$ and $P \in \mathbb{R}^{\text{latentfactors} \times \text{items}}$, such that the product $Q^T P$ approximates the original matrix and predicts the missing ratings (Ricci et al., 2010; Elahi et al., 2016):

$$R \approx Q^T P \tag{3.2}$$

The formal definition for the basic matrix factorization model Singular Value Decomposition (SVD), is as follows (Koren et al., 2009; Elahi et al., 2016; Ricci et al., 2010):

Matrix factorization models map both users and items to a joint latent factor space of dimensionality d, such that user-item interactions are modeled as inner products in that space. Each item i is associated with a vector $q_i \in \mathbb{R}^d$, and each user u is associated with a vector $p_u \in \mathbb{R}^f$. For a given item i, the elements of q_i reflect how well the item possesses a particular factor, positive or negative. For a given user u, the elements of p_u measure the extent of interest the user has in items that are high on the corresponding factors, again positive or negative. The resulting dot product, $q_i^T p_u$, captures the interaction between user u and item i-the user's overall interest in the item's characteristics. Since the observed interactions can also be partially explained by the effect of the corresponding user or item, global bias μ, user bias b_u, and item bias b_i are included. This approximates user u 's rating of item i, which is denoted by r_{ui}, leading to the estimate.

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{3.3}$$

As the system learns the model by fitting previously observed ratings, it should avoid overfitting the observed data by regularizing the learned parameters (Koren et al., 2009). The model parameters (b_u, b_i, p_u and q_i) can be found by minimizing the regularized squared error:

$$\min_{p,q,b} \sum_{(u,i)\in K} \left(a_{ui} - \mu - b_u - b_i - p_u^T q_i\right)^2 + \lambda_1 \left(b_u^2 + b_i^2\right) + \lambda_2 \left(\|p_u\|^2 + \|q_i\|^2\right) \tag{3.4}$$

Where $\mathcal{K} = \{(u,i) \mid r_{ui} \text{ is known}\}$. The regularization constant for bias term λ_1, and the regularization constant for the latent factors λ_2 can be tuned by cross-validation.

The equation 3.4 can be solved with alternating least squares or stochastic gradient descent, which the latter has become a more popular method (Koren et al., 2009). Stochastic gradient descent loops through all ratings in the training set and lowers the computation per iteration. Error for each observation in the training set is computed as follows:

$$\varepsilon_{ui}^{\text{pred}} = a_{ui} - \hat{a}_{ui}^{*} \tag{3.5}$$

Accordingly, b_u, b_i, p_u, and q_i are updated using the following rules,

$$\begin{aligned}
b_u &\leftarrow b_u + \gamma_1 \left(\epsilon_{ui}^{\text{pred}} - \lambda_1 \cdot b_u\right), \\
b_i &\leftarrow b_i + \gamma_1 \left(\epsilon_{ui}^{\text{pred}} - \lambda_1 \cdot b_i\right), \\
p_u &\leftarrow p_u + \gamma_2 \left(\epsilon_{ui}^{\text{pred}} \cdot q_i - \lambda_2 \cdot p_u\right), \\
q_i &\leftarrow q_i + \gamma_2 \left(\epsilon_{ui}^{\text{pred}} \cdot p_u - \lambda_2 \cdot q_i\right),
\end{aligned}$$

where γ_1 and γ_2 are the learning parameters. After finding the parameters and computing the mapping for each item and user, the model can easily approximate the user rating for any given item.

Matrix factorization has become widely popular since the Netflix Prize competition due to its performance and scalability (Koren et al., 2009). It is flexible as it can use both explicit and implicit feedback to understand user preferences. On the other hand, regular SVD matrix factorization tends to suffer from sparsity in the user-item rating matrix. There tends to be a high portion of missing values as users only rate a small portion of the items on the platform, increasing the chance of overfitting. To combat the sparsity problem, more advanced models, such as SVD++, make additional use of memory-based approaches and implicit data to capture user preferences regardless of the item ratings (Ricci et al., 2010).

4 The new user cold start problem in collaborative filtering

In this chapter, the new user cold start problem and its significance in collaborative filtering is discussed briefly.

