Collaborative filtering:
Onderzoek & Implementatie

Greet Dolvelde Joren Six

12 juni 2008

Woord vooraf

Onze dank gaat speciaal uit naar Kunstencentrum Vooruit dat ons de mogelijkheid gegeven heeft deze masterproef uit te werken. De uitwerking van de masterproef is heel interessant om hun bezoekers te leren kennen.

Daarnaast willen wij de heer Cnops en Henk Catry bedanken voor het begeleiden van onze masterproef. Zij gaven kritische opmerkingen waarvan we dankbaar gebruik maakten. Ook dank aan Karen Vander Plaetse en Eddy Naert, die steeds onze vragen beantwoorden. Verder willen we Mathias Timmermans bedanken die ons heeft geholpen met de grafische aspecten van onze poster en ook Wim Roose, die ons technisch bijstond. Ook bedanken we de heer Stoop die voor ons enkele artikelen opzocht.

Speciale dank gaat uit naar onze medestagiairs Anthony Soen, Anthony Van Peteghem en Tom Maeckelberghe. Zij waren een luisterend oor tijdens onze moeilijkere momenten en waren aangename collega's.

Wij richten ook een woord van dank aan onze familie en vrienden die ons steunden doorheen alles en tevens onze masterproef wilden herlezen. Pieter Van Leuven willen we nog eens extra bedanken voor zijn wijze woorden en de momenten van ontspanning.

Joren Six
Greet Dolvelde
12 juni 2008

Samenvatting

In dit werk wordt besproken hoe een aanbevelingssysteem gerealiseerd kan worden, in het bijzonder voor Kunstencentrum Vooruit. Uit het onderzoek blijkt dat Collaborative Filtering (CF) het meest succesvol is bij aanbevelingssystemen voor smaakgebaseerde producten. In de literatuurstudie wordt onder meer de werking en performantie van CF-algoritmen besproken. Ook de praktische kant van de zaak komt aan bod: het Tasteframework, een open-source softwarebibliotheek met CF-algoritmen, wordt samen met andere gebruikte technologieën besproken. Daarna wordt verteld hoe de algoritmen getest kunnen worden en welke metrieken daarbij belangrijk zijn. Er wordt besloten dat de praktische realisatie van het aanbevelingssysteem goede suggesties geeft en dat het systeem gebruiksklaar is.

This paper documents the implementation of a recommender system, especially for Vooruit Arts Centre. Collaborative Filtering (CF) is the most successful approach when recommending taste related products. The different aspects of CF algorithms, like the inner workings and performance, are discussed in an extensive research section. The practical side of the matter is also covered: the application of Taste, an open-source software library with CF algorithms, and other technologies is documented. How to evaluate algorithms and which evaluation metrics matter are described. The conclusion is that the recommender system generates good recommendations and can be used in a production environment.

Sleutelwoorden: Informatie filteren, Collaborative Filtering, Recommender Systems, Information Retrieval.

Inhoudsopgave

Lijst met afkortingen

AUC	Area Under Curve
CF	Collaborative Filtering
MAE	Mean Absolute Error
MVC	Model View Controller
NMAE	Normalized Mean Absolute Error
ROC	Receiver Operating Characteristic
SVD	Singular Value Decomposition

Lijst met figuren

Lijst met tabellen

Inleiding

Iedereen neemt elke dag verschillende beslissingen gaande van “Welke film wil ik zien?” tot “Welke stad zou ik bezoeken?”. We hebben te veel keuze en veel te weinig tijd om alle mogelijkheden te onderzoeken. De uitdaging om een juist antwoord te vinden is eigenlijk hetzelfde als een naald zoeken in een hooiberg. Er zijn niet alleen een ruim aantal mogelijkheden om uit te kiezen, ook variëren deze doorgaans in kwaliteit. Ook weet de gebruiker niet altijd waarnaar te zoeken en als hij/zij dan toch een te grote lijst krijgt van mogelijkheden, is het onmogelijk om eruit te kiezen.

De afgelopen jaren is er een explosieve groei geweest van het volume informatie, wat het probleem nog vergroot. Het aantal boeken, films, nieuws, reclameboodschappen en online informatie neemt exponentieel toe. Het volume ervan is veel te groot om te filteren met als doel die informatie eruit te halen die een persoon verkiest. Een gebruiker wordt als het ware overweldigd met hoeveelheden informatie en kan deze, zonder hulp, moeilijk de baas. Al deze alternatieven evalueren, zal hoe dan ook nog altijd evenveel tijd en werk in beslag nemen. De explosie in informatie is namelijk niet samengegaan met een sprong in de evolutie van de mens. Daarom kunnen mensen niet hopen om alle mogelijke keuzes zelf te evalueren, behalve als het onderwerp heel beperkt is. Op het internet alleen al zijn er miljarden pagina’s te vinden met de meest verscheidene onderwerpen. Ook Usenet Net News groeit exponentieel. Het probleem wordt enkel groter door alle informatie die organisaties online zetten in mappen, databases en dergelijke. Niet enkel is de hoeveelheid informatie die beschikbaar is extreem groot in vergelijking met wat opgevraagd kan worden, maar enkel een kleine fractie ervan is accuraat, up to date en relevant [33]. De gegevens zijn zo niet meer gelimiteerd tot het zoeken op inhoud, maar wel door de kwaliteit van de middelen die er zijn om iets te zoeken.

Mensen proberen deze overdreven hoeveelheid informatie zelf te verwerken, via anderen of via blind geluk [56]. Eerst en vooral wordt er een groot deel van de informatie en de items verwijderd simpelweg omdat ze ontoegankelijk zijn of onzichtbaar voor de gebruiker. Ten tweede wordt een groot deel van het filteren gedaan voor ons: krantredacteurs selecteren bijvoorbeeld welke artikels hun lezers vermoedelijke interessant vinden. Ook zullen boekenwinkels beslissen welke boeken ze aanbieden. Deze barrière zal echter minder en minder meespelen door de huidige tendens waarin elektronische

informatie steeds belangrijker wordt. Ten slotte vertrouwen we op vrienden en anderen (experts) met gelijkaardige smaak om ons dingen aan te raden. Er wordt bijvoorbeeld gerekend op de filmrecensent die zijn/haar mening geeft omtrent een bepaalde film of op de restaurantcriticus. Hierbij wordt het sociale proces van ontmoeten en praten met mensen die dezelfde interesses delen even belangrijk geacht als de aanbevelingen die we verkrijgen. Toch zijn die experts enkel te vertrouwen in hun eigen domein en hebben ze elk hun persoonlijke smaak. Een andere vorm van aanraden via anderen zijn de lijsten die elke week gepubliceerd worden in verschillende bladen of op het internet met daarin de bestsellers. Een mooi voorbeeld van dit alles wordt gegeven in [13]: een eigenares van een boekenwinkeltje raadt een ongekend boek aan aan een klant. Deze raadt het op zijn beurt weer aan aan vrienden, die vervolgens allemaal naar het winkeltje komen om het boek daar te kopen.

