Kontextbaserade Rekommendationssystem

Dennis Mattsson, 36138

Kandidatavhandling i Datavetenskap

Handledare: Mats Neovius

Fakulteten för Naturvetenskaper och Teknik

Åbo Akademi

2016

Innehållsförteckning

[1.0 Inledning 4](#_Toc446430357)

[2.0 Vad är ett rekommendationssystem 5](#_Toc446430358)

[3.0 Design av rekommendationssystem 6](#_Toc446430359)

[3.1 Kollaborativ Filtrering 6](#_Toc446430360)

[3.2 Innehållsbaserat System 7](#_Toc446430361)

[3.3 Kunskapsbaserat och Demografiskt System 8](#_Toc446430362)

[3.4 Hybridsystem 9](#_Toc446430363)

[4.0 Kontextbaserade Rekommendationssystem 10](#_Toc446430364)

[4.1 Definition av Kontext 10](#_Toc446430365)

[4.2 Modellering av kontextbaserad information (förbättra rubrik, skriv om kapitlet efter diskussion med Mats) 11](#_Toc446430366)

[5.0 Implementering av kontextuell information 12](#_Toc446430367)

[5.1 Kontextuell Förfiltrering 13](#_Toc446430368)

[5.2 Kontextuell postfiltrering 14](#_Toc446430369)

[5.3 Kontextuell Modellering 15](#_Toc446430370)

[6.0 Diskussion 16](#_Toc446430371)

[7.0 Sammanfattning 16](#_Toc446430372)

[8.0 Källförteckning 16](#_Toc446430373)

# 1.0 Inledning

Det är allt vanligare att man istället för att söka någon specifik video att se på, produkt att köpa eller musik att lyssna på bläddrar igenom utbudet tills man hittar något som fångar ens uppmärksamhet. Problemet i dag är att utbudet ofta är extremt stort, redan Spotify har över 30 miljoner ljudklipp att välja mellan [1].

Ett sätt att lösa det här problemet är att implementera ett rekommendationssystem. Den här avhandlingen börjar med att beskriva vad datoriserade rekommendationssystem är och vanliga implementationstekniker för dessa. Detta är nödvändig bakgrundsinformation inför de kapitel där kontextbaserade rekommendationssystem behandlas mera ingående. Syftet med avhandlingen är att ge en överblick över hur rekommendationssystem fungerar och hur kontextuell information kan inkorporeras i dessa för att ge bättre rekommendationer.

# 2.0 Vad är ett rekommendationssystem

Ett datoriserat rekommendationssystem är en samling algoritmer som givet insamlade data söker fram ett antal rekommendationer av produkter åt en användare [2]. Med produkt avser jag de resultat som rekommendationssystemet vill att den specifika användaren ska konsumera. Sådana produkter kan exempelvis vara filmer, tjänster eller webbutiksprodukter. Då produkt används i fortsättningen är det den här specifikationen som avses.

Det första datoriserade rekommendationssystemet utvecklades redan i början av 1990-talet [3]. Detta system sparade kommentarer ihop med dokument, så att man kunde filtrera bort de dokument andra användare tyckte var dåliga eller tråkiga. Det här var den första versionen av kollaborativ filtrering, vilket betyder att användare samarbetade för att hitta de intressanta dokumenten.

Då e-handeln utvecklades ökade även utbudet och alternativen mångfaldigt, vilket ledde till att kunderna hade svårt att hitta de produkter som var lämpliga för dem. Det fanns ett behov av ett system som kunde hitta de produkter som är relevanta just för en specifik kund. Rekommendationssystem visade sig vara utmärkta för denna uppgift.

För att kunna skapa rekommendationer bygger systemet en profil av användaren, baserat på data från användarens tidigare interaktioner med systemet. I dess enklaste form kan dessa data vara betyg en användare gett produkter den tidigare konsumerat.

I mer komplexa system kan även andra data tas i beaktande såsom produkters egenskaper eller i vilken kontext produkten är tänkt att konsumeras. En sådan kontext skulle kunna vara om en film ses på vardag eller veckoslut, och med vilket sällskap. Den grundläggande idén är att profilen kan användas för att hitta de produkter som matchar en användares specifika användarprofil, och rekommendera dem [4].

