JÄMFÖRELSE AV GRANNBASERAD OCH MODELLBASERAD SAMARBETSFILTRERING I REKOMMENDATIONSSYSTEM

Otto Westerlund, 35686
Kandidatavhandling i datavetenskap
Handledare: Sepinoud Azimi
Fakulteten för naturvetenskaper och teknik
Åbo Akademi
2021

Referat

Sedan början av 2000-talet har världen i snabb takt digitaliserats. Allt mera tjänster erbjuds via Internet. Mängden data på nätet har ökat enormt, och det har blivit opraktiskt att manuellt filtrera fram den information man behöver. Rekommendationssystem är en lösning till detta problem.

I denna avhandling undersöks rekommendationssystem, mer specifikt metoder inom samarbetsfiltrering. Samarbetsfiltrering brukar delas in i grannbaserade och modellbaserade metoder, teori om dessa kategorier presenteras. Sedan undersöks fallstudier som erbjuder förbättringsförslag till grannbaserade metoder eller till modellbaserade metoder. Till slut jämförs bägge kategorierna och resultaten från fallstudierna.

Grannbaserade metoder är populära och används mycket och har en del fördelar; de är enkla att förstå och implementera, och det är enklare att tolka varför de rekommenderar som de gör. Modellbaserade metoder, speciellt latent faktor metoder, ger dock i regel mera noggranna rekommendationer.

1. Inledning

Under 2000-talet har världen i snabb takt digitaliserats. Allt flera människor har tillgång till Internetuppkoppling, och allt flera tjänster erbjuds numera via nätet. I och med den digitala transformationen har mängden data på nätet ökat explosionsartat, t.ex. erbjuder Amazon.com leverans av 45 miljoner produkter [1], och över 500 timmar video laddas upp till YouTube varje minut [2]. Datamängderna har blivit så enorma, att det har blivit opraktiskt att manuellt leta efter och filtrera fram den information man vill komma åt. Som en lösning till detta problem har det utvecklats *rekommendationssystem*.

Ett rekommendationssystem är ett mjukvarusystem som med hjälp av algoritmer försöker härleda en användares intressen, och baserat på dessa rekommenderar det produkter som användaren kunde tänkas vara intresserad av. En av fördelarna med ett väl designat rekommendationssystem är att det kan rekommendera varor åt en kund, vilka hen annars inte ens skulle vara medveten om att existerar. Detta leder till att kunden har en högre chans att konsumera varor som verkligen motsvarar sina intressen, och samtidigt leder det till att producenter kan nå kunder som annars inte är medvetna om deras existens. Rekommendationssystem kan alltså öka konsumtionsdiversitet. Det är dock värt att nämna att designbeslut av rekommendationssystem kan även leda till att färdigt populära varor blir ännu populärare, medan nischvaror inte får någon synlighet hos kunderna [3].

Rekommendationssystem har blivit en standardservice för olika webbaserade tjänster. Ett av de mest kända tillämpningsområdena för rekommendationssystem är e-handelssektorn. Forskning av rekommendationssystem inom e-handeln visar att rekommendationssystem kan påverka hur producenter konkurrerar i prissättningen av produkter, t.ex. kan systemet rekommendera konkurrerande produkter till en vara som en kund håller på att köpa, vilket leder till att producenterna tvingas konkurrera hårdare i hur de prissätter produkter. Däremot kan rekommendationssystem låta producenter använda mindre pengar på reklam, då deras produkter blir sedda i samband med köp av andra produkter [3]. Överlag har rekommendationssystem en stor ekonomisk inverkan på e-handelns

försäljningsmängd. År 2008 rapporterades det att 35 % av Amazons försäljning och 60 % av Netflix dvd-uthyrningar var en konsekvens av rekommendationer [4] [5].

Förutom e-handels- och underhållningssektorn så finns det forskning om användning av rekommendationssystem i nischbranscher. Inom utbildningsbranschen erbjuder rekommendationssystem möjligheter till att exempelvis hjälpa studerande med att hitta läromaterial, att identifiera studerande med inlärningssvårigheter, att hänvisa studerande till stipendier, eller till att rekommendera personliga undervisningsprogram [6]. Dessutom undersöks möjligheterna som erbjuds av rekommendationssystem inom hälso- och mentalvård [7].

Det har blivit klart att rekommendationssystem har en stor påverkan på dagens samhälle, både ur ett ekonomiskt och ett allmännyttigt perspektiv. Graden av nytta hos ett rekommendationssystem beror på faktorer såsom hur noggrant systemet är. Syftet med denna avhandling är att undersöka och jämföra olika modeller som används för att implementera rekommendationssystem, och att redogöra för deras för- och nackdelar.

2. Bakgrund

Rekommendationssystem kan grovt delas in i några kategorier beroende på hur de fungerar; de vanligaste kategorierna är *samarbetsfiltrering* (collaborative filtering) och *innehållsbaserad filtrering* (content-based filtering). Även andra kategorier och hybrider av dessa förekommer [8] [9].

Oftast samlar rekommendationssystem betyg givna av användare till artiklar. I samarbetsfiltrering försöker algoritmerna sedan uppskatta vilka användare respektive artiklar som påminner om varandra. Sedan rekommenderas artiklar till en användare som andra liknande användare har gett höga betyg, eller artiklar som liknar de artiklar som användaren själv har gett höga betyg.

I innehållsbaserad filtrering är rekommendationerna baserade på artikelns innehålls egenskaper – t.ex. om en användare har gett högt betyg åt en viss film, så kan algoritmerna i innehållsbaserad filtrering rekommendera andra filmer som är av samma genre, eller med samma skådespelare [8].

Samarbetsfiltrering och innehållsbaserad filtrering har olika styrkor och svagheter. Innehållsbaserad filtrering kan ge rekommendationer utgående från ett fåtal betyg av en användare. Däremot har innehållsbaserad filtrering den nackdelen att användaren lätt fastnar i en ”innehållsbubbla”, dvs. hens rekommendationer saknar variation. Detta kan leda till att användaren blir uttråkad och slutar följa rekommendationer. Samarbetsfiltrering däremot brukar ge större variation i rekommendationerna, men har svårigheter att ge noggranna rekommendationer om användaren har betygsatt endast ett fåtal artiklar.

I detta arbete undersöks samarbetsfiltrering; dessa metoder är de mest använda och de framgångsrikaste [10] [11].