Professionele menselijke critici kunnen echter niet alle problemen oplossen. De informatienood en smaken van individuen verschillen genoeg om een aantal recensenten ineffectief te maken [52]. Ook is het aantal documenten op het internet, in onderzoeksbibliotheken en in discussiearchieven te groot geworden om te trotseren. Het is soms heel moeilijk om items te organiseren in een overzichtelijke manier en dus verwachten de bezoekers van de websites betekenisvolle aanbevelingen. Maar zo'n aanbevelingen mogen niet beperkt worden tot de populairste items, die reeds gekend zijn door de gebruiker.

Belangrijk op te merken is dat er een verschil is tussen informatie zoeken en informatie ontdekken. Het zoeken naar informatie betekent het lokaliseren van een gekend object in een bepaalde gegevensbank, zoals via Google. Met informatie ontdekken wordt eerder het verkennen bedoeld van een belovende ruimte van ongekende objecten. Vele tools zijn gekend om informatie te zoeken tegenover slechts luttele ontdekkingsomgevingen. Het zoeken naar informatie is iets algemeen en ongepersonaliseerd, in tegenstelling tot het ontdekken van informatie, waarbij elke klant eigenlijk zijn eigen winkelruimte heeft[1]. Personalisatie is de bekwaamheid om op maat gemaakte inhoud en services aan te bieden aan individuele personen, gebaseerd op kennis over hun voorkeuren en hun gedrag. Het doel ervan is om gebruikers mogelijkheden aan te bieden zonder dat ze het expliciet vragen.

Het probleem dat hiervoor geschetst werd, is ook heel belangrijk bij e-commerce. Bedrijven hebben als doel zoveel mogelijk te verkopen en wat ze kunnen bekomen door hun klanten te helpen die producten te vinden waarin ze mogelijk geïnteresseerd zijn. Hierbij maken ze gebruik van gepersonaliseerde homepages waarop verschillende artikels aangeboden worden, relevant aan vorige aankopen. De hoofdbezigheid van e-commerce

[1]Of zoals Jeff Bezos, CEO van Amazon.com het stelt: "Als ik 3 miljoen klanten heb op het Web, dan moet ik 3 miljoen winkels op het Web hebben.".

sites is de laatste jaren dan ook veranderd in rigoureuze personalisatiestudies om dit te verwezenlijken. De verkoper haalt hier veel voordelen uit, zoals:

- De interacties worden sneller en gemakkelijker.
- Verhogen van de klantloyaliteit.
- Verhogen van de kans op herhaalde bezoeken van gebruikers.
- Maximaliseren van het kijken-om-te-kopen ratio.

Een gekend voorbeeld hiervan is Amazon.com.

Er is technologie nodig om zich een weg te banen door alle informatie om die items te vinden die we echt zoeken en nodig hebben en ons afhelpen van die dingen waardoor we niet belemmerd willen worden. Aanbevelingssystemen (recommender systems) helpen mensen beslissingen te nemen in een complexe informatieruimte. Deze systemen suggereren aan de gebruiker items die ze mogelijk waarderen op basis van kennis over hem/haar en alle mogelijke items. Een nieuwsdienst kan bijvoorbeeld de artikels bijhouden die een gebruiker ooit gelezen heeft. De volgende keer dat deze gebruiker de site bezoekt, zal het systeem nieuwe artikels aanraden op basis van die die hij/zij de vorige keren doornam.

De meest algemene en vanzelfsprekende benadering om het probleem van informatiefilteren aan te pakken is content-based filtering. Content-based filtering is, zoals het woord zegt, filteren op basis van inhoud, zoals bijvoorbeeld op kernwoorden. Hierbij worden items aangeraden voor consumptie bij de gebruiker gebaseerd op correlaties tussen de inhoud van de items en de gebruikersvoorkeuren. Bijvoorbeeld kan het systeem proberen de aanwezigheid van kernwoorden in een artikel te verbinden met de smaak van de gebruiker. Er zijn echter enkele beperkingen, die later vermeld worden, waardoor er gezocht moest worden naar vernieuwende aanbevelingstechnieken.

Om de problemen aan te pakken van content-based filtering, werd er social information filtering [54] voorgesteld, een algemene benadering van gepersonaliseerde informatiefiltering. Social information filtering, tegenwoordig Collaborative Filtering genoemd, automatiseert het proces van 'word-of-mouth' aanbevelingen: items worden aangeraden aan een gebruiker, gebaseerd op waarden toegewezen door andere mensen met een gelijkaardige smaak. Het systeem omschrijft welke gebruikers soortgelijke waarderingen hebben aan de hand van standaardformules voor het berekenen van statistische correlaties.

Deze masterproef is als volgt gestructureerd: eerst wordt het probleem geschetst in Kunstencentrum Vooruit. In hoofdstuk 1 worden enkele bestaande systemen besproken, zoals Amazon.com. Hoofdstuk 2 geeft een overzicht van de soorten aanbevelingssystemen. Het verschil tussen expliciete en impliciete ratings wordt duidelijk gemaakt

in hoofdstuk 3. Hoofdstuk 4 geeft specifieke informatie over Collaborative Filtering zoals hoe dit aangepakt kan worden en wat de voor- en nadelen telkens zijn. De datasets en hun eigenschappen worden behandeld in hoofdstuk 5. Vervolgens wordt Taste besproken, samen met de uitbreidingen en andere technologieën in hoofdstuk 6. Ten slotte worden de verschillende algoritmen geëvalueerd in hoofdstuk 7 aan de hand van enkele metrieken. In bijlage kunnen de documentatie voor Kunstencentrum Vooruit, het masterproefvoorstel en een klassendiagram gevonden worden.