The new user cold-start problem is a common challenge in recommender systems where the system has not yet gathered enough data to make accurate predictions (Elahi et al., 2016). This issue has gained significant attention in the research of collaborative filtering. Traditional collaborative filtering methods struggle to accurately profile users with limited ratings, resulting in inadequate recommendations (Koren et al., 2009). When a new user is initiated into the system, they need to make some number of actions before the system can identify their preferences (Ricci et al., 2010). If early recommendations fail to satisfy the user, they may abandon the service. To address this, additional techniques have been developed to gather valuable data with minimal user effort.

	Green Speaker	Fast Motorcycle	Ripe Banana	Long Shorts
Dijkstra	4	∅	5	4
Lovelace	5	5	∅	2
Torvalds	3	5	4	2
Turing	∅	∅	∅	∅

Table 4.1: Rating matrix when a new user (Turing) enters the system. Turing's row contains only undefined values

There is a wide variety of solutions proposed for the new user cold start problem. The most common way is to utilize additional information that is not traditionally used by the recommendation method (Jesús et al., 2012). The additional information can be collected either explicitly by directly querying the user or implicitly without the user's knowledge. In addition, external sources such as social networks and demographical data can be used for collecting information about the user (González Camacho and Souza, 2018). While implicit methods require no explicit user actions, they lack the possibility of improving the system from user feedback. On the other hand, explicit methods may require more user

effort but offer opportunities for user feedback and improvement. From now on this paper will lay focus on explicit methods for acquiring knowledge, and such implicit methods are not discussed anymore (Gope and Jain, 2017).

Cold start strategies can be evaluated online or offline (Elahi et al., 2016). As mentioned in the section 3.2, online evaluation with real users is rarely carried out, as it requires a fully developed recommender system with a large user base, making it challenging to compare several strategies in different system configurations (Elahi et al., 2016). In contrast, offline evaluations use pre-collected datasets, which do not completely encapsulate natural user interactions. The system can only ask users to rate items that they have already rated in the past, included in the training set, and cannot simulate the scenario where the system suggests new or unknown items to the user for rating. Therefore, the offline evaluation may not accurately reflect the user's real behavior and preferences when interacting with the system in a live environment (A. Rashid et al., 2002; Karimi et al., 2013).

The effectiveness of cold start recommendations is commonly evaluated by using accuracy measures such as RMSE or MAE (Gope and Jain, 2017; Elahi et al., 2016). If the experiments are not able to replicate true online outcomes, one should also consider the potential biases, adaptability to real-world scenarios, and diversity of the recommendations.

5 Active learning in recommender systems

In this chapter, we briefly introduce active learning and explore how it can be used as a tool for eliciting new user preferences in collaborative filtering.

Recommender systems may actively elicit the user to rate items, in a process known as active learning (Rubens et al., 2016). Active learning is a machine learning technique where data points that minimize the loss function are actively gathered to the training set (Melville and Sindhwani, 2017; Gope and Jain, 2017). In recommender systems, the goal of active learning is to present users with items that offer the most insightful information about their preferences, thereby collecting missing ratings and improving recommendations. It is well-motivated for situations where the original training data is not sufficient or acquiring it is impractical, such as the cold start problem (Elahi et al., 2016).

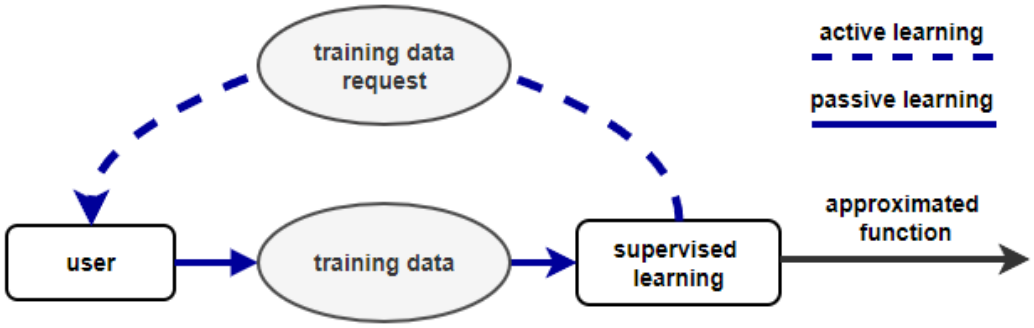

Figure 5.1: Active learning employs an iterative process for obtaining training data, unlike passive learning, where the data is simply given (Rubens et al., 2016)

However, choosing the most informative items is not a trivial task, as the outcome is not known in advance. Active learning methods implement various heuristics for selecting the items. Therefore, the gist of active learning in recommender systems is finding a heuristic that improves the accuracy of the recommendation system the most (Elahi et al., 2016).