2.1 Rekommendationsproblemet

Rekommendationssystem behandlar tre typer av entiteter: användare, artiklar och betyg. Användarna motsvaras av t.ex. kunder i en webbtjänst, medan artiklarna motsvaras av t.ex. filmer, musiklåtar, varor i en nätbutik och liknande. Systemet samlar in betyg givna av användarna till artiklarna, och dess uppgift är att rekommendera artiklar som användaren inte ännu har sett åt användaren. Rekommendationen baseras på en uppskattning av vilka betyg användaren troligen skulle ge artiklarna i fråga.

Betyg som samlas av användarna kan vara *explicita*, t.ex. kunde en användare betygsätta en film på en skala från 1 till 5 stjärnor. Betygen kan också vara *implicita*, dvs. om en användare köper en vara från en nätbutik, så anses det vara en (positiv) betygsättning av artikeln trots att användaren inte specifikt har givit den ett betyg [8] [12]. I praktiken används ofta en kombination av explicita och implicita betyg för att maximera noggrannheten på rekommendationer.

Användare, artiklar och betyg representeras ofta av 3-tupler av formen [användare, artikel, betyg]. Genom att samla ihop dessa tupler kan man bilda en *betygsmatris* (User-Item rating matrix) av storleken $m \times n$ för m användare och n artiklar, där varje element r_{ui} motsvarar ett betyg givet av användare u till artikel i [10]. Systemet försöker sedan räkna ut värdet på element som saknas i matrisen, dvs. gissa hur användare skulle betygsätta artiklar som de inte har gett betyg åt ännu.

Metoderna inom samarbetsfiltrering kan ses som en generalisering av klassificerings- eller regressionsproblem inom maskininlärning [8]. Som exempel kan man nämna den klassiska irisdatamängden (en samling med data om olika arter av irisblommor).

Inom klassificeringsproblem kan datamängden ses som en matris, där varje rad motsvarar ett exemplar av något slag (t.ex. en enskild blomma i irisdatamängden), och varje kolumn motsvarar ett visst attribut som beskriver exemplaret (t.ex. längden på blommans kronblad). Den sista kolumnen i matrisen motsvarar *målvariabeln* (target variable/feature), som i irisfallet motsvarar blommans art. En algoritm försöker sedan lära sig mönster baserat på attributen som motsvarar målvariablerna, och utgående från den inlärda modellen försöker den sedan bestämma nya exemplars målvariabel baserat på deras karaktärsdrag (se figur 1).

Ex 1	5.0	3.2	4.7	2.9	Setosa
Ex 2	3.8	3.7	4.6	2.7	Virginica
Ex 3	5.1	3.6	4.9	2.9	Versicolor
Ex 4	4.3	3.7	4.2	2.7	Virginica
Ex 5	4.6	4.2	4.4	2.8	Setosa
Ex 6	3.9	3.9	4.5	2.6	
Ex 7	4.0	3.8	4.3	2.6	
	K 1	K 2	K 3	K 4	Mål

Figur 1. Raderna i matrisen motsvarar exemplar Ex av irisblommor, kolumnerna motsvarar exemplarets attribut K (t.ex. bladens längd eller bredd i centimeter). Målvariabeln är blommans art. Maskininlärningsalgoritmen försöker gissa värdet på de tomma målvariablerna. Figuren tillämpad från [8].

I klassificeringsproblem är det alltid samma kolumn i datamatrisen som fungerar som målvariabel. Däremot kan alla kolumner fungera som målvariabel i samarbetsfiltrering [8]. Se figur 2 för en visualisering av rekommendationsproblemet inom samarbetsfiltrering. Det är värt att notera att oftast är det endast få värden som är ifyllda i betygsmatrisen, dvs. ger användarna i praktiken betyg åt endast ett fåtal artiklar [8] [13] [14].

	P 1	P 2	P 3	P 4	P 5
A 1	3		1	2	
A 2		4	4		
A 3	5	3	4	2	
A 4	1				3
A 5			2		
A 6		3			3
A 7	2			4	

Figur 2. Figuren representerar en betygsmatris. Raderna motsvarar användare, kolumnerna motsvarar artiklar. Värdena motsvarar betyget som användare A har gett åt artikel P. Algoritmerna försöker gissa värdet på de betyg som saknas i matrisen. Figuren tillämpad från [8].

Grovt sett brukar metoderna i samarbetsfiltrering delas in i kategorierna *grannbaserade* eller *minnesbaserade* metoder (neighborhood-based/memory-based methods) och *modellbaserade* metoder (model-based methods) [8] [13] [15]. Till följande beskrivs bägge klasserna av metoder.

2.2 Grannbaserade metoder

Grannbaserade metoder är de vanligaste metoderna som används i samarbetsfiltrering [16]. De baserar sig på K-Närmaste Grannar (K-Nearest Neighbors eller KNN) algoritmen i maskininlärning [17] [14]. I dessa metoder försöker algoritmen beräkna likheter mellan användare, och om användare A liknar användare B, så kan algoritmen rekommendera användare B:s favoritartiklar som A inte har sett åt A och tvärtom; eller så försöker algoritmen beräkna likheter mellan artiklar, och rekommenderar artiklar åt användare A som liknar A:s favoritartiklar [14]. Det första sättet kallas *användarbaserad samarbetsfiltrering* (user-based collaborative filtering) och det andra sättet *artikelbaserad samarbetsfiltrering* (item-based collaborative filtering) [8] [17]. Tekniskt så liknar dessa lösningar varandra väldigt mycket, därför kommer de inte att beskrivas skilt utan arbetet kommer att koncentrera sig på användarbaserad samarbetsfiltrering.

Det finns olika formler för att mäta likheten (som motsvaras av avstånd mellan element i KNN-algoritmen) mellan användare. Ett exempel är *Pearsons korrelationskoefficient*, som beräknas enligt följande princip:

Låt I_u vara mängden av betyg på artiklar som användare *u* har gett i betygsmatrisen med m användare och n artiklar. Då kan man beteckna mängden av artiklar som både användare *u* och användare *v* har gett betyg åt som $I_u \cap I_v$. Till först beräknas medelvärdet μ_u på de betyg som *u* har gett:

$$\mu_u = \frac{\sum_{k \in I_u} r_{uk}}{|I_u|}, \forall u \in \{1 \dots m\}$$

[8]. Medelvärdet används sedan för att beräkna likheten $sim_{(u,v)}$ (Pearsons korrelationskoefficient) mellan användare *u* och *v* enligt följande formel:

$$sim_{(u,v)} = Pearson_{(u,v)} = \frac{\sum_{k \in I_u \cap I_v}(r_{uk} - \mu_u)(r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \cap I_v}(r_{uk} - \mu_u)^2}\sqrt{\sum_{k \in I_u \cap I_v}(r_{vk} - \mu_v)^2}}$$

[8] [17], där r_{uk} betecknar ett betyg som användare *u* har gett åt artikel *k* i betygsmatrisen.