Probleemschets

Elk seizoen worden er een hoop evenementen georganiseerd door Kunstencentrum Vooruit. Het aanbod is heel divers: van concerten en dans tot performance en theater. Dit maakt het moeilijk voor de bezoekers om uit het aanbod te kiezen wat perfect bij hen past. Er zijn natuurlijk wel programmaboekjes beschikbaar en ook wordt alle informatie over de verschillende activiteiten op de website[2] geplaatst.

Om de bezoeker van de website van Kunstencentrum Vooruit te helpen, worden op de verschillende activiteitenpagina's links gelegd naar andere toekomstige activiteiten die gelijkaardig zijn (Figuur 1). Al deze aanbevelingen worden manueel ingegeven door een medewerkster van Kunstencentrum Vooruit, die hier aanzien kan worden als de experte. Dit impliceert natuurlijk dat dit om haar persoonlijke mening gaat en dus onafhankelijk is van de bezoeker van de website. In het manueel toevoegen van aanbevelingen kruipt heel wat tijd en daarmee samengaand kosten (personeelskosten). Om deze kosten te beperken is er de vraag gekomen naar automatisatie van aanbevelingen genereren.

Figuur 1: Huidige manier van aanbevelen bij Kunstencentrum Vooruit

In 2007 lanceerde Kunstencentrum Vooruit als eerste culturele organisatie in Vlaanderen een nieuwe website die volledig op web 2.0-principes geënt is. De gebruiker kan op de communitywebsite onder andere:

- zelf zijn/haar agenda bijhouden met de voorstellingen die hij/zij wil zien of reeds zag;

[2] `http://www.vooruit.be`

- een recensie schrijven of een korte reactie posten;
- foto's, audio en videomateriaal toevoegen bij voorstellingen;
- de afgelopen voorstellingen beoordelingen door er een waarde aan toe te kennen.

Via de communitywebsite kan dus over de verschillende gebruikers en voorstellingen veel informatie opgehaald worden. Naast de communitywebsite is er een ticketingsysteem dat gebruikt wordt in het bespreekbureau (waar tickets gekocht kunnen worden) of via het internet. Ook hier bevindt zich nuttige informatie over de gebruikers.

Kunstencentrum Vooruit heeft zijn eigen visie op het aanraden van *cultuur*. Zo is het voor hen niet prioritair om enkel die voorstellingen aan te raden die zich afspelen binnen Vooruit: ook andere muziekgroepen en dergelijke mogen aangeraden worden. Het doel is voornamelijk cultuur bijbrengen. Heel belangrijk voor Vooruit is het aanraden van vernieuwende dingen. Hiermee wordt bedoeld dat de meeste bezoekers naar die voorstellingen gaan die ze reeds kennen. Toch willen ze iets nieuws ontdekken, maar ze weten niet concreet wat hen nog kan interesseren.

Het is dus voornamelijk de bedoeling om per voorstelling andere voorstellingen aan te bevelen en ook per gebruiker specifiek voorstellingen aan te raden. Tevens moet het mogelijk zijn om nog andere dingen buiten het centrum aan te raden zonder al te veel problemen.

In deze thesis wordt er dieper ingegaan op Collaborative Filtering en de toepasbaarheid ervan in Kunstencentrum Vooruit. Er zullen voorstellingen aanbevolen worden aan de hand van profielen van gebruikers en ook zal er getracht worden op andere manieren cultuur bij te brengen.

Deel I

Onderzoek

Hoofdstuk 1

Bestaande systemen

Om meer inzicht te krijgen in het gebruik van Collaborative Filtering, werden een aantal bestaande systemen bestudeerd: Amazon.com, MovieLens en Last.fm. Hierbij werd vooral gelet op de gebruiksvriendelijkheid en hoe aan de gebruiker gevraagd wordt om verschillende items te beoordelen. Ook werd bekeken welke soorten aanbevelingen er allemaal gebruikt worden. Er zijn nog heel wat andere systemen, zoals Usenetfilters en Doubleclick.net, met heel verschillende doelen, [48] geeft een uitgebreid overzicht. De hierboven vermelde systemen zijn heel uiteenlopend qua doel en methoden die ze gebruiken. Deze drie zijn echter kenmerkend voor het overgrote deel van de CF-systemen.

Amazon.com [2] is een Amerikaans e-commerce bedrijf gevestigd in Seattle, Washington, dat werd opgericht in 1994. Het gebruikt aanbevelingen als marketingtool in vele e-mailcampagnes en op hun websites. Ze gebruiken aanbevelingen om hun website te personaliseren volgens de interesses van elke klant. Hun grootste troef is tegelijk hun grootste probleem: er worden heel wat producten aangeboden en er zijn heel wat klanten. Toch hebben ze een snelle website weten te maken waar de aanbevelingen on-the-fly worden berekend.

MovieLens [17] is een onderzoekswebsite opgestart door GroupLens Research [16] aan de universiteit van Minnesota. MovieLens maakt gebruik van Collaborative Filtering om films aan te bevelen die de gebruikers graag zullen bekijken en zal hen helpen die films te vermijden die ze niet leuk vinden. De verkregen suggesties zijn gepersonaliseerd a.d.h.v. de smaak van de gebruiker, die bepaald wordt door zijn/haar ratings van andere films.

Last.fm [32] werd gecreëerd door het team dat de Audioscrobbler music engine gemaakt heeft. Het werd opgericht door Felix Miller, Martin Stiksel en Richard Jones. Meer dan tien miljoen keer per dag laten gebruikers van Last.fm hun gespeelde muziek naar de servers scrobblen om zo samen werelds grootste sociaal muziekplatform te helpen bouwen. Een liedje scrobblen betekent zoveel als wanneer iemand ernaar luistert, het

verzonden wordt naar Last.fm en vervolgens gevoegd wordt bij zijn/haar muziekprofiel. Last.fm gebruikt de wijsheid van het volk om hieruit voor elke gebruiker gepersonaliseerde aanbevelingen te maken, gebruikers te verbinden die een gemeenschappelijke smaak hebben, ...