5.1 Active learning for the new user cold start problem

Active learning can be employed in collaborative filtering to effectively select and query informative items for new users, taking into consideration their previous feedback and preferences (A. Rashid et al., 2002; Elahi et al., 2016; Gope and Jain, 2017). This process involves an initial interview, asking the user to rate a carefully curated set of items based on some heuristic, such as popularity or entropy. By iteratively choosing items that provide the most information about user preferences, active learning progressively improves its recommendations for new users. The main goal of active learning strategies for new users is to minimize user effort (Gope and Jain, 2017; Rubens et al., 2016). Therefore, the selected items should be highly informative, while still retaining a minimal number of items to avoid overwhelming the user (A. Rashid et al., 2002; Geurts et al., 2020).

Active learning strategies can be divided into personalized and non-personalized strategies. Non-personalized strategies do not take the user into account and try to maximize information gained per presented item (A. Rashid et al., 2002). On the other hand, personalized strategies also consider users' previous ratings. Both of these categories can be split into single-heuristic and multiple-heuristic strategies. Single-heuristic strategies use a single measure to evaluate each item while multiple-heuristic strategies take advantage of multiple criteria (Elahi et al., 2016).

The heuristic is some criteria or strategy which works as an objective function to choose the interview items (Gope and Jain, 2017). The heuristics are evaluated against each other in order to find which strategy provides the most information about the user's preferences Rubens et al., 2016. For example, entropy, "The relative frequency of each of the five possible ratings" (A. Rashid et al., 2002), for item i is computed accordingly (Geurts et al., 2020):

$$\text{Entropy}(i) = -\sum_{j \in 0,1} p(j \mid i) \log_2 p(j \mid i)$$

where $p(j \mid i)$ is the relative frequency of a positive interaction ($j = 1$) or negative interaction ($j = 0$) with item i. Positive interaction means a rating of 4 or 5, and negative interaction is 3 or less. This strategy aims to find items that have the most dividing opinions, thus potentially giving the most information (Geurts et al., 2020).

6 Non-personalized active learning strategies

This chapter evaluates non-personalized active learning strategies for tackling the new user cold start problem. The discussion revolves around studies conducted by A. Rashid et al. (2002) and Geurts et al. (2020), which compare different non-personalized strategies, including one simple single-heuristic personalized strategy. This chapter serves as a basis for comparing non-personalized strategies with personalized strategies discussed in the next chapter.

6.1 Experiments for non-personalized strategies

Study	**Best Performing Strategy (No. Questions)**				**Dataset**	**Performance Measure**
A. Rashid et al., 2002	PopEnt (30)	PopEnt (45)	PopEnt (60)	PopEnt (90)	MovieLens	MAE
Geurts et al., 2020	Variance (10)	Random (25)	PopError (50)	PopGini (100)	Dutch department store	RMSE

Table 6.1: Comparison between the two non-personalized active learning experiments by Geurts et al. (2020) and A. Rashid et al. (2002)

A. Rashid et al., 2002 compared non-personalized strategies and one personalized strategy for new users. Users were asked to rate 30, 45, 60, or 90 movies chosen by each heuristic (A. Rashid et al., 2002). The offline experiment used a snapshot of the MovieLens database with 7,335 users, 4,117 movies, and over 2.7 million ratings. Users who had fewer than 200 ratings were eliminated. Mean Absolute Error (MAE) was calculated for all strategies, considering every user in the test set and the overall average MAE.

In a similar study by Geurts et al. (2020), only non-personalized strategies were compared by asking new users to rate either 10, 25, 50, or 100 different items. The experiment was conducted offline and used a dataset by a Dutch department store consisting of 563,495 users and 242,020 items and 2,563,878 total ratings. The results were evaluated by calculating the RMSE of each user's ratings for each strategy (Geurts et al., 2020).