Ett problem som har upptäckts är att olika användare har olika skala på hur de betygsätter artiklar, dvs. vissa användare brukar i regel ge höga betyg medan andra brukar i regel ge lägre betyg [8] [15]. Detta är någonting som brukar beaktas i rekommendationsformlerna, till exempel genom att subtrahera medelbetyget från egentliga betyget för alla betyg som användare *u* har gett enligt följande formel:

$$s_{ui} = r_{ui} - \mu_u, \forall u \in \{1 \dots m\}$$

Sedan beräknas ett uppskattat betyg $\hat{r}_{ui}$ för användare u på artikel i (som u inte ännu har gett ett betyg åt) enligt följande formel: Låt $P_u(i)$ vara mängden av k stycken närmaste grannar till u (de k stycken användare som har högsta Pearsons korrelationskoefficient med u), som har gett ett betyg åt artikel i. Då är formeln för det uppskattade betyget åt artikel i för användare u:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in P_u(i)} sim(u,v) * s_{vi}}{\sum_{v \in P_u(i)} |sim(u,v)|}$$

[8]. Av de uppskattade betygen som räknats ut, rekommenderas åt användaren n stycken artiklar med de högsta uppskattade betygen.

Pearsons korrelationskoefficient är ett exempel på en formel för att uppskatta likheten mellan användare, men även andra mått forskas; några exempel är cosinuslikhet, viktad Pearsons korrelationskoefficient och Jaccard. Senare i avhandlingen jämförs Feng et al.:s likhetsmått [11] med andra förbättringsförslag för grannabaserade metoder och samarbetsfiltrering överlag.

2.3 Modellbaserade metoder

I modellbaserade metoder bygger någon algoritm en modell som beskriver användarnas beteende när de betygsätter artiklarna. Modellen fungerar som bas för att räkna ut ospecificerade värden i betygsmatrisen. Många metoder som används för klassificering eller regression i maskininlärning kan generaliseras till samarbetsfiltrering, t.ex. beslutsträd, naiv bayesiansk klassificerare och andra. Ett gemensamt problem som dessa metoder har är det faktum att betygsmatrisen har endast ett fåtal specificerade element. Ofta används dimensionell minskning för att skapa en minskad betygsmatris som är fullt specificerad, vilket gör det möjligt att använda de ovannämnda metoderna för samarbetsfiltrering. Dimensionell minskning gör en annan metod, latent faktor metoden, speciellt effektiv för att noggrant uppskatta de ospecificerade elementen i betygsmatrisen.

2.3.1 Beslutsträd

Ett beslutsträd är en modell som kan användas för klassificering eller regression i maskininlärning. I ett beslutsträd delas attributen från en datamängd upp i en trädliknande struktur, där varje attribut är en nod i trädet, och varje gren motsvarar hur attributet förhåller sig till målvariabeln. Löven i träden ger sedan ett värde till målvariabeln.

En utmaning i användningen av beslutsträd för samarbetsfiltrering är att till skillnad från vanliga klassificeringsproblem, så är inte betygsmatrisen tydligt indelad i attributkolumner och målvariabelkolumn. Varje kolumn kan fungera som målvariabel.

Lösningen till detta är att skapa ett skilt beslutsträd för varje kolumn (artikel) i betygsmatrisen, dvs. för en $m \times n$ betygsmatris får man n stycken beslutsträd, en för varje artikel [8].

2.3.2 Naiv Bayes

Naiv bayesiansk klassificerare utför klassificering baserat på sannolikheter utgående från observerat data. Om betygen kan anta värden $\{v_1, v_2, \ldots v_l\}$ (en vanlig skala är 1 – 5), så beräknar naiva bayesianska klassificeraren sannolikheten att ett betyg r_{ui} som saknas antar värdet $v_s \in \{v_1, v_2, \ldots v_l\}$. Denna beräkning kan utföras med formeln:

$$\hat{r}_{ui} = argmax_{v_s} P(r_{ui} = v_s) \cdot \prod_{k \in I_u} P(r_{uk} | r_{ui} = v_s)$$

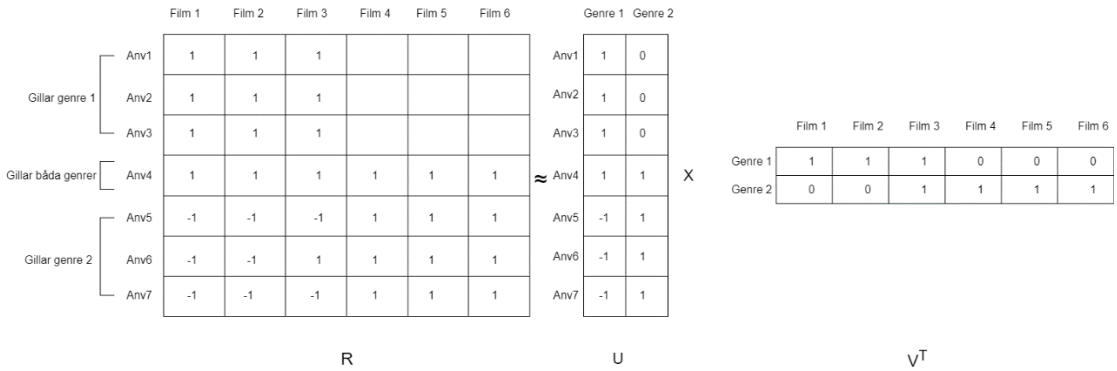
Värdet för termen $P(r_{ui} = v_s)$ är andelen av användare som har gett betyget v_s åt artikel i. Värdet för termen $P(r_{uk} \mid r_{ui} = v_s)$ är andelen av användare som har gett betyget r_{uk} åt artikel k i mängden I_u, givet att de har gett betyget v_s åt artikel i. Mängden I_u är mängden av artiklar som användare u har gett betyg åt. Som resultat tas det betyg v_s som maximerar resultatet för uttrycket.

2.3.3 Latent faktor metod

Latent faktor metoder baserar sig på antagandet att man behöver inte explicit känna till faktorerna som påverkar en användares betygsättning av en artikel [18]. En $m \times n$ betygsmatris R kan ungefärligt faktoriseras till en $m \times k$ matris U och en $n \times k$ matris V enligt följande princip:

$$R \approx UV^T$$

[8].Kolumnerna i U och V kallas latenta faktorer, och raderna i U och V kallas latenta vektorer. Latenta faktorerna väljs enligt antagandet att deras komponenter är starkt korrelerade.