Figuur 1.1: MovieLens: startsituatie voor een gebruiker

De startsituatie is bij de verschillende toepassingen anders. Door zelf het initialisatieproces mee te maken, m.a.w. de eerste keer aanmelden bij de verschillende systemen, werd het gevoel gesimuleerd van een nieuwe gebruiker. Zo werd alles ontdekt wat als vervelend ervaren wordt aan een systeem en wat er heel positief aan is. Door de gebruiker bijvoorbeeld op te leggen om direct 15 items te beoordelen, zoals bij MovieLens (Figuur 1.1), wordt er een druk gelegd op hem/haar, waardoor de ratings niet altijd even correct zijn. De gebruiker wil zich hier snel doorheen werken, waardoor ongekende items beoordeeld worden of hij/zij afhaakt. Hiertegenover staan die systemen, zoals Amazon.com, die direct de gepersonaliseerde pagina van de klant tonen en die direct items aanbevelen. Die aanbevelingen zijn dan wel heel algemeen (de best verkochte producten), maar er wordt aangeraden om wat meer profielinformatie vrij te geven. Bij Last.fm is de drempel nog lager: de gebruiker moet weliswaar een programma installeren om liedjes te scrobblen, maar moet vervolgens niets meer doen.

Figuur 1.2: Last.fm: aanbevelingen voor een gebruiker

Door een verklaring te geven van aanbevelingen krijgt een gebruiker meer vertrouwen in een systeem, aangezien de aanbevelingen dan niet uit het niets tot stand komen [19].

Dit wordt toegepast bij Amazon.com en bij Last.fm (Figuur 1.2). Wat ook nog bij een aanbeveling geplaatst kan worden, is de zekerheidswaarde van een suggestie. Zo zal MovieLens bepaalde films aanraden met een bepaalde rating: de rating die dit systeem verwacht dat de gebruiker zal geven aan de film, zoals getoond in Figuur 1.3.

Figuur 1.3: MovieLens: aanbevelingen voor een gebruiker

Elk systeem heeft echter zijn eigen informatiebronnen en een specifieke manier van verwerken. In het helpmenu wordt meestal vermeld hoe Collaborative Filtering eigenlijk werkt. Dit wordt normaal heel simpel voorgesteld zodat de gebruiker toch enige notie heeft van wat erachter zit.

Het te ontwikkelen Collaborative Filteringsysteem voor Kunstencentrum Vooruit zal waarschijnlijk heel dicht liggen bij dat van Amazon.com. Het doel van het systeem is niet enkel verkoop, maar vooral cultuur bijbrengen en nieuwe 'dingen' ontdekken. Bij elke nieuwe gebruiker zullen die items getoond worden die het populairst zijn. Tegelijk zal de gebruiker aangeraden worden enkele items te beoordelen om zo gepersonaliseerde aanbevelingen te verkrijgen. Indien het mogelijk blijkt binnen de beperkte tijdspanne, zal ook bij elke aanbeveling aangegeven worden hoe deze is ontstaan. Een algemene beschrijving van hoe het aanbevelingssysteem werkt, mag zeker en vast niet ontbreken. Hierdoor verstaan de gebruikers namelijk waarom ze best zoveel mogelijk hun profiel aanvullen.

Hoofdstuk 2

Soorten aanbevelingssystemen

Aanbevelingssystemen raden items aan aan gebruikers op basis van hun expliciete en impliciete voorkeuren, de voorkeuren van andere gebruikers en gebruikers- en product-attributen. Bijvoorbeeld zal een filmaanbevelingssysteem expliciete (gebruiker X gaf *Shrek* 7/10) en impliciete data (gebruiker Y kocht *101 dalmatiërs*) combineren met gebruikersdemografische informatie (gebruiker Z is vrouwelijk) en filminformatie (*Finding Nemo* is een animatiefilm) om zo films aan te bevelen. Afhankelijk van het soort aanbevelingssysteem worden slechts een aantal van die eigenschappen gebruikt.

2.1 Content-based Filtering

Content-based Filteringtechnieken raden items aan aan een gebruiker gebaseerd op de correlatie tussen de *content* van de items en de voorkeuren van de gebruiker. Bijvoorbeeld zal het systeem proberen het voorkomen van sleutelwoorden in een artikel te correleren met de smaak van de gebruiker.

Puur content-based of informatiefilteren zal dus enkel gebruik maken van de gegevens van de gebruiker die aanbevelingen wenst en zal die proberen te vergelijken met productinformatie, zonder rekening te houden met gegevens van andere gebruikers.

Content-based Filtering heeft enkele nadelen [54, 30]:

- De items moeten in een interpreteerbare vorm (bv. tekst) staan of er moeten manueel attributen toegewezen worden aan de items. Met de huidige technologie kunnen media zoals geluid, foto's, kunst, video of fysieke items niet automatisch geanalyseerd worden voor relevante attribuutinformatie. Meestal is het praktisch onmogelijk om attributen manueel toe te wijzen door gebrek aan bronnen.

- Content-based Filteringtechnieken hebben geen inherente methode om toevallige bevindingen te genereren. Het systeem raadt voornamelijk items aan die de gebruiker reeds gezien heeft.
- Content-based Filteringmethoden kunnen items niet filteren gebaseerd op kwaliteit of stijl. Ze kunnen bijvoorbeeld geen onderscheid maken tussen goed of slecht geschreven artikels wanneer twee artikels dezelfde termen blijken te bevatten.

Kunstencentrum Vooruit hecht veel belang aan het aanraden van ongekende voorstellingen aan hun bezoekers. Zij willen namelijk iets nieuws ontdekken. Daarom is het onmogelijk om Content-based Filteringtechnieken te gebruiken aangezien deze voornamelijk items aanraden die reeds gekend zijn.

2.2 Collaborative Filtering

Collaborative Filtering of Social Information Filtering overkomt enkele beperkingen van Content-based Filtering. Items die gefilterd worden hoeven niet interpreteerbaar zijn voor een computer. Daarenboven kan het systeem items aanbevelen die heel verschillend zijn van wat de actieve gebruiker voordien heeft aangegeven als zijn/haar smaak. Tenslotte worden aanbevelingen gebaseerd op de kwaliteit van items in plaats van meer objectieve eigenschappen van de items zelf [18].

Collaborative Filtering gebruikt gelijkheden tussen de smaak van verschillende gebruikers om items aan te raden. Het rust op het feit dat de smaak van mensen niet willekeurig is: er zijn algemene trends en patronen in de smaak van een persoon en ook tussen groepen personen. CF automatiseert het proces van "word-of-mouth" aanbevelingen. In plaats van een paar vrienden hun mening te vragen over een aantal items, zal het systeem duizenden mensen in beschouwing nemen evenals items. Het idee is als volgt:

1. Het systeem haalt het gebruikersprofiel op met daarin de interesses van de actieve gebruiker in specifieke items.
2. Het systeem vergelijkt dit profiel met het profiel van andere gebruikers en berekent een similariteitswaarde.
3. Het systeem bepaalt de meest gelijkaardige profielen en gebruikt deze om items aan te raden aan de gebruiker.