Rekommendationerna är till nytta både för användaren och den som har hand om systemet [5]. Ett bra rekommendationssystem ökar antalet sålda produkter genom att användare hittar produkter som bättre passar deras behov och således skapas en större efterfrågan. En annan funktion är att man kan rekommendera mindre populära eller nischade produkter för de personer som med större sannolikhet uppskattar dem. Fördelen med detta är att man kan få sälja sådana produkter som användare inte annars skulle ha hittat.

# 3.0 Design av rekommendationssystem

Beroende på systemets krav och informationen tillgänglig kan man implementera ett rekommendationssystem på olika sätt. Burke [2] delar in rekommendationssystem i följande fyra klasser: kollaborativ filtrering, innehållsbaserade system, kunskapsbaserade system och demografiska rekommendationssystem. Utöver dessa nämner också Burke hybridsystem, som är en kombination av två eller flera av dessa system, och kontextbaserade rekommendationssystem. I de följande kapitlen kommer vi kortfattat gå igenom dessa fyra rekommendationssystem och på en abstrakt nivå hur de fungerar. Kontextbaserade rekommendationssystem behandlas mera ingående i följande kapitel.

## 3.1 Kollaborativ Filtrering

*Kollaborativ filtrering* använder sig av de betyg en användare tidigare har gett olika produkter för att beräkna rekommendationer [6]. Eftersom den enda informationen som används är användarnas betygsättningar så behövs ingen specifik användarprofil skapas. Istället representeras data i en användare–produkt-matris där elementen motsvarar de betyg användare *a* gett produkten *p*. Målet är att hitta andra användare med liknande smak, och rekommendera produkter de har tyckt om. Metoden kan även användas för att hitta produkter liknande de användaren tidigare har tyckt om.

För att hitta dessa användare eller produkter används huvudsakligen likhetsalgoritmer (eng. similarity) såsom k-närmaste grannar-algoritmer[7]. Algoritmerna försöker hitta de *k* mest lika användarna från en specifik användare, eller de *k* mest lika produkterna från de produkter användaren tidigare har betygsatt högt. Det enklaste sättet att göra detta på är att använda det euklidiska avståndet mellan användarnas betygsättningar, med formeln:

Det formeln gör är att den mäter avståndet(*d)* mellan betygsättningar (*R)* för alla gemensamt betygsatta produkter för användare *a1* och *a2* i det euklidiska rummet, vilket illustreras i figur 1:

*Figur 1 visar avståndet mellan två användare för två gemensamt betygsatta filmer.*

I figur 1 har en användare gett film A betyget 3 och film B betyget 4, och således fått position (3,5, 3.0) i diagrammet. En annan användare har med sina betygsättningar fått position (4.5, 4.0). Formeln för det euklidiska avståndet kommer i det här fallet motsvara pythagoras sats eftersom det endast finns två dimensioner i diagrammet. Avståndet för de två användarna i figur 1 blir enligt formeln . I exemplet används endast två dimensioner för tydlighetens skull, men formeln gäller även för *n* dimensioner. Det vill säga att formeln gäller för *n* gemensamt betygsatta filmer för två användare.

För att få en likhetsfaktor för detta kan man använda följande formel på resultatet:

Formeln ger som resultat ett tal mellan 0 och 1, och ju närmare 1 resultatet är desto mer lika är användarna [8]. De *k* närmaste grannarna kommer således vara de *k* användarna med högsta värdet från funktionen. För att den här metoden skall ge ett pålitligt resultat behövs en normaliseringsfaktor. Fast en användarehar gett en produkt betyget tre av fem och en annan användare har gett samma produkt fyra av fem så kan de tycka lika mycket om produkten. Detta för att den andra användaren har en tendens att betygsätta sina produkter högre än den första användaren.