Figur 3. Exempel på faktorisering av betygsmatris R till latenta faktorer i matriserna U och V. Figur adapterad från [8].

Figur 3 visar ett exempel på faktorisering av en betygsmatris till latenta faktorer. I matris R används betyg -1 för att indikera missnöje med en artikel (filmer i detta exempel), och 1 för att indikera att man gillar artikeln. Användarna 1 – 3 gillar filmerna 1 – 3 , vilka är av samma genre (genre 1), men har inte specificerat vad de tycker om filmerna 4 – 6. Användarna 5 – 7 gillar filmerna 4 – 6 som är av en annan genre (genre 2), men har indikerat att de ogillar filmerna 1 – 3. Användare 4 gillar filmer av båda genrerna.

R har faktoriserats till latenta faktorer i matriserna U och V. Antalet latenta faktorer (kolumner) i matriserna är en parameter som bestäms av den som implementerar rekommendationssystemet.

I matris U visar värdena hur användarna förhåller sig till de olika genrerna, där 1 betyder att användaren gillar genren och -1 att användaren inte gillar genren. I matris V^T visar värdena hur varje film förhåller sig till de olika genrerna, där 1 betyder att filmen hör till genren. Genom att multiplicera matriserna U och V^T (de ospecificerade värdena sätts till 0) kan man approximera R och får uppskattade värden på de element som saknas i R, alltså en uppskattning på hur varje användare skulle betygsätta de artiklar som de inte gett betyg åt [8] [18]. Baserat på de uppskattade betygen kan man rekommendera artiklar åt användarna.

Att bestämma värdena för elementen i de latenta vektorerna i matriserna U och V kan ses som ett optimiseringsproblem. Formellt kan man definiera det enligt följande formel:

$$min\, J\ = \frac{1}{2}||R - UV^T||^2$$

Här representerar $||R-UV^T||^2$ resultatet av matrissubtraktionen $R-UV^T$, och vidare summan av kvadraterna av varje element i resultatmatrisen. Detta kallas *Frobeniusnormen*. Målet är alltså att hitta sådana värden i matriserna U och V att resultatet J för uttrycket blir så litet så möjligt. Detta kan åstadkommas med gradientnedstigning (gradient descent) [8].

3. Fallstudier

I detta kapitel presenteras och jämförs fyra fallstudier: två som handlar om grannbaserade metoder och två som handlar om latent faktor metoder. Studierna har testat sina metoder mot den populära MovieLens datauppsättningen, dock mot varierande versioner av den.

MovieLens har samlat en databas med data om användare, filmer och betyg. Betygen är givna på en skala från 1 till 5. Som prestandamått använde fallstudierna genomsnittligt absolut fel (Mean Absolute Error – MAE), dvs. absoluta felet (verkligt betyg - uppskattat betyg), och vidare medeltalet av alla dessa fel som fås som resultat av metoden i fråga.

3.1 Fall 1 - Ett förbättrat likhetsmått

I kapitel 2.2 diskuterades grannbaserade metoder för samarbetsfiltrering. Grunden för dessa metoder är likhetsmåttet, som definierar avståndet mellan användare och användare, eller mellan artiklar och artiklar i betygsmatrisen. Pearsons korrelationskoefficient är ett populärt exempel, men även många andra mått finns.

Feng et al. [11] ger i sin studie ett förslag till ett förbättrat likhetsmått. De berättar att de märkt att traditionellt använder sig samarbetsfiltrering av mängden av artiklar som två användare båda betygsatt. Detta fungerar dock inte om ett användarpar inte har några gemensamma betygsatta artiklar.

Feng et al.:s likhetsmått består av tre komponenter: S_1, S_2 samt S_3. S_1 är en beräkning av likheten mellan två användare. S_2 är en komponent som bestraffar par av användare som har endast ett fåtal gemensamma betygsatta artiklar. S_3 är en komponent som tar i beaktande användarparets betygsättningspreferens. Likheten mellan användare *u* och användare *v* definieras alltså enligt följande:

$$sim(u,v) = S_1(u,v) \cdot S_2(u,v) \cdot S_3(u,v)$$

För att beräkna S_1 beaktas glesheten i betygsmatrisen, dvs. hur många element som är specificerade och saknas i matrisen. Man väljer en tröskel ρ (ett flyttal mellan 0 och 1 för att beteckna hur många procent av elementen saknas i matrisen), och beräknar likheten mellan användare *u* och *v* enligt följande formel:

$$S_1(u,v) \begin{cases} \dfrac{\sum_{i \in I_u \cap I_v} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in I_u} r_{ui}^2} \cdot \sqrt{\sum_{i \in I_v} r_{vi}^2}}, & då\ glesheten < \rho \\ \dfrac{\sum_{i \in I_u \cap I_v} r_{ui} \cdot r_{vi} + \sum_{i \in I_u - I_u \cap I_v} r_{ui} \cdot \mu_v + \sum_{i \in I_v - I_u \cap I_v} r_{vi} \cdot \mu_u}{\sqrt{\sum_{i \in I_u} r_{ui}^2 + \sum_{i \in I_u \cup I_v - I_u} \mu_u^2} \cdot \sqrt{\sum_{i \in I_v} r_{vi}^2 + \sum_{i \in I_u \cup I_v - I_v} \mu_v^2}}, & gleshet \geq \rho \end{cases}$$

Här betecknar r_{ui} ett betyg r givet av användare u till artikel i. I_u är mängden av artiklar som användare u har gett betyg åt. μ_u är medeltalet av betygen som användare u har gett.

Komponenten S_2 finns för att minska likheten mellan användare som har endast få gemensamma betygsatta artiklar. Detta inkluderas eftersom när antalet gemensamma betygsatta artiklar mellan två användare är stort, så fås det mera korrekt information om användarnas likhet. S_2 definieras av följande formel:

$$S_2 = \frac{1}{1 + \exp\left(-\frac{|I_u \cap I_v|^2}{|I_u| \cdot |I_v|}\right)}$$

Exp(n) betyder e^n. $|I_u|$ betyder ”antalet element i mängden I_u”. När antalet gemensamma betygsatta artiklar mellan användare u och v är litet, så blir uttrycket S_2 mindre, och som resultat blir även totala likhetsmåttet sim(u,v) mindre.

Den tredje komponenten S_3 tar i beaktande användarnas allmänna betygsättningspreferens, dvs. det faktum att vissa användare brukar allmänt ge höga betyg medan andra brukar allmänt ge låga. S_3 definieras av följande formel:

$$S_3(u,v) = 1 - \frac{1}{1 + \exp\left(-|\mu_u - \mu_v| \cdot |\delta_u - \delta_v|\right)}$$

Här betecknar δ_u variansen för betygen givna av användare u, exp(n) betyder e^n, μ_u är medeltalet för betygen givna av u och |n| ger absolutbeloppet för uttrycket.