Pure Collaborative Filtering zal zijn aanbevelingen baseren op communityvoorkeuren, terwijl gebruikers- en productattributen genegeerd worden.

2.3 Hybride systemen

Hybride systemen laten toe Content-based Filtering en Collaborative Filtering te combineren en zo van elk de beste eigenschappen te gebruiken [1]. Dit kan handig zijn wanneer er bijvoorbeeld te weinig gegevens beschikbaar zijn om CF toe te passen, zoals [39]. Er wordt dus zowel gebruik gemaakt van de voorkeuren van alle gebruikers (de community) als van alle attributen behorend bij de gebruikers en de producten.

Meestal worden hybride systemen gebruikt om Collaborative Filtering aan te vullen met Content-based Filtering. De reden hiervoor is dat er soms te weinig beoordelingen zijn van de gebruikers. Zo zal in [39] inhoudsgebaseerde voorspellingen gebruikt worden om de beoordelingen van de gebruikers aan te vullen met andere beoordelingen. Hierna wordt CF gebruikt om de effectieve aanbevelingen te doen aan de gebruikers.

De meeste hybride aanbevelingssystemen vallen in één van de volgende drie categorieën [26]:

- **Lineair combinatiemodel.** Hierbij worden de resultaten van het Collaborative Filteringsysteem en het content-based systeem gecombineerd. Aan de resultaten van de verschillende systemen kunnen gewichten gehangen worden om zo een op maat gemaakt systeem te bekomen.
- **Sequentieel combinatiemodel.** In dit model worden gebruikersprofielen geconstrueerd door het content-based algoritme gebaseerd op de producten. Hierna wordt een CF-algoritme toegepast om voorspellingen te doen, gebaseerd op de gebruikersprofielen. De kwaliteit van de voorspellingen is hierdoor volledig afhankelijk van de content-based technieken. Inaccurate profielen resulteren in inaccurate correlaties met andere gebruikers zodat hieruit zwakke aanbevelingen ontstaan.
- **Gemengd combinatiemodel.** Hierbij worden zowel semantische inhoud als ratings gebruikt om aan te bevelen.

Er zijn heel wat verschillende hybride systemen beschikbaar. Zo worden in [52] filterbots gebruikt. Deze geven automatisch ratings aan de nieuwe producten. Het CF-systeem behandelt de filterbot als een gewone gebruiker die heel gul is met het beoordelen en geen aanbevelingen verwacht. Bijvoorbeeld zal zo'n filterbot bij een documentaanbevelingssysteem kijken hoe lang een document is en aan de hand hiervan een rating geven. Of deze kan evengoed kijken naar het aantal spellingsfouten die voorkomen in het document. Hiermee worden bepaalde problemen voorkomen die in hoofdstuk 4.2.6 besproken zullen worden.

Hoofdstuk 3

Expliciete t.o.v. impliciete ratings

Om verder in te gaan op Collaborative Filtering is het belangrijk eerst het verschil te kennen tussen expliciete en impliciete ratings. Ratings of beoordelingen kunnen verzameld worden op verschillende manieren. Expliciete ratings worden bekomen door specifieke gebruikersinput: de gebruiker moet op iets klikken of het product beoordelen op een bepaalde schaal. Impliciete ratings daarentegen worden verzameld door het systeem op basis van het gedrag van de gebruiker. De gebruiker merkt hier echter niets van.

3.1 Expliciete ratings

Het gebruik van expliciete ratings is alledaags: studenten worden beoordeeld voor hun werk, producten worden beoordeeld voor hun kwaliteit, ... Hoewel sommige ratings in vrije tekstvorm voorkomen, is het meestal zo dat ratings gegeven worden binnen een vastgelegde schaal. Dit brengt een cognitieve kost mee voor de evaluator, maar dit is zeker geen slechte eigenschap, zo verwacht de gemeenschap dat onze leerkrachten nadenken over de punten die ze geven. De waarde van vele ratingsvormen wordt afgeleid van deze intellectuele inspanning en geeft de motivatie aan voor de verdienste die vele beoordeelde informatiestromen begeleidt.

Expliciete beoordelingen zijn doorgaans numerieke metingen voor elk item, waarbij hoge waarden een sterke interesse voorstellen in dat item en lage waarden een sterke desinteresse. De gebruikers worden over het algemeen belast om items te beoordelen met die waarde waarmee ze het aangeraden wilden zien.

Wanneer expliciete ratings gebruikt worden in social filteringsystemen, worden de kos-

ten en de voordelen duidelijk gepresenteerd. Het beoordelen verandert het normale gedrag van de gebruiker, namelijk het lezen. Nog meer, de voordelen van elke individuele gebruiker die beoordeelt, worden ervaren door andere gebruikers van het systeem. Behalve als de gebruiker ervaart dat er een voordeel verbonden is aan het beoordelingssysteem, zal deze snel de intentie hebben om het te verlaten of om verder te lezen en niet meer te beoordelen. Hierdoor kan er een tekort ontstaan aan ratings [8].

Expliciete benaderingen veronderstellen dat ratings een waarde hebben die onafhankelijk is van de omstandigheden waarin deze werden gegenereerd. Dit is echter niet vanzelfsprekend, aangezien onder andere het humeur van de evaluator al kan zorgen voor een verschillende beoordeling.

3.2 Impliciete ratings

Er zijn verschillende types impliciete data die kunnen gevat worden en bestudeerd. Zo kan er in een nieuwssysteem bijvoorbeeld informatie opgehaald worden over artikels die bij de favorieten gevoegd worden, die gelezen worden of waarop een reactie geformuleerd wordt. Bij een online winkel kan de aankoop, het plaatsen in het winkelmandje of het bekijken van een product als impliciete rating gezien worden. Een impliciete rating is dan ook een voorbeeld van loggebaseerd werken: ofwel doet de gebruiker iets ofwel niet, wat leidt tot ratings nul en één.

De grootste motivatie om impliciete ratings te gebruiken [40] is dat het de kost wegwerkt bij de evaluator om het item te onderzoeken en te beoordelen. Er komt enkel een kleine computationele kost bij om de impliciete rating op te slaan en te verwerken, maar dit wordt verborgen voor de gebruiker. In een netwerkomgeving is het doorgaans heel moeilijk om het verschil te merken tussen netwerkvertraging en extra applicatieverwerking. Aangezien het hoofdprobleem van expliciete ratings de verwervingskost is, zou er een groter aantal impliciete ratings moeten zijn. Potentieel zou elke gebruikersinteractie met het systeem nieuwe impliciete data moeten genereren.