Ett sätt att förbigå den här normaliseringen är att istället lägga in betygen för två användares betygsättningar i ett linjediagram, och beräkna pearsons korrelation mellan de två linjerna. Då kommer korrelationskoefficienten vara lika, även om den ena användaren har en tendens att betygsätta sina filmer högre eller lägre [9].

En annan metod som går att använda för att hitta de *k* närmaste användarna är klustrering. Med klustrering menas att man givet en mängd med element försöker gruppera element enligt hur lika det är, det vill säga att de element som är lika varandra placeras i samma grupp. Det finns ett antal algoritmer för detta, med olika fördelar och nackdelar, men idén är att de *k* användare eller produkter som är mest lika varandra finns inom samma kluster [10]. Ett problem med klustrering är att det kan finnas element som inte är nära något kluster. Dessa element kallas brus (eng. noise). Kluster med brus kommer generera sämre rekommendationer än kluster utan brus eftersom bruselementet inte är lika de övriga elementen i samma kluster. Detta är något algoritmen som beräknar klustren måste ta i beaktande för att rekommendationssystem skall ge bra rekommendationer åt alla användare.

En fördel med kollaborativa filtreringsalgoritmer är att de är enkla att implementera, eftersom användare och produkter representeras utan komplexa användarprofiler eller produktmodeller. Fördelen med klustrering är att man inte behöver för varje rekommendation räkna om de närmaste användarna eftersom de finns sparade i klustret. Hur ofta man beräknar om dessa kluster beror på applikationen. Allt eftersom smaken utvecklas hos användarna, kommer klustren att ändras. För produkter som konsumeras snabbt, som exempelvis filmer, kan man behöva beräkna om klustren oftare än för produkter som konsumeras långsamt, som exempelvis böcker. Detta eftersom då filmer konsumeras i snabbare takt än böcker, så får man snabbare in data om vad användaren föredrar.

Ett problem kollaborativa filtreringsmetoder lider av är det så kallade kallstartsproblemet, det vill säga att de behöver en del inmatningsdata innan de kan producera bra rekommendationer. Detta gör det också problematiskt att rekommendera nya produkter som läggs in eftersom systemet inte kan rekommendera produkter som inte har fått något betyg. Ett annat problem är att om all information lagras i samma matris kommer denna matris vara enorm och gles med information (eng. sparse), vilket leder till långa beräkningstider och bortkastat lagringsutrymme. Detta kan lösas med matrisfaktoriseringsalgoritmer, det vill säga att man delar upp den stora matrisen i flera mindre matriser som inte är lika glesa. Eftersom kollaborativa filtreringsmetoder endast ser på betyg användare har gett kan de inte ta i beaktande produkter användare endast visat intresse för, vilket också är en nackdel.

## 3.2 Innehållsbaserat System

Ett *Innehållsbaserat system*, till skillnad från kollaborativ filtrering, försöker hitta produkter som bäst motsvarar de produkter användaren tidigare har tyckt om genom att se på produkternas egenskaper [12]. Det här systemet ser endast på användarprofilen och produkters egenskaper för att hitta rekommendationer, det görs inga jämförelser med andra användares betygsättningar. Exempelvis om en användare har sett på mycket komedier kommer användaren också rekommenderas komedier.

En utmaning med det här systemet ligger i representationen av produkternas egenskaper, speciellt då det handlar om produkter som inte har en standardiserad struktur eller som ändras såsom webbsidor och textdokument. Filmer kommer alltid att ha samma egenskaper såsom genre och skådespelare, medan det i en webbutik kan finnas produkter med varierande egenskaper. En annan utmaning är att ”träna” användarprofilen så att den ska motsvara en användares preferenser. Till exempel för en ny användare betyder det inte att användarens favoritgenre är drama fastän användaren valt att i början se på sådana filmer.

Fördelen med ett innehållsbaserat system är att den inte är beroende av övriga användare, eftersom det endast ser på produkter och en given användarprofil. Detta gör att det inte är något problem att introducera nya produkter, eftersom rekommendationssystemet inte är beroende av tidigare betygsättningar. Eftersom systemet baserar sina rekommendationer på produkters egenskaper kan det också förklara varför det gett en viss rekommendation, exempelvis rekommenderas dessa filmer för att de innehåller en användares favoritskådespelare.