6.2 Results for non-personalized strategies

Both studies found that combining popularity with another heuristic gives the best performance for non-personalized strategies in terms of recommendation accuracy for cold users. The studies demonstrated that choosing items solely by popularity achieves many ratings but suffers from low information as the ratings of popular items tend to be widely liked (A. Rashid et al., 2002; Elahi et al., 2016). By combining popularity with impurity measures such as entropy or error, the information gain was increased while each item maintained a relatively good rate of ratings (Geurts et al., 2020; A. Rashid et al., 2002). However, popularity was still the best candidate in the single-heuristic category in both experiments.

The results also showed that the best-performing strategy depends on the number of items shown to the cold user. For Geurts et al. (2020), PopError and PopGini strategies performed the best when the number of shown items was 50 and 100, respectively. The same was observed by A. Rashid et al. (2002). Strategies that required less user effort had lower accuracy, and vice versa. Therefore, one needs to acknowledge the possible trade-off when choosing the right strategy.

Additionally, A. Rashid et al. (2002) demonstrated that while the only personalized item-item strategy was able to gather the most ratings in the interview, it did not translate into better recommendations. The problem lies in its tendency to focus on clusters of similar items, limiting its ability to make accurate predictions on a wider variety of movies. The lack of coverage and diversity contributed to lower performance in accuracy, even worse than random sampling (A. Rashid et al., 2002).

One of the main problems with offline experiments is that they "implicitly and erroneously assume that all the items selected and proposed by the system to rate are known to the user" (Elahi et al., 2016). Neither of the studies was able to gather comparable online data. A. Rashid et al. (2002) explained that it was hard to calculate the MAE immediately after the sign-up, due to unexplained reasons. Offline datasets tend to inherit biases, so testing the strategies in an online environment would verify the true effectiveness of the system.

7 Personalized active learning strategies

Now we move on to experiments that improve the interview process by taking users' previous answers into account. These personalized strategies are based on a decision tree algorithm originally proposed by Golbandi et al. (2011). Karimi et al. (2013) and Zhou et al. (2011) utilize multiple heuristics by incorporating matrix factorization into the decision tree algorithm in two different ways. The results are used to draw a few practical guidelines on the design and evaluation of active learning strategies in eliciting new user preferences.

7.1 Experiments for personalized strategies

Decision trees have been shown to fit well into the initial interview process (Golbandi et al., 2011). In the decision tree model by Golbandi et al. (2011), the interview process is modeled by a decision tree where each internal node corresponds to a single item that the user is asked to rate. In the interview, the user can choose to "like", "dislike", or "unknown" to the item, which directs the user to the next node. The interview items are chosen by the best splitting item that partitions the users evenly into three groups. The item rating is predicted simply by taking the mean rating by its corresponding users in the node. Final recommendations are generated after the user terminates the questionnaire, offering the highest-rated items by the users belonging to the current node. The interview process can terminate anywhere, so the user can choose how many questions they want to answer, at the cost of recommendation quality (Golbandi et al., 2011).

Golbandi et al. (2011) trained and tested the decision tree algorithm with Netflix training and testing data with 100 million and 2.8 million ratings, respectively. Four different non-personalized methods were chosen for comparison.

Karimi et al. (2013) introduced a Factorized Decision Tree (FDT), which improved the original decision tree accuracy by incorporating matrix factorization for predicting the item ratings. "The motivation is that as MF outperforms item average in rating prediction for active users, it makes sense to use it for new users as well" (Karimi et al., 2013). The core idea is that after constructing the decision tree by Golbandi et al. (2011), a new matrix

factorization model is trained for each level of the decision tree, and all new ratings are calculated for the whole dataset while maintaining the tree structure. This way, predicted ratings are calculated whenever the user stops the interview.