Då man har bestämt komponenterna S_1, S_2 och S_3 kan man räkna ut likhetsmåttet sim(u,v) mellan alla par av användare. Därefter kan man beräkna ett uppskattat betyg $\hat{r}_{ui}$ av användare u till artikel i (som användare u inte har gett ett betyg åt) enligt formeln:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in NN_u} sim(u,v) \cdot (r_{vi} - \mu_v)}{\sum_{v \in NN_u} |sim(u,v)|}$$

där NN_u är mängden av närmaste grannar till användare u.

3.2 Fall 2 – Imputeringsteknik för data som saknas

I den andra fallstudien [19] har Ren et al. erbjudit en alternativ metod för att förbättra noggrannheten i grannbaserad samarbetsfiltrering. Istället för att jobba på själva likhetsmåttet, undersöker de en lösning till ett av grundproblemen i grannbaserad samarbetsfiltrering, nämligen glesheten i betygsmatrisen. Då endast ett fåtal av elementen är specificerade överlag, så är det svårt att formulera ett likhetsmått som noggrant uppskattar likheten mellan användare.

Som lösning föreslår Ren et al. en teknik som baseras på dataimputering. Istället för att beräkna likheten på basis av endast specificerade element, så hittar en algoritm s.k. nyckelgrannar till den användare som processeras, och fyller in artificiella betygsättningar till de artiklar som det saknas ett betyg åt. Då kan man göra en bättre uppskattning då man beräknar vilket betyg en användare skulle ge åt en artikel, givet att imputeringen kan simulera verkliga betygsättningar.

Ren et al. säger att alla data som saknas i betygsmatrisen har inte likvärdig information när det gäller att beräkna ett uppskattat betyg från användare u till artikel t. Därför är det viktigt att identifiera en nyckelmängd av data som saknas i relation till u.

För att göra en prediktion till artikel t_s för användare u_a, så definierar Ren et al. först mängden av relaterade användare till u_a som U_a, och mängden av artiklar relaterade till artikel t_s som T_s, formellt:

$$U_a = \{u_{a'} | r_{a's} \neq \emptyset\}$$
$$T_s = \{t_i | t_i \subset [T_{aa'_1} \cup \ldots \cup T_{aa'_i} \cup \ldots \cup T_{aa'_l}]\}$$

dvs. en användare $u_{a'}$ är en användare relaterad till u_a, sådan att $u_{a'}$ har specificerat ett betyg till artikel t_s. $T_{aa'_i}$ är mängden av artiklar som både användare u_a och $u_{a'_i}$ har gett ett betyg åt, antalet sådana mängder är l, vilket är antalet element i mängden U_a. Till exempel om användarna u_a och $u_{a'}$ har specificerat följande betyg: $u_a = [r_{a1}, 0,0, r_{a4}, r_{a5}]$, $u_{a'} = [r_{a'1}, r_{a'2}, 0,0, r_{a5}]$ där r är ett betyg och 0 är ett betyg som saknas, så är $T_{aa'} = [t_1, t_5]$ eftersom användarna har de gemensamma betygsatta artiklarna 1 och 5. Mängden av relaterade artiklar T_s är unionen av alla dessa mängder $T_{aa'}$ för varje användare $u_{a'} \in U_a$.

När dessa data har definierats, så bestäms mängden av *nyckelgrannar* $N_{a,s}$ till användare u_a med avseende på artikel t_s enligt:

$$N_{a.s} = \{r_{a'i} | u_{a'} \in U_a, t_i \in T_s\}$$

Nyckelgrannarna är alltså sådana betyg att användaren som den tillhör hör till U_a och artikeln den gäller tillhör T_s. Man bör märka att $r_{a'i}$ kan vara ett specificerat betyg eller så kan det vara ett betyg som saknas. För de betyg som saknas för nyckelgrannarna så beräknas dess värde enligt följande formel:

$$\hat{r}_{a'i} = \mu_{a'} \frac{\sum_{u_x \in N_k(u_{a'})} sim(u_{a'}, u_x) \cdot (r_{xi} - \mu_x)}{\sum_{u_x \in N_k(u_{a'})} sim(u_{a'}, u_x)}$$

$Sim(u_{a'}, u_x)$ är likhetsmåttet mellan användarna $u_{a'}$ och u_x (u_x är en av $u_{a'}$:s k närmaste grannar), vilket kan beräknas enligt Pearsons korrelationskoefficient. Efter att värdet har beräknats imputeras det in i betygsmatrisen, då kan man beräkna ett uppskattat betyg åt användare u_a för artikel t_s enligt formlerna från kapitel 2.2.

3.3 Fall 3 – Sannolikhetsbaserad matrisfaktorisering

Latent faktor metoden som baserar sig på matrisfaktorisering har visat sig ge goda resultat för att uppskatta betyg. Metoden grundar sig på antagandet som gäller överlag för samarbetsfiltrering: det finns grupper av användare som betygsätter

artiklar på liknande sätt. Om betygsmatrisen är en $m \times n$ matris, så kan man representera dessa grupper av användare i en $m \times k$ matris, där k är ett antal faktorer som man antar att användarna kan delas in i, och vidare kan man representera artiklarna med en $n \times k$ matris. Om vi kallar dessa matriser U och V, så kan man uppskatta betygsmatrisen R genom att multiplicera matriserna: $R \approx UV^T$.

Klassisk matrisfaktorisering har dock några nackdelar. Om man inte lägger regler på hur matriserna faktoriseras, så kan elementen i U och V vara hurdana som helst reella värden, även negativa värden. Detta gör det svårt att tolka resultaten när latenta vektorerna multipliceras till betyguppskattningar.

Hernando et al. [20] presenterar ett förbättrat alternativ till klassisk matrisfaktorisering. I deras modell delas betygsmatrisen upp i K stycken latenta faktorer (som i klassisk matrisfaktorisering), men här representerar användarnas vektorer grupper som har samma smak (betygsätter artiklar på liknande sätt), och artiklarnas vektorer representerar hur varje grupp brukar betygsätta artikeln i fråga. Till exempel om användare u representeras av latenta vektorn $\overrightarrow{a_u} = (a_{u1} \dots a_{uK})$, så är varje komponent a_{uk} i vektorn sannolikheten att användare u hör till grupp k. Alltså gäller att $\sum_{i=1}^{K} a_{uk} = 1$.