Det finns också nackdelar med den här metoden. Ett krav är att det måste finnas tillräckligt med information om produkterna så att systemet kan skilja på produkter en användare tycker om och sådana användaren inte tycker om. Ett annat problem är att ge pålitliga rekommendationer till en ny användare. Väljer en ny användare att endast se på en viss typ av produkter, exempelvis filmer med Adam Sandler, kommer användaren endast att rekommenderas sådana produkter i framtiden. Vidare om Adam Sandler också skulle spela i dramafilmer, och användaren har sett alla filmer av honom, så måste systemet kunna omvandla detta till parametrar. Till exempel måste systemet väga genre (komedi, drama) mot skådespelare lika Adam Sandler. Att utvinna dessa parametrar och deras vikt är en stor utmaning i innehållsbaserade rekommendationssystem.

I likhet med kollaborativ filtrering, lider den här metoden också av att matrisen som informationen lagras i kan bli gles. Speciellt om det lagras mycket egenskaper med om filmen, eftersom varje egenskap för filmen blir en ny dimension i matrisen. Detta går att lösa med klustrering, det vill säga att liknande filmer grupperas ihop i ett kluster. Dessa kluster kan sedan faktoriseras ihop till mindre informationsglesa matriser.

## 3.3 Kunskapsbaserat

Ett *kunskapsbaserat system* liknar ett innehållsbaserat system så till vida att båda ser på egenskaper hos de produkter som rekommenderas. Skillnaden här är att ett kunskapsbaserat system fokuserar på kunskap om till exempel användarens nuvarande situation och önskemål [13]. För att systemet ska veta vad användaren är ute efter krävs det en viss interaktion mellan systemet och användaren. Den här interaktionen kan se ut som en dialog, eller så kan användaren lägga in restriktioner på vad som skall visas (prisnivå, genre etc). Ett alternativ är att använda respons för att ge rekommendationer. Det vill säga att systemet rekommenderar en produkt, så får användaren antingen acceptera rekommendationen eller specificera att den vill ha en viss egenskap. Användaren kan till exempel svara på rekommendationen med att den söker efter något billigare, så genereras en ny rekommendation med en liknande men billigare restaurang.

## 3.4 Demografiska rekommendationssystem

I ett *demografiskt rekommendationssystem*, till skillnad från kollaborativ filtrering, grupperas användarna enligt deras demografiska tillhörighet. Sådan demografisk information kan vara exempelvis kön, ålder eller nationalitet. Användaren placeras i den grupp närmast sin egen demografiska tillhörighet och rekommenderas produkter de i den gruppen betygsatt högt.

Fördelen med demografiska rekommendationssystem är att de inte behöver annan information om en ny användare förutom dess demografiska information för att kunna ge den rekommendationer. Det behövs inte heller specifik information om produkters egenskaper, som i kunskapsbaserade system.

Nackdelen är viss demografisk information, såsom kön och ålder, är svårt att insamla automatiskt. Wang et al kommer också fram till att endast demografisk information antagligen inte är tillräckligt för att kunna ge bra rekommendationer [14].

## 3.5 Hybridsystem

Ett *hybridsystem* är en kombination av två eller flera av de tidigare nämnda rekommendationssystemen [2]. Idén är att kombinera flera olika tekniker så att de kompletterar varandra, det vill säga att man använder en tekniks styrkor för att täcka en annan tekniks svagheter för att nå ett bättre resultat. Som tidigare nämndes så har kollaborativa filtrerings-system problem med nya produkter, ett problem som inte innehålls-baserade system har. Detta eftersom innehållsbaserade system ser på produkternas egenskaper, och inte tidigare betygsättningar. Genom att kombinera dessa två system försöker man eliminera kallstartsproblemet, då produkternas egenskaper finns tillgängliga från början. Burke nämner flera olika metoder för att implementera ett hybridsystem, men de är utanför den här avhandlingens omfattning. [2]

# 4.0 Kontextbaserade Rekommendationssystem

I det här kapitlet behandlas kontextbaserade rekommendationssystem. Vi börjar med att ge en allmän definition av kontext i rekommendationssystem, och beskriver sedan hur kontextuell information kan modelleras. Följande kapitel går in på hur detta implementeras till ett kontextbaserat rekommendationssystem.