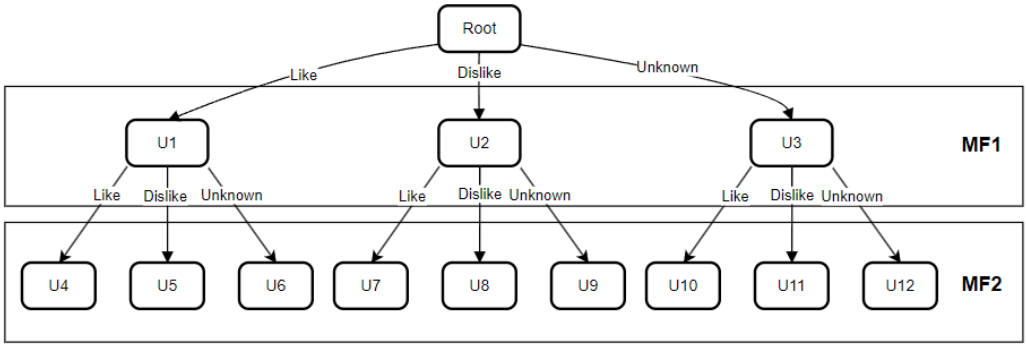

Figure 7.1: The first two levels of the decision tree in the FDT model. It considers one MF model per each level. Each node contains a unique set of users who reached that part of the tree.

Karimi et al. (2013) also introduced a new faster tree construction method, Most Popular Sampling (MPS). Instead of checking all the candidate items for each node, only the most popular items within the users associated with the node are considered (Golbandi et al., 2011). The reason behind the heuristic is to overcome the problem with sparse data. Choosing popular items among the specific domain increases the chances of splitting users more efficiently, as fewer "unknown" answers would be gathered, which do not offer much information about the user.

Karimi et al. (2013) used the Netflix dataset to test FDT in an offline setting, but instead of using the given sets, they split the users into two disjoint subsets, the training set and test set, containing 75% and 25% users, respectively. The issue was that the original dataset contained the same users in the training and test set, so it was not suitable for new users, which was not addressed by Golbandi et al. (2011). The performance of FDT was reported in terms of RMSE.

(Zhou et al., 2011) implemented Functional Matrix Factorization (fMF), which applied matrix factorization in constructing the decision tree itself. fMF recursively constructs a new optimal latent user profile for each node of the tree, which represents all the users that have answered similarly thus far in the questionnaire. Therefore, each user profile in a node is a function of the previous replies by the set of users. The item profiles are learned by minimizing the regularized prediction error by the user profiles in each node (Zhou et al., 2011).

Zhou et al. (2011) evaluated fMF with three large datasets: MovieLens, EachMovie, and

Netflix, by splitting two disjoint 75% and 25% training sets and test sets, respectively. The prediction accuracy was calculated by RMSE. fMF was compared against the original decision tree and TreeU method which also incorporates matrix factorization.

7.2 Results for personalized strategies

Firstly, decision tree, FDT, and fMF were all found to outperform non-personalized strategies (Golbandi et al., 2011; Zhou et al., 2011; Karimi et al., 2013). Therefore, they are all capable of refining user preferences through the interview process and can be applied to solve the cold-start problem. The final prediction accuracy was mostly influenced by how far the user gets in the decision tree and the number of “unknown" answers (Karimi et al., 2013; Zhou et al., 2011). It was also shown that asking more than 8 questions does not significantly improve the final prediction, so there is no need to train the decision tree model further than that (Karimi et al., 2013).

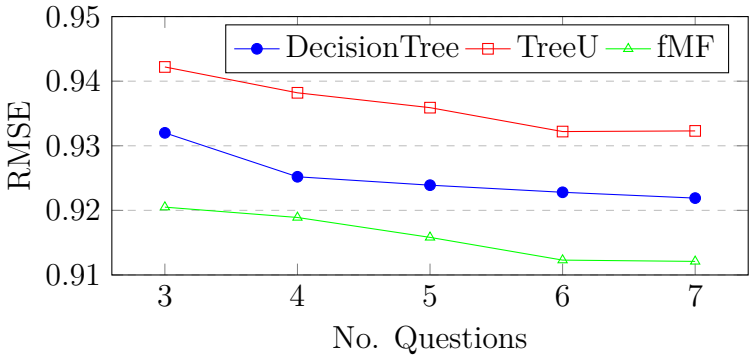

Figure 7.2: RMSE on the Netflix dataset for cold-start users with respect to the number of interview questions by Zhou et al. (2011). FDT was not included in the comparison.