För att bestämma matriserna för de latenta vektorerna använder Hernando et al. täthetsfunktioner. Betrakta följande formel:

$$q(\overrightarrow{\phi_u}, K_{i,k}, z_{u,i}) = \prod_{u=1}^{N} q_{\overrightarrow{\phi_u}}(\overrightarrow{\phi_u}) \prod_{1}^{M} \prod_{1}^{K} q_{K_{i,k}}(K_{i,k}) \prod_{r_{ui} \neq \emptyset} q_{z_{u,i}}(z_{u,i})$$

Till näst förklaras termerna i formeln:

$q_{\overrightarrow{\phi_u}}(\overrightarrow{\phi_u})$är en fördelning som betecknas: $q_{\overrightarrow{\Phi_u}}(\overrightarrow{\phi_u}) \sim Dir(\gamma_{u,i} \dots \gamma_{u,K})$ vilket betyder att värdena för komponenterna i vektorn $\overrightarrow{\phi_u}$ tas ur Dirichlets täthetsfunktion. Värdena på komponenterna i vektorn $\overrightarrow{\phi_u}$ representerar sannolikheten att användare u tillhör en viss grupp. Formen för Dirichlets täthetsfunktion bestäms av parametrarna $\gamma_{u,i} \dots \gamma_{u,K}$, målet är att hitta optimala värden på dessa med en algoritm.

$q_{K_{i,k}}(K_{i,k})$ är en fördelning som betecknas: $q_{K_{i,k}}(K_{i,k}) \sim Beta(\varepsilon_{i,k}^{+}, \varepsilon_{i,k}^{-})$. $K_{i,k}$ betecknar sannolikheten att en användare i grupp *k* gillar artikel *i*. Värdet för variabeln K tas ur Betafördelningen (en sannolikhetsfördelning/täthetsfunktion), som bestäms av parametrarna $\varepsilon_{i,k}^{+}, \varepsilon_{i,k}^{-}$. Målet är att hitta optimala värden på dessa parametrar med en algoritm.

$q_{z_{u,i}}(z_{u,i})$ är en fördelning som betecknas: $q_{z_{u,i}}(z_{u,i}) \sim Cat(\lambda_{u,i,1} \dots \lambda_{u,i,k})$. $z_{u,i}$ är en variabel för vilken ett värde *k* betyder att användare *u* skulle betygsätta artikel *i* på samma sätt som andra användare i grupp *k* gör. Värdet för $z_{u,i}$ fås ur en kategorisk täthetsfunktion som bestäms av parametrarna $\lambda_{u,i,1} \dots \lambda_{u,i,k}$. Målet är att hitta optimala värden på dessa parametrar med en algoritm.

Parametrarna beror på varandra, och värdet för dem fås med hjälp av koordinatsuppstigningsalgoritmen (coordinate ascent algorithm). När värden på parametrarna bestämts, så definierar Hernando et al. värdena för komponenterna i latenta vektorerna enligt följande formler:

$a_{u,k} = P(användare\ u\ hör\ till\ grupp\ k) = \frac{\gamma_{u,k}}{\sum_{j=1\dots K} \gamma_{u,j}}$ dvs. sannolikheten att användare u hör till grupp k.

$b_{k,i} = P(anv.u\ gillar\ artikel\ i | u\ hör\ till\ grupp\ k) = \frac{\varepsilon_{i,k}^{+}}{\varepsilon_{i,k}^{+} + \varepsilon_{i,k}^{-}}$ dvs. sannolikheten att användarna i grupp k gillar artikel i.

En uppskattad sannolikhet för att användare u gillar artikel i beräknas enligt ekvationen:

$p_{ui} = \sum_{k=1}^{K} a_{u,k} b_{k,i}$ där K är antalet grupper som har bestämts av den som implementerar algoritmen att man letar efter. Sedan kan sannolikheten översättas till ett uppskattat betyg, exempelvis genom följande schema (på en betygsskala 1 – 5):

$$\hat{r}_{ui} = \begin{cases} 0 \le p_{ui} \le 0.2 \rightarrow 1 \\ 0.2 \le p_{ui} \le 0.4 \rightarrow 2 \\ 0.2 \le p_{ui} \le 0.6 \rightarrow 3 \\ 0.6 \le p_{ui} \le 0.8 \rightarrow 4 \\ 0.8 \le p_{ui} \le 1 \rightarrow 5 \end{cases}$$

3.4 Fall 4 – Matrisfaktorisering med Linked Open Data

Linked Open Data är ett projekt som har som mål att organisera och samla ihop öppna datauppsättningar. I och med att allt flera data samlas, så kan det hända att när man behöver data för ett projekt, så finns det en lämplig färdig datauppsättning. Det kan vara svårt att veta var man ska leta bland alla uppsättningar; Linked Open Data (LOD) är ett projekt som länkar till öppna datauppsättningar.

Natarajan et al. har i en publikation [21] använt sig av LOD för att skapa noggranna rekommendationsalgoritmer. Specifikt använder hänvisar de till DBpedia, som är en datauppsättning som representerar data från Wikipedia och andra Wikimediaprojekt; dessa data utnyttjar Natarajan med sina kolleger i sitt arbete. Data på DBpedia representeras i form av 3-tupler av strängar. Man kan utföra frågor mot DPpedias databas med frågespråket SPARQL, som kunde jämföras med frågespråket SQL i sitt syfte.

Data från DBpedia utnyttjas i beräkning av likhetsmått i Natarajans och kollegernas metod. För sitt likhetsmått använder de två delar, en modifierad version av Pearsons korrelationskoefficient, och ett likhetsmått baserat på data från DBpedia. Deras modifierade version av Pearsons korrelationskoefficient mellan artikel p och artikel q definierar de som:

$$Pearson(p,q)_{modifierad} = \max\left(0, Pearson(p,q) \cdot \frac{\min\left(|M(p) \cap M(q)|, \Delta\right)}{\Delta}\right)$$

var Δ representerar en tröskel för antalet användare som har betygsatt artiklarna p och q, och M(p) och M(q) är mängderna av användare som har betygsatt artiklarna p och q.