## 4.1 Definition av Kontext

Ett kontextbaserat rekommendationssystem är ett system som använder sig av kontextuell information i sina rekommendationer. För att beskriva vad som innebär med kontextuell information tänker vi först definiera vad som menas med ordet kontext då det handlar om rekommendationssystem. Dey, Abowd och Salber definierar denna kontext som:

Vilken som helst relevant information som kan användas för att beskriva sammanhanget av interaktionen mellan användaren och systemet, även användaren och systemet själva. Kontext är typiskt människors, gruppers och objekts plats, identitet och tillstånd [15].

Eftersom kontext är ett så brett begrepp behövs det en generell definition. Definitionen inkluderar både automatiskt och manuellt insamlade data, då till exempel data om sällskap är svårt att samla in automatiskt.

Nyckelordet i definitionen ovan är ”sammanhang”, det vill säga i vilket sammanhang är användaren i vid interaktion med systemet. Ett sådant sammanhang skulle kunna vara tidpunkten för användningen av systemet eller med vem produkten avses konsumeras. En film som rekommenderas åt en användare som skall se på film med sin flickvän, är inte nödvändigtvis en bra rekommendation åt samma användare i sällskap med sina föräldrar.

Ett annat sammanhang är plats, var användaren befinner sig. Det är en dålig rekommendation ifall den inte är relevant för användarens nuvarande position, det vill säga att det inte finns någon möjlighet att använda sig av rekommendationen. Exempelvis rekommendation av restauranger är ett område där din position är viktig eftersom en användare inte är intresserad av restauranger den inte har en chans att ta sig till.

Det finns många olika sammanhang utöver dessa ett rekommendationssystem kan ta i beaktande, men det lönar sig att hålla dem på en allmän nivå med tanke på skalbarheten av systemet. Istället för att använda veckans alla dagar kan man exempelvis använda vardag och helg. Sådana attributhierarkier beskrivs närmare i kapitel 4.2.

## 4.2 Modellering av kontextbaserad information (förbättra rubrik, skriv om kapitlet efter diskussion med Mats)

De enklaste formerna av rekommendationssystem använder sig av endast betyg givna av användare för att skapa rekommendationer, som till exempel kollaborativ filtrering som beskrevs tidigare. Efter att man fyllt i sin användare–produkt-matris med det data man har tillgänglig, försöker rekommendationssystemet uppskatta betygsfunktionen

R: Användare x Produkt → Betyg

för de (användare, produkt) par som ännu inte har ett betyg. Detta kallas ett traditionellt system eller ett tvådimensionellt system eftersom det endast finns två dimensioner i funktionen, användare och produkt. [16]

Användare och produkt i funktionen är användarprofiler och produktmodeller som var för sig kan ha flera olika attribut, beroende på implementeringen systemet. Användarprofilen skulle exempelvis kunna ha attribut som användar-id, namn, kön och nationalitet beroende på vilken information systemet har om användarna. Produktmodellen beror mycket på vad för produkt det handlar om, en film kan ha attribut om skådespelare eller genre medan en vara från en webbutik kan ha helt andra attribut.

I ett kontextbaserat system lägger man till ytterligare en dimension av information, kontext, i funktionen. Funktionen blir således och se ut som följande:

R: Användare x Produkt x Kontext → Betyg

Här är kontext den kontextuella information systemet samlar in om användaren. Kontext visas i funktionen som endast en dimension, men i praktiken kan det finnas flera än en kontextdimension.

Adomavicius och Tuzhilin ger i [16] ett filmrekommendationssystem som exempel på detta där den kontextuella modellen innefattar tre olika kontextdimensioner där varje attribut är en unik nyckel:

* Biograf: Den biograf som visar filmen, med attributen: biograf-id, namn, adress, kapacitet, stad och land.
* Tidpunkt: Tidpunkten då en film har setts eller kan ses, med attributen: datum, dag i veckan, tid i veckan(vardag/veckoslut), månad, kvartal och år.
* Sällskap: I vilket sällskap filmen har setts eller ska ses, med endast ett attribut: sällskap(ensam/med vänner/partner/familj/medarbetare/övrig).