Er zijn vele types van impliciete data die kunnen opgevangen en onderzocht worden. Een aantal worden hieronder opgesomd:

- aankoop van een product
- evaluatie van een product
- bekijken van een product via een webpagina
- ...

Het is mogelijk om impliciete data te halen uit expliciete data, zoals hierboven kan gezien worden via de evaluatie van een product. De gebruiker geeft bijvoorbeeld een

beoordeling op een schaal van één tot vijf en dit wordt omgezet in een impliciete rating. Een expliciete rating van vier of vijf wordt omgezet in een impliciete rating van één, de andere in nul.

Het bezoeken van een webpagina kan op twee manieren bekeken worden. Het simpelweg bekijken van de pagina geeft eigenlijk gewoon aan dat de gebruiker iets meer wil weten over dit product. Dit kan zowel zijn dat hij/zij het product aantrekkelijk vindt of dat het ongekend is. Anderzijds kan de tijd hierbij betrokken worden: de tijd dat een gebruiker kijkt naar een bepaalde pagina. Hoe langer die is, hoe meer hij/zij in het product geïnteresseerd is. Toch is het zo dat een langere pagina langer zal bekeken worden dan een pagina met weinig tekstinhoud. Ook wanneer iemand suggereert om eens een bepaalde pagina te bekijken zal deze gebruiker evenveel tijd hieraan besteden, ook als het product hem/haar niet bevalt [8].

Het is natuurlijk ook mogelijk om aan elke impliciete rating een gewicht toe te kennen. Hoe groter dit gewicht, hoe meer de impliciete rating meetelt. Zo is de aankoop van een product een goede indicator voor hoe graag de gebruiker dit product heeft, terwijl het bezoeken van een webpagina dit al heel wat minder is [44].

Hoofdstuk 4

Collaborative Filteringbenaderingen

Collaborative Filtering omvat een resem aan methoden om producten aan te raden aan gebruikers. Er kan hierbij zowel actief als passief gefilterd worden. Actieve filtering is die waarbij de gebruiker het heft in handen neemt en rechtstreeks aan andere gebruikers producten aanraadt [33]. Dit kan via een pointersysteem waarbij een link en wat informatie gestuurd wordt naar een andere gebruiker om een product aan te raden. De gebruiker die deze informatie ontvangt, kan vervolgens kiezen of deze nuttig is door bijvoorbeeld te kijken of hij/zij de persoon in kwestie vertrouwt. Hiertegenover staat passief filteren waarbij het systeem zelf zal bepalen wat er aangeraden wordt aan wie. Aangezien in Kunstencentrum Vooruit reeds gebruik gemaakt wordt van aanbevelingen via vrienden, wordt niet verder ingegaan op actief filteren.

Collaborative Filteringbenaderingen worden meestal ingedeeld als geheugengebaseerd of modelgebaseerd [59]. In de geheugengebaseerde benadering worden alle ratings opgeslagen zoals ze zijn in het geheugen, zonder specifieke transformaties (in contrast met abstraheren). In de voorspellingsfase worden gelijke gebruikers of items gesorteerd op basis van ratings die zich in het geheugen bevinden. Gebaseerd op de ratings van deze gelijkaardige gebruikers of items kan een aanbeveling voor de actieve gebruiker gegenereerd worden. Voorbeelden van geheugengebaseerde Collaborative Filtering bevatten user-based methoden [6] en item-based methoden [25]. Het voordeel van geheugengebaseerde methoden ten opzichte van modelgebaseerde alternatieven is dat minder parameters afgestemd moeten worden.

In de modelgebaseerde benadering worden trainingsdata gebruikt om een model te genereren dat ratings kan voorspellen voor items die nog niet beoordeeld werden door de actieve gebruiker. Aanbevelingen worden dan gegenereerd door beroep te doen op het model. Voorbeelden hierbij worden gegeven in [6]: er wordt hier gebruik gemaakt

van clustermodellen en Bayesiaanse netwerken. De nood om een significant aantal parameters af te stemmen, zorgt ervoor dat dit niet praktisch is in gebruik [59].

Geheugengebaseerde aanbevelingssystemen zijn eenvoudiger, werken goed in de praktijk en nieuwe data kunnen snel en gemakkelijk toegevoegd worden. Deze manier van werken kan echter computationeel duur uitvallen, zowel in tijd als in geheugengebruik. Modelgebaseerde algoritmen vereisen heel wat minder geheugen, aanbevelingen worden snel berekend eens het model gegenereerd is, maar de tijd om de data in het model te compileren kan enorm zijn en het toevoegen van één nieuw datapunt kan een volledige hercompilatie vereisen [43].

Er bestaat nog een andere classificatie voor CF-systemen: ratinggebaseerde en loggebaseerde CF [60]. De informatie komt hier respectievelijk uit ratings of uit logbestanden (iemand heeft bijvoorbeeld iets gekocht). Het doel van ratinggebaseerde CF is om een rating te voorspellen voor de gebruikers, terwijl dat bij loggebaseerde CF is om items te ordenen in dalende relevantie voor de gebruiker. Bij de ratinggebaseerde variant kan er opnieuw gebruik gemaakt worden van zowel geheugengebaseerde als modelgebaseerde methoden. Er is nog heel weinig onderzoek verricht naar loggebaseerde CF, behalve het reeds gekende item-based top-N CF [10, 25] en item-based CF bij Amazon.com [31].

Het aantal artikels dat gevonden kan worden over CF is enorm. Rich [47] wordt aanzien als een eerste referentie. In 1992 kwamen D. Goldberg *et. al.* met de term 'Collaborative Filtering' in de context van een systeem om e-mails te filteren, gebruik makend van binaire categorievlaggen [14]. Shardanand en Maes [57, 54] ontwerpten een CF-systeem voor muziek, Ringo genaamd, en experimenteerden met een aantal metingen van de afstanden tussen gebruikers, waaronder Pearsoncorrelatie en vectorsimilariteit. Ze vergeleken vier recommandatiealgoritmen gebaseerd op de Mean Absolute Error van de voorspellingen. Al hun neighbourhoodgebaseerde algoritmen vereisen tijd die lineair is met het aantal gebruikers.