Den andra delen av likhetsmåttet tar i beaktande en artikels semantiska särdrag från databasen DBpedia. Natarajan et al. beskriver inte sim^{PICSS}, som är ett likhetsmått mellan två artiklar baserat på semantiska särdrag uttagna från DBpedia, i detalj. De berättar dock att det är tidskrävande att beräkna likhet baserat på alla semantiska särdrag som DBpedia erbjuder, därför tar de i beaktande endast de viktigaste särdragen med hjälp av principalkomponentanalys (PCA). Sedan definierar de det slutliga likhetsmåttet som summan av de två hittills nämnda delarna:

$$LODsim(p,q) = Pearson(p,q)_{modifierad} + sim^{PICSS}(p,q)$$

Uppskattat betyg $\hat{r}_{ui}$ av användare u till artikel i beräknas sedan enligt formeln:

$$\hat{r}_{ui} = \mu + b_u(i) + b_t(j) + \left(q_j + |C^k(j)|^{-\frac{1}{2}} \sum_{V_h \in C^k(j)} X_h\right)^T \cdot \left(t_i + |D(i)|^{-\frac{1}{2}} \sum_{u_l \in D(i)} Y_l\right)$$

μ är medeltalet av betygen för artikel i. $b_u(i)$ och $b_t(j)$ är biasvärden för en användare i till artikel j. Detta betyder att om en artikel är en film, exempelvis Titanic, så kunde medeltalet av betygen för Titanic vara 3.2, men användare *i* betygsätter i regel artiklar 0.3 högre än en genomsnittsanvändare, och användare *i* gillar i regel filmer av samma typ som Titanic och betygsätter dessa 0.2 högre än andra filmer. Dessa tendenser modelleras av $\mu+b_u(i)+b_t(j)$, i detta fall skulle det bli 3.2+0.3+0.2.

Komponenten q_j är latenta vektorn för artikel j och t_i är latenta vektorn för användare i. $C^k(j)$ är mängden av k stycken närmaste grannar till artikel j, dvs. de artiklar som har största värdet enligt funktionen LODsim(p,q). X_h är vektorn av semantiska särdrag för varje artikel i $C^k(j)$. D(i) är mängden av artiklar som användare i har betygsatt, och Y_l är vektorn av semantiska särdrag för varje artikel i D(i).

Värden för komponenterna bestäms genom att minimera en optimeringsfunktion. Detta åstadkommer man genom att köra träningsdata från betygsmatrisen genom stokastiska gradientnedstigningsalgoritmen, vilket minimerar en optimeringsfunktion. På detta sätt lär sig modellen vilka värden på komponenterna ger resultatet från träningsdata, och kan generalisera denna information till osedda betyg.

4. Jämförelse av metoder

Fallstudierna som beskrevs i föregående kapitel presenterade unika lösningar och förbättringar till metoder inom samarbetsfiltrering. För att jämföra dem är ett alternativ för prestandamått genomsnittligt absolut fel (mean absolute error, MAE) på de data som metoderna är testade.

FALLSTUDIE	**MAE**
FALL 1 [11] – FÖRBÄTTRAT LIKHETSMÅTT	0,73
FALL 2 [19] – IMPUTERINGSTEKNING	0,7172
FALL 3 [20] – SANNOLIKHETSBASERAD MATRISFAKTORISERING	0,664
FALL 4 [21] – LINKED OPEN DATA MATRISFAKTORISERING	0,670

Tabellen ovan visar bästa MAE-värdet som varje fall rapporterade mot MovieLens datauppsättningen. Användarna har i MovieLens betygsatt artiklar (filmer) på en skala från 1 till 5. MAE-värdet skall alltså tolkas som hur mycket ett uppskattat betyg från metoden i fråga genomsnittligt skiljer sig från en verklig betygsättning (lägre värde är bättre). Genom att betrakta resultaten naivt kunde man dra slutsatsen att latent faktor metoder (fall 3 och 4) är entydigt bättre än grannbaserade metoder (fall 1 och 2).

En brist i den slutsatsen är dock att fallen utförde sina experiment mot olika versioner av MovieLens. Ett fall hade en datamängd med 100 000 givna betyg, medan ett annat hade en datamängd med 20 miljoner givna betyg. Storlekarna på betygsmatriserna varierade också. Eftersom alla metoder uppskattar en användares preferenser utgående från andra användares betyg, så påverkas resultatet bl.a. av hur många gemensamma betygsatta artiklar varje användare har med de andra användarna. Därför kan man inte direkt dra slutsatsen att t.ex. fall 3 skulle vara 9% noggrannare än fall 1 såsom tabellen visar. Det är dessutom värt att notera att fall 1 inte nödvändigtvis är menad att vara noggrannare än de andra fallen ensam, utan är menad att vara noggrannare än andra likhetsmått, och menad att användas i samband med andra metoder. Dock är trenden i dessa fall, och i litteraturen överlag, att latent faktor metoder brukar ge noggrannare resultat än grannbaserade metoder.

Trots att latent faktor metoder brukar ge noggrannare rekommendationer än grannbaserade metoder, så har grannbaserade metoder några fördelar. Grannbaserade metoder är intuitiva och lätta att implementera, och det är lätt att förstå hur de fungerar. Det är också ofta lätt att tolka varför de rekommenderar som de gör. Tolkbarheten av resultaten är dock något som Hernando et al. i fall 3 har tagit extra hänsyn till. I deras modell är resultaten lättare att tolka än i typiska latent faktor modeller [20], då modellen beräknar först hur sannolikt det är att användare u hör till grupp k, och sedan hur sannolikt det är att grupp k gillar artikel i. Tolkbarheten är en viktig faktor, eftersom det ligger i mänsklig natur att vilja förstå varför man blir rekommenderad en viss artikel.

Fall 4 har en speciell fördel när det gäller kallstartsproblemet (cold start problem). Kallstartsproblemet är då en ny användare eller artikel införs i betygsmatrisen, då finns det inga betyg att utgå ifrån och det är omöjligt att ge noggranna rekommendationer endast på basis av likhetsmått eller latenta faktorer. Metoden i fall 4 hämtar data om en ny artikel från en öppen databas, och kan därför beräkna likheten mellan en artikel som inte har några betyg och andra artiklar.

En aspekt som ingen av fallen nämner är beräkningstids- och lagringsutrymmeskrav. Då betygsmatrisen är stor (miljontals användare och/eller artiklar) blir det viktigt att ha effektiva modeller, så att rekommendationssystemet kan rekommendera så snabbt som möjligt. Fall 2 (imputeringsteknik) nämner att

nyckelgrannar och imputerade värden beräknas på nytt för varje enskild betyguppskattning; det blir beräkningsmässigt dyrt för en stor betygsmatris där många värden saknas.

5. Slutsats

Rekommendationssystem har en stor påverkan på samhället, både ur ekonomisk och allmännyttig synvinkel. I denna avhandling har rekommendationsalgoritmer inom samarbetsfiltrering undersökts. Teori om skillnader mellan grannbaserade och modellbaserade metoder presenterades, samt fallstudier med förbättringar till grannbaserade metoder eller latent faktor metoder.