I det här exemplet kommer en rekommendation variera beroende på när man skall se filmen, med vem och var. Exemplet visar även att det är möjligt att använda fler än en dimension i kontextmodellen. Attributen i sig kan också innehålla en eller flera fördefinierade hierarkiska strukturer. En sådan hierarkisk struktur skulle kunna vara dag→månad→år för tidpunkt-dimensionen, eller biografid→stad→land för biograf-dimensionen. Dessa hierarkiska strukturer är nödvändiga för att man skall kunna göra en generalisation av attributet ifall man inte har tillräckligt med data för en viss nivå.

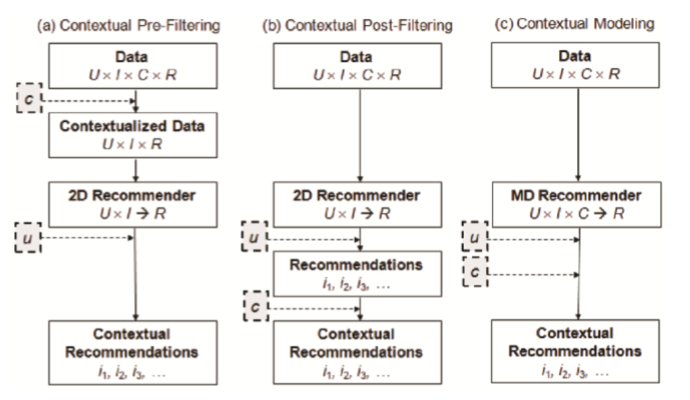
# 5.0 Implementering av kontextuell information

I förra kapitlet beskrevs funktionen för ett traditionellt rekommendationssystem. Funktionen förutsäger vilket betyg användaren skulle ge de produkter som ännu inte har betygsatts, genom exempelvis kollaborativ filtrering. Rekommendationen sker så att systemet hittar de mest liknande användarna och rekommenderar de produkter de betygsatt högt.



Figur 1 visar i vilket skede filtreringen sker för ett traditionellt system(INFOGA KÄLLA recommender systems handbook s.232).

I ett kontextbaserat rekommendationssystem utvecklar man vidare den här modellen till att även ta kontextuell information i beaktande. Detta kan göras på tre olika sätt: genom kontextuell förfiltrering, kontextuell postfiltrering och kontextuell modellering [16]. Se figur 2.



Figur 2 visar tre olika implementationer av kontextuell filtrering (KÄLLHÄNVISNING)

## 5.1 Kontextuell Förfiltrering

I kontextuell förfiltrering använder man den kontextuella informationen för att filtrera bort irrelevanta data innan man lägger in det i ett traditionellt system. Med andra ord om man ser på en produkt en lördag, kommer endast betyg givna på en lördag tas i beaktande. En metod för att göra detta är att reducera problemet till att utföra filtreringen i två steg [17]. Fördelen med den här metoden är att den kan kombineras med vilket som helst traditionellt system.

Problem kan uppstå ifall man är för specifik med kontexten. Det finns inte nödvändigtvis tillräckligt med data för besök på biograf med ett visst sällskap en viss dag. Då lönar det sig att använda en mer generell specifikation, det vill säga att man går högre upp inom någon av attributhierarkierna som beskrevs i föregående kapitel(kan jämföras med OLAP inom datautvinning). Adomavicius et al ger följande exempel, låt oss säga att vi har följande hierarkier inom de kontextuella dimensionernas attribut;

* Sällskap: Partner→vänner→inte ensam
* Plats: Biograf→var som helst
* Tidpunkt: lördag→veckoslut→när som helst