Incorporating matrix factorization into decision trees has shown to improve recommendations for new users, as both FDT and fMF were found to significantly outperform the original decision tree (Karimi et al., 2013; Zhou et al., 2011). Karimi et al. (2013) was able to speed up the tree construction without harming the accuracy with MPS, and prediction accuracy with FDT. Zhou et al. (2011) demonstrated how matrix factorization can be naturally integrated into the initial interview process with fMF, in both the tree construction and item predictions.

It is difficult to determine the overall best method as each has its strengths. However, the results indicate that fMF outperforms FDT in user profiling. As each node in the tree of

fMF is described by a user profile, it can more accurately describe the item preferences of that domain for new users, resulting in better recommendations. However, according to Karimi et al. (2013), fMF is really slow for large data sets such as Netflix, whereas FDT is much simpler and has significantly lower time complexity. The time complexity can make quite a significant difference in a real-world setting, as the model needs to generate the interview questions in real time, and too long wait times increase user effort.

7.3 The problem with "unknown" answers

It was observed that the earlier the user selects an "unknown" (resulting in a missing value) in the interview, the lower the accuracy of the final recommendations was (Golbandi et al., 2011; Karimi et al., 2013). Golbandi et al. (2011) suggests the use of multiple decision trees which the user would traverse simultaneously, giving the user a choice to rate from multiple items, increasing the chance of knowing at least one. In addition, A. Rashid et al. (2002) observed that users preferred to rate several movies per page. Therefore, using multiple decision trees could possibly decrease user effort while increasing prediction accuracy. This, however, comes at the cost of computational expenses.

Zhou et al. (2011) observed that missing values in the training data itself introduce bias, which had been largely ignored in previous works. They conducted an experiment that varied the number of users with few ratings in the training set and found that performance is indeed affected by the bias introduced by missing values in the training set (Zhou et al., 2011).

7.4 Bias in active learning

A. Rashid et al. (2002) noted that "active learning methods exploit intelligence by assuming user behaviour may induce biases in rating distributions". The initial interview process assumes that users rate items honestly, and consistently, and have expertise in the items they are provided. To reduce bias, Karimi et al. (2013) and Golbandi et al. (2011) use both user- and item-bias terms in the tree learning algorithm, which captures the inclination of certain users giving, and items receiving higher ratings than others. A. Rashid et al. (2002) recommends including a small amount of randomness in the initial interview process to get rid of potential biases but also offer more diverse content, which especially non-personalized methods might struggle with.

8 Conclusion

Overall, this work demonstrates the effectiveness of active learning strategies in improving collaborative filtering recommender systems for new users and highlights the importance of personalized approaches in eliciting user preferences. To summarize, chapters 2 and 3 provided an overview of recommender systems and collaborative filtering, introducing matrix factorization as the base recommendation method. Moreover, chapters 4 and 5 explored the relationship between collaborative filtering, new user cold start, and active learning. Finally, chapter 6 and 7 evaluated non-personalized and personalized active learning strategies through experimental analysis.

Non-personalized methods, although easy to implement and computationally efficient, often demand a significant user effort that may not be feasible in online settings. To address this, a non-personalized active learning strategy should combine multiple heuristics, such as popularity and an impurity measure, to enhance the information gained from a single item while maintaining a consistent number of ratings (Elahi et al., 2016; Geurts et al., 2020; A. Rashid et al., 2002). It is important to note that the optimal non-personalized strategy also relies on the total number of items presented in the initial interview, highlighting the need to find the right trade-off between user effort and recommendation accuracy.

According to the results, decision tree-based personalized methods seem to outperform non-personalized methods while being computationally more expensive (Golbandi et al., 2011; Karimi et al., 2013; Zhou et al., 2011). While non-personalized strategies overlook user subjectivity, taste, decision-making, and item knowledge, personalized methods leverage past responses, emphasizing user subjectivity. Incorporating matrix factorization into decision trees effectively integrates the recommendation model into the initial interview, increasing the chance of acquiring as many ratings as possible for making accurate recommendations.

Lastly, one should keep in mind that active learning methods can introduce biases due to assumptions about user behavior (Elahi et al., 2016; Karimi et al., 2013; A. Rashid et al., 2002). The lack of proper online experimentation in the previously discussed experiments hinders the ability to verify the effectiveness of the models. To address this, conducting online experiments would validate the offline results and ensure the selected method's effectiveness, especially since the offline setting assumes user familiarity with all items.