Resultatet av studien verkar vara att bägge klasserna av samarbetsfiltrering (grannbaserade vs modellbaserade) har egna för- och nackdelar. Grannbaserade metoder är populära tack vare deras intuitiva natur, vilket gör dem enkla att implementera. Dock bör man beakta att dessa metoder har stor utrymmeskomplexitet; för att göra snabba rekommendationer behöver likhetsmått mellan alla par av användare respektive artiklar lagras. Det krävs också en skild ”offlinefas” för att beräkna dessa likheter. När likheterna väl är beräknade och lagrade, kan systemet snabbt reagera på nya data. Exempelvis i ett artikelbaserat system, då en användare ger betyg åt en artikel, så kan systemet omedelbart hitta andra artiklar med höga likhetsmått.

Modellbaserade metoder (speciellt latent faktor metoder) har med tiden blivit allt mera populära, eftersom de har visats ge noggranna resultat. De är dock ofta mera komplicerade till sin natur, deras resultat kan vara svåra att tolka, och de har svårare att reagera på nya data.

I praktiken har dock skillnaden mellan grannbaserad och modellbaserad samarbetsfiltrering blivit allt suddigare. Koren & Bell [15] beskriver i sin artikel hur man kan kombinera element från bägge kategorier för att nå resultat som har styrkor från vardera kategorin: noggrannheten av latent faktor metoder, med flexibiliteten av grannbaserade metoder.

Överlag är det sannolikt att stora, kommersiella, långt utvecklade rekommendationssystem är hybridsystem, med element från grannbaserade

metoder och modellbaserade metoder. I och med att neurala nätverk har blivit populära, forskas idag även användning av djupinlärning för rekommendationssystem. I denna avhandling beskrevs algoritmer som fokuserade på att uppskatta betyg utgående från explicita betyg; i praktiken är även implicita betyg mycket viktiga, eftersom betygsmatriserna kan typiskt ha 99% av explicita betygen ospecificerade.

Denna avhandling svarade på frågor om grunderna i rekommendationssystem, med ett fokus på samarbetsfiltrering. Under processen uppkom möjligheter för nya forskningsfrågor. Vidare forskning kunde gå in på detaljer om samspelet mellan grannbaserade och modellbaserade metoder, rollen av implicita betyg för noggranna rekommendationer, eller undersöka hur man kan kombinera samarbetsfiltrering med innehållsbaserad filtrering.

Referenser

[1] Amazon, ”Amazon,” [Online]. Available: https://www.amazon.com/. [Använd 16 februari 2021].

[2] S. Wojcicki, ”YouTube,” 14 februari 2020. [Online]. Available: https://blog.youtube/news-and-events/youtube-at-15-my-personal-journey. [Använd 16 februari 2021].

[3] L. LI, *Economics of Recommender Systems in Online Marketplaces,* PhD Thesis, 2017.

[4] C. Thompson, ”If you liked this, you're sure to love that,” *The New York Times Magazine,* 21 november 2008.

[5] S. G. Paul Lamere, ”Oracle.com,” 2008. [Online]. Available: https://www.oracle.com/technetwork/systems/ts-5841-159144.pdf. [Använd 16 februari 2021].

[6] A. C. Rivera, M. Tapia-Leon och S. & Lujan-Mora, ”Recommendation systems in education: A systematic mapping study,” i *International Conference on Information Technology & Systems,* 2018.

[7] F. Gräßer, S. Beckert, D. Küster, S. Abraham, H. Malberg, J. Schmitt och S. & Zaunseder, ”Neighborhood-based Collaborative Filtering for Therapy Decision Support,” i *HealthRecSys@ RecSys,* Online, 2017.

[8] C. C. Aggarwal, Recommender systems (Vol. 1)., Springer International Publishing., 2016.

[9] N. Manouselis, H. Drachsler, K. Verbert och E. Duval, Recommender Systems for Learning, Springer, 2012.

[10] X. Guan, C. Li och Y. Guan, ”Matrix Factorization With Rating Completion: An Enhanced SVD Model for Collaborative Filtering Recommender Systems,” *IEEE Access,* vol. 5, pp. 27668-27678, 2017.

[11] J. Feng, X. Fengs, N. Zhang och J. Peng, ”An improved collaborative filtering method based on similarity,” *PLoS ONE,* vol. 13(9), nr e0204003, 2018.

[12] A. S. Lampropoulos och G. A. Tsihrintzis, Machine Learning Paradigms. Applications İn Recommender Systems, Switzerland: Springer International Publishing., 2015.

[13] X. Luo, Y. Xia, Q. Zhu och Y. Li, ”Boosting the K-Nearest-Neighborhood based incremental collaborative filtering,” *Knowledge-Based Systems,* vol. 53, nr 0950-7051, pp. 90-99, 2013.

[14] H.-M. Wang och G. Yu, ”Personalized recommendation system K-neighbor algorithm optimization,” i *Proceedings of the 1st International Conference on Information Technologies in Education and Learning*, Atlantis Press, 2016, pp. 105-108.

[15] Y. Koren och R. Bell, ”Advances in collaborative filtering,” i *Recommender systems handbook*, Springer, 2015, pp. 77-118.

[16] Y. E. M. E. Alami, E. h. Nfaoui och O. E. Beqqali, ”Improving Neighborhood-Based Collaborative Filtering by A Heuristic Approach and An Adjusted Similarity Measure,” i *Proceedings of the International Conference on Big Data, Cloud and Applications,* Tetuan, Morocco, 2015.

[17] N. Sivaramakrishnan, V. Subramaniyaswamy, S. Arunkumar, A. Renugadevi och K. Ashikamai, ”NEIGHBORHOOD-BASED APPROACH OF COLLABORATIVE FILTERING TECHNIQUES,” *International Journal of Pure and Applied Mathematics,* 2018.

[18] A. Mongia, N. Jhamb, E. Chouzenoux och A. Majumdar, ”Deep latent factor model for collaborative filtering,” *Signal Processing,* vol. Volume 169, nr ISSN 0165-1684, 2020.

[19] Y. Ren, G. Li, J. Zhang och W. & Zhou, ”The efficient imputation method for neighborhood-based collaborative filtering,” i *Proceedings of the 21st ACM international conference on Information and knowledge management*, Maui, Hawaii, 2012, October.

[20] A. Hernando, J. Bobadilla och F. Ortega, ”A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model,” *Knowledge-Based Systems,* vol. 97, nr 0950-7051, pp. 188-202, 2016.

[21] S. Natarajan, S. Vairavasundaram, S. Natarajan och A. H. Gandomi, ”Resolving data sparsity and cold start problem in collaborative filtering recommender system using Linked Open Data,” *Expert Systems With Applications,* vol. 149, nr 113248, 2020.