För att generalisera går man ett steg högre inom hierarkierna. Följande problem blir då vilket av attributen man ska välja att generalisera. En metod är att på förhand definiera vilken generalisering som ska ske först, exempelvis att alltid börja med att gå från lördag till veckoslut. En annan metod är att automatiskt beräkna vilken generalisering som ger bäst resultat. Beräkningskostnaden för detta stiger i förhållande till antal attribut, vilket kan göra att detta inte är lönsamt för system med många attribut. Man måste också beräkna att resultatet man får med den kontextuella informationen är bättre än det resultat man får med det traditionella systemet. [16]

En annan metod för kontextuell förfiltrering är mikroprofilering [18]. Istället för att använda en användarprofil per användare delar man profilen i flera mindre profiler beroende på vilken kontext användaren är i. Idén är att använda den profil som bäst passar användarens nuvarande kontext. Utmaningen här ligger i att skapa relevanta mikroprofiler utifrån kontextuella data och i att kombinera dem till en rekommendation.

Det är även möjligt att applicera samma mikroprofileringskoncept på produkten, det vill säga att man istället för användaren delar in produkten i mindre profiler. I den här metoden sker delningen endast ifall betygen märkbart skiljer sig beroende på kontexten. Baltrunas och Ricci visar att metoden ger bättre rekommendationer än traditionella kollaborativa system, förutsatt att man använder relevant kontextuell data [19]. Metoden kan också användas för att förklara varför användaren fått en specifik rekommendation. Till exempel rekommenderas den här resan till detta resmål eftersom där den här årstiden är säsong, vilket har lett till att den har högre betyg den här årstiden. Systemet har då delat in resmålet i två produkter, baserat på årstiden.

## 5.2 Kontextuell postfiltrering

Idén med kontextuell postfiltrering påminner om kontextuell förfiltrering så till vida att man kombinerar metoden med ett traditionellt rekommendationssystem. Skillnaden är att i kontextuell postfiltrering görs filtreringen, eller justeringen av rekommendationslistan, efter att det traditionella rekommendationssystemet har generat sina rekommendationer. Ett sätt ett postfiltreringssystem gör detta på är att fokusera på att hitta samband mellan attribut för en produkt och en viss användare för att justera listan. Adomavicius et al. [16] ger som exempel att en användare som endast ser på komedier på veckosluten endast kommer få komedier som rekommendation på veckosluten. Filmer från övriga genrer har filtrerats bort från den ursprungliga rekommendationslistan. Ett annat sätt att justera listan är att ändra på rankningen baserat på attribut såsom vilka skådespelare är med i filmen. Ju fler av användarens favoritskådespelare som är med, desto högre rankning.

Det går också att implementera en modellbaserad lösning, det vill säga att räkna ut sannolikheten för att användaren kommer att välja en film ur en viss genre i en viss kontext och justera listan enligt det. Då väljer man ett gränsvärde, och antingen helt filtrerar bort de resultat som är under det värdet eller använder det som en vikt kombinerat med det förutsagda betyget för att ranka om listan.

Huruvida kontextuell förfiltrering eller postfiltrering är bättre att använda varierar från applikation till applikation. Panniello et al. skriver ändå att det kan vara kostsamt beräkningsmässigt och arbetsdrygt att hitta den postfiltreringsmetod som är bäst att använda för den specifika applikationen, även om det skulle vara en potentiellt bättre lösning än både kontextuell förfiltrering och ett traditionellt system. [20]

## 5.3 Kontextuell modellering

I kontextuell modellering används den kontextuella informationen direkt i funktionen som genererar rekommendationer. Detta leder till att denna metod använder en flerdimensionell funktion (jämfört med ett traditionellt system som endast använder två dimensioner).

Ett av de enklaste sätten att göra detta på är att bara ta i beaktande betyg som getts inom samma kontext. Detta är mycket lik den kontextuella förfiltreringen, enda skillnaden är att filtreringen sker direkt i beräkningen:

Avståndet beräknas endast om kontextdimensionen är den samma (k = k’), om inte läggs avståndet till , vilket i praktiken betyder att avståndet och därmed likheten inte är beräknad.