Bibliography

Aggarwal, C. C. (2016). *Recommender Systems - The Textbook*. Springer, pp. 1–498. DOI: 10.1007/978-3-319-29659-3.

Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrez, A. (2013). "Recommender systems survey". In: *Knowledge-Based Systems* 46, pp. 109–132. DOI: 10.1016/j.knosys.2013.03.012.

Elahi, M., Ricci, F., and Rubens, N. (2016). "A survey of active learning in collaborative filtering recommender systems". In: *Computer Science Review* 20, pp. 29–50. DOI: 10.1016/j.cosrev.2016.05.002.

Elngar, A. (2020). *Recommender System with Machine Learning and Artificial Intelligence - Practical Tools and Applications in Medical, Agricultural and Other Industries*. DOI: 10.1002/9781119711582.

Geurts, T., Giannikis, S., and Frasincar, F. (2020). "Active learning strategies for solving the cold user problem in model-based recommender systems". In: *Web Intelligence* 18.4, pp. 269–283. DOI: 10.3233/WEB-210448.

Golbandi, N., Koren, Y., and Lempel, R. (2011). "Adaptive Bootstrapping of Recommender Systems Using Decision Trees". In: *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*. New York, NY, USA: Association for Computing Machinery, pp. 595–604. DOI: 10.1145/1935826.1935910.

González Camacho, L. A. and Souza, S. N. A. (July 2018). "Social network data to alleviate cold-start in recommender system: A systematic review". In: *Information Processing and Management* 54, pp. 529–544. DOI: 10.1016/j.ipm.2018.03.004.

Gope, J. and Jain, S. (May 2017). "A survey on solving cold start problem in recommender systems". In: pp. 133–138. DOI: 10.1109/CCAA.2017.8229786.

Jesús, B., Fernando, O., Antonio, H., and Bernal, J. (2012). "A collaborative filtering approach to mitigate the new user cold start problem". In: pp. 225–238. DOI: 10.1016/j.knosys.2011.07.021..

Karimi, R., Wistuba, M., Nanopoulos, A., and Schmidt-Thieme, L. (2013). "Factorized Decision Trees for Active Learning in Recommender Systems". In: *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, pp. 404–411. DOI: 10.1109/ICTAI.2013.67.

Koren, Y., Bell, R., and Volinsky, C. (2009). "Matrix Factorization Techniques for Recommender Systems". In: 42.8, pp. 30–37. DOI: 10.1109/MC.2009.263.

Linden, G., Smith, B., and York, J. (2003). "Amazon.com recommendations: item-to-item collaborative filtering". In: *IEEE Internet Computing* 7.1, pp. 76–80. DOI: 10.1109/MIC.2003.1167344.

Melville, P. and Sindhwani, V. (2017). *Recommender Systems*. Ed. by C. Sammut and G. I. Webb, pp. 1056–1066. DOI: 10.1007/978-1-4899-7687-1_964.

Rashid, A., Albert, I., Cosley, D., Lam, S., McNee, S., Konstan, J., and Riedl, J. (Feb. 2002). "Getting to Know You: Learning New User Preferences in Recommender Systems". In: *International Conference on Intelligent User Interfaces, Proceedings IUI*. DOI: 10.1145/502716.502737.

Rashid, A. M., Karypis, G., and Riedl, J. (2008). "Learning Preferences of New Users in Recommender Systems: An Information Theoretic Approach". In: *SIGKDD Explor. Newsl.* 10.2, pp. 90–100. DOI: 10.1145/1540276.1540302.

Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B. (2010). *Recommender Systems Handbook*. 1st. Berlin, Heidelberg: Springer-Verlag. DOI: 10.1007/978-0-387-85820-3.

Rubens, N., Elahi, M., Sugiyama, M., and Kaplan, D. (Feb. 2016). "Active Learning in Recommender Systems". In: pp. 809–846. ISBN: 978-1-4899-7637-6. DOI: 10.1007/978-1-4899-7637-6_24.

Zhou, K., Yang, S.-H., and Zha, H. (2011). "Functional Matrix Factorizations for Cold-Start Recommendation". In: pp. 315–324. DOI: 10.1145/2009916.2009961.