GroupLens is een baanbrekende inspanning in CF [27, 18]. Het GroupLensteam implementeerde aanvankelijk een neighbourhoodgebaseerd CF-systeem om Usenetartikels te beoordelen. De gebruikerscommunities beoordeelden de verschillende artikels op een schaal van één tot en met vijf en vervolgens berekent het systeem de afstand tot andere gebruikers door middel van Pearsoncorrelatie. Na het GroupLensteam ontstonden nog vele andere onderzoeksgroepen die Collaborative Filtering probeerden te verbeteren.

Er bestaan reeds menig aantal Collaborative Filteringsystemen. Deze dekken dan ook alles waarin potentieel interesse kan zijn bij gebruikers: van webpagina's tot nieuwsfeiten over muziek tot boeken en van restaurants tot wijnen. Enkele van deze systemen veranderden in commerciële bedrijven. Verder worden CF-technieken populairder als deel uitmakend van online winkels.

4.1 Notatie

Aangezien een aantal verschillende algoritmen besproken worden in deze thesis, werd de omgeving waarin Collaborative Filtering werkt geformaliseerd. De omgeving is samengesteld uit:

- Een set P van m uniek identificeerbare personen (gebruikers of klanten):

  $$P = \{p_1, p_2, p_3, \ldots, p_m\}$$

  Eigenlijk is de term personen niet goed gekozen, aangezien dit in Collaborative Filteringsystemen evengoed intelligente webservers kunnen zijn (die willen bronnen delen of ruilen), nodes van een peer-to-peer netwerk, software agents, ... Toch zal in deze thesis telkens sprake zijn van personen of individuen, aangezien onze toepassing enkel gebruik maakt van personen. De belangrijkste eis is dat de individuen uniek identificeerbaar zijn, bijvoorbeeld aan de hand van een gebruikersnaam.

- Een set I van n uniek identificeerbare items:

  $$I = \{i_1, i_2, i_3, \ldots, i_n\}$$

  Als unieke identifiers kan er gebruik gemaakt worden van globaal overeengekomen gegevens, zoals het ISBN-nummer bij boeken of van de omschrijving van het item.

- m sets van ratings of beoordelingen. Elk individu kan zijn mening uitdrukken voor elk item aan de hand van ratings. Deze ratings kunnen omgezet worden via een ratingfunctie van het domein I (bijvoorbeeld een rating tussen 0 en 5) naar het codomein [0,1] waarbij 0 totale afkeer betekent en 1 maximale appreciatie. Een ongekende waarde (een waarde waarvoor de functie niet gedefinieerd werd) betekent dat de gebruiker het item niet heeft beoordeeld en heeft als symbool $\perp$.

  $$r_{p_i}\ I \rightarrow [0, 1] \cup \{\perp\}$$

  Bijvoorbeeld, $r_{p_1}(i_1) = 0.1$ betekent dat persoon p_1 het item i_1 beoordeeld heeft en de functie aan die rating waarde 0.1 gaf, een lage ratingwaarde die dus een afkeer betekent voor dit item. Er kan ook onmiddellijk gebruik gemaakt worden van binaire waarden: iemand heeft iets gekocht of niet, iets bezocht of niet, ...

- De actieve gebruiker is de gebruiker waarvoor op dit moment aanbevelingen gegenereerd worden.

Het doel van Collaborative Filtering is het top-N recommandatieprobleem op te lossen. In [10] wordt volgende definitie gegeven hiervoor:

Gegeven een gebruiker-productmatrix $\mathbf{R}$ en een set van producten I die aangekocht werden door een gebruiker, identificeer dan een geordende set van producten X zodat $\|X\| \leq N$ en $X \cap I = \emptyset$.

4.2 User-based

User-based Collaborative Filtering is momenteel de meest succesvolle technologie om aanbevelingssystemen te bouwen. De algoritmen vertrouwen op het feit dat elke persoon behoort tot een grotere groep van individuen die zich gelijkaardig gedragen. Bijgevolg kunnen items die frequent gekocht worden door verschillende groepsleden, gebruikt worden als basis voor de aangeraden items.

Nadat de representatie van de gegevens vastgesteld werd, kunnen er aanbevelingen gedaan worden. Dit gebeurt in drie stappen [18, 51, 39, 34]:

1. **Gelijkaardigheid berekenen.** Ten eerste worden de gelijkaardigheden berekend tussen de actieve gebruiker en de rest van de gebruikers. Dit wordt bekomen door de ratings van de huidige gebruiker te vergelijken met die van andere gebruikers: een gelijkheidswaarde wordt berekend voor elke andere gebruiker, waarbij 1 volledig gelijk betekent en -1 volledig verschillend. De gelijkheidswaarde is enkel te berekenen indien er items beoordeeld werden door beide gebruikers. Indien dit niet het geval is, kunnen de gebruikers niet vergeleken worden.

2. **Buren bepalen.** Daarna wordt er een neighbourhood gevormd, dit zijn de dichtste buren van de actieve gebruiker.

3. **Aanbevelingen genereren.** Ten slotte kunnen er items aanbevolen worden met de hoogst voorspelde rating. Gebaseerd op de ratings van de buren worden de ratings voorspeld voor de ongekende items voor de actieve gebruiker.

4.2.1 Representatie

Met representatie wordt het voorstellen van de bruikbare gegevens voor het systeem bedoeld. Meestal is dit de voorstelling van m gebruikers die n producten kunnen aankopen of gebruiken. Dit leidt dus tot een $m \times n$ user-item matrix, ook interactiematrix genoemd, zoals Tabel 4.1. In dit voorbeeld werd uitgegaan van impliciete ratings: een gebruiker koopt/gebruikt het product of niet, zodat:

$$r_{i,j} = \begin{cases} 1 & \text{als gebruiker } i \text{ product } j \text{ heeft gekocht, beoordeeld, } \ldots \\ 0 & \text{anders} \end{cases}$$

Het kan ook zo zijn dat er uitgegaan wordt van expliciete ratings, *punten* die gegeven worden aan het item (Tabel 4.2). Die ratings liggen hier in het bereik $[0, 5]$ met 0 als extreem slecht en 5 als absolute aanrader. Wanneer $\perp$ ingevuld staat in deze matrix, betekent dit dat er geen gegevens beschikbaar zijn, m.a.w. die gebruiker heeft het item niet aangekocht of kent het niet. Het doel is dan $\perp$ te berekenen.

	item 1	item 2	item 3	item 4
Anthony	1	0	0	1
Karen	0	0	0	1
Tom	0	1	0	1

Tabel 4.1: Voorbeeld user-itemmatrix met binaire data

	item 1	item 2	item 3	item 4
Anthony	1	4	$\perp$	2
Karen	$\perp$	$\perp$	2	2
Tom	$\perp$	5	$\perp$	1

Tabel 4.2: Voorbeeld user-itemmatrix met expliciete ratings

Er zijn een aantal problemen met de representaties, waardoor deze vorm nog verbeterd kan worden. Zo kan bijvoorbeeld de grootte van de matrix verkleind worden door een aantal items als 'typisch' te aanzien voor een bepaalde groep items. Een te grote matrix leidt namelijk tot problemen met het geheugen, terwijl Collaborative Filtering toch heel schaalbaar moet blijven. Ook zijn er problemen met het tekort aan gegevens, de *sparsity* of ijlheid. Wanneer er bijvoorbeeld een miljoen boeken aangeraden kunnen worden, dan is het praktisch onmogelijk dat een gebruiker er één procent, of 10000 boeken, van gelezen heeft.

4.2.2 Gelijkaardigheid berekenen

De belangrijkste stap in user-based Collaborative Filtering aanbevelingssystemen is het berekenen van de similariteit tussen de gebruikers en de actieve gebruiker. Het doel hiervan is om de neighbourhood te vormen van de actieve gebruiker. Dit zijn alle personen die een geschiedenis hebben waarin ze het eens zijn met de actieve gebruiker. Deze neighbourhood is een geordende lijst van l klanten $N = \{N_1, N_2, \ldots, N_l\}$ voor elke gebruiker u zodat $u \notin N$ en s_{u,N_1} maximaal is, s_{u,N_2} het op één na grootste en zo verder. s_{u,N_1} is hier de gelijkaardigheidswaarde tussen u en N_1.

De nabijheid van twee verschillende gebruikers wordt meestal berekend aan de hand van vectorsimilariteit of via de Pearsoncorrelatie. Er worden nog andere methoden gebruikt, maar deze zijn meestal variaties op de basismethoden.

Vectorsimilariteit

In dit geval worden de twee gebruikers a en b aanzien als twee vectoren in een n-dimensionale productruimte (of de k-dimensionale ruimte indien er sprake is van een gereduceerde representatie). De componenten van de vector zijn positief voor aangekochte of positief beoordeelde items. Deze kunnen ook negatief zijn indien een gebruiker ze negatief beoordeeld heeft. De dichtheid tussen de twee vectoren wordt gemeten door de cosinus te berekenen van de hoek tussen de twee, gegeven door [6]:

$$s_{\vec{a},\vec{b}} = \frac{\vec{a}.\vec{b}}{\|\vec{a}\|_2 \times \|\vec{b}\|_2} \tag{4.1}$$

'.' betekent hier het scalair product van de twee vectoren. Formule 4.1 kan herschreven worden als volgt:

$$s_{a,b} = \sum_i \frac{r_{a,i}}{\sqrt{\sum_{k \in I_a} r_{a,k}^2}} \frac{r_{b,i}}{\sqrt{\sum_{k \in I_b} r_{b,k}^2}} \tag{4.2}$$

Hierbij loopt i over alle items die de twee gebruikers a en b gemeenschappelijk beoordeeld hebben. $r_{a,i}$ is de rating gegeven aan item i door gebruiker a; I_a een verzameling van alle items waaraan gebruiker a een rating gegeven heeft. De vierkantswortels in de noemer dienen om de ratings te normaliseren zodat gebruikers die verschillende items beoordelen niet a priori meer gelijk zijn aan andere gebruikers.

Een voorbeeld zal alles duidelijker maken. Hierbij wordt gestart van de interactiematrix $\mathbf{R}$ met 3 gebruikers en 4 producten. Deze matrix wordt eerst genormaliseerd per gebruiker indien er met binaire ratings wordt gewerkt, m.a.w. als een gebruiker drie producten heeft beoordeeld, wordt elke rating die hij/zij gegeven heeft gedeeld door drie. Hierdoor wordt de genormaliseerde matrix $\mathbf{R}'$ bekomen waarbij een gebruiker die meer producten beoordeelde, niet per se meer gelijk zal zijn aan een andere gebruiker. Deze stap is in principe niet nodig, maar leidt wel tot een beter CF-systeem.

$$\mathbf{R} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \quad \mathbf{R}' = \begin{bmatrix} 0 & 0.5 & 0 & 0.5 \\ 0 & 0.33 & 0.33 & 0.33 \\ 0.5 & 0 & 0.5 & 0 \end{bmatrix} \tag{4.3}$$

Vervolgens worden de verschillende similariteiten berekend. Zo zal er geen similariteit zijn tussen gebruikers 1 en 3 aangezien zij geen enkel product gemeenschappelijk beoordeeld hebben. De similariteit berekenen tussen gebruikers 1 en 2 is echter wel mogelijk aan de hand van Formule 4.2:

$$s_{1,2} = \frac{0.5}{\sqrt{0.5^2 + 0.5^2}} \frac{0.33}{\sqrt{0.33^2 + 0.33^2 + 0.33^2}} + \frac{0.5}{\sqrt{0.5^2 + 0.5^2}} \frac{0.33}{\sqrt{0.33^2 + 0.33^2 + 0.33^2}}$$
$$= 0.82$$

Hierna moet enkel nog de similariteit berekend worden tussen gebruikers 2 en 3, aangezien al de rest ofwel nul is ofwel één. Zo bekomt men uiteindelijk de similariteitsmatrix **M**.

$$\mathbf{M} = \begin{bmatrix} 1 & 0.82 & 0 \\ 0.82 & 1 & 0.41 \\ 0 & 0.41 & 1 \end{bmatrix}$$

Pearsoncorrelatie

De dichtheid tussen gebruiker a en gebruiker b kan ook bepaald worden aan de hand van de Pearsoncorrelatie, die wordt gegeven door [18, 6]:

$$s_{a,b} = \frac{\sum_i (r_{a,i} - \bar{r}_a)(r_{b,i} - \bar{r}_b)}{\sqrt{\sum_i (r_{a,i} - \bar{r}_a)^2 \sum_i (r_{b,i} - \bar{r}_b)^2}} \tag{4.4}$$

Hierbij is $r_{a,i}$ de rating gegeven aan item i door gebruiker a; loopt i over alle items die beide gebruikers a en b beoordeeld hebben en is $\bar{r}_a$ de gemiddelde rating die gegeven wordt door gebruiker