Ett annat sätt att göra detta på är att utvidga formeln för det euklidiska avståndet som beskrevs tidigare i kapitel 3.1 till:

Där d1, d2 och d3 är avståndsfunktioner för de olika dimensionerna användare, produkt och kontext [16]. V1, v2 och v3 är vikten given till de olika dimensionerna, det vill säga hur mycket resultatet skall bero på den specifika dimensionen. Noteras bör att de här funktionerna kan utvidgas till flera dimensioner än de tre som visas, då det inte är ovanligt att det finns mer än en kontextdimension i beräkningarna.

# 6.0 Diskussion?

# 7.0 Avslutning

8.0 Källförteckning

|  |  |
| --- | --- |
| [1] | Spotify, ”About, Fast Facts,” 15 02 2016. [Online]. Available: https://press.spotify.com/us/about/. |
| [2] | R. Burke, ”Hybrid Web Recommender Systems,” i *The Adaptive Web*, Springer Berlin Heidelberg, 2007, pp. 377-408. |
| [3] | D. Goldberg, D. Nichols, B. M. Oki och D. Terry, Communications of the ACM, December 1992. [Online]. Available: https://www.ischool.utexas.edu/~i385d/readings/Goldberg\_UsingCollaborative\_92.pdf . [Använd 15 02 2016]. |
| [4] | T. Mahmood och F. Ricci, 2009. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.395.8513&rep=rep1&type=pdf. [Använd 16 02 2016]. |
| [5] | F. Ricci, L. Rokach och B. Shapira, ”Introduction to Recommender Systems,” i *Recommender Systems Handbook*, Springer US., 2011, pp. 1-38. |
| [6] | R. M. Bell och Y. Koren, ”Scalable Collaborative Filtering with Jointly Derived Neighborhood,” i *IEEE International Conference on Data Mining*, 2007. |
| [7] | Y. Koren och R. Bell, ”Advances in Collaborative Filtering,” i *Recommender Systems Handbook*, Springer Science+Business Media, 2011, pp. 145-186. |
| [8] | T. Segaran, i *Programming Collective Intelligence: Building Smart Web 2.0 Applications*, O'Reilly Media, inc, 2007. |
| [9] | G. Adomavicius och A. Tuzhilin, ”Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions,” *IEEE Transactions on Knowledge and Data Engineering,* 2005. |
| [10] | C. C. Aggarwal och C. K. Reddy, Data Clustering: Algorithms and Applications, CRC Press, 2013. |
| [11] | X. Amatriain, *Recommender Systems. Collaborative filtering and other approaches,* Pittsburgh, 2014. |
| [12] | P. Lops, M. d. Gemmis och G. Semeraro, ”Content-based Recommender Systems: State of the Art and Trends,” i *Recommender Systems Handbook*, Springer US, 2011, pp. 73-105. |
| [13] | A. Felfernig och R. Burke, ”ACM Digital Library,” 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1409544. [Använd 18 02 2016]. |
| [14] | Y. Wang, S. Chi-fai Chan och G. Ngai, ”Applicability of Demographic Recommender System to Tourist Attractions: A Case,” i *International Conferences on Web Intelligence and Intelligent Agent Technology*, 2012. |
| [15] | A. K. Dey, G. D. Abowd och D. Salber, ”A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications,” *HUMAN-COMPUTER INTERACTION,* pp. 97-166, 2001. |
| [16] | G. Adomavicius och A. Tuzhilin, ”Context-Aware Recommender Systems,” i *Recommender Systems Handbook*, Springer US, 2011, pp. 217-253. |
| [17] | G. Adomavivius, R. Sankaranarayanan, S. Sen och A. Tuzhilin, ”Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach,” *ACM Transactions on Information Systems,* pp. 103-145, 2005. |
| [18] | L. Baltrunas och X. Amatriain, ”Towards Time-Dependant Recommendation based on,” 2009. |
| [19] | L. Baltrunas och F. Ricci, ”Experimental evaluation of context-dependent collaborative filtering using item splitting,” Springer Science+Business Media, 2013. |
| [20] | U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano och A. Pedone, ”Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems,” ACM New York, 2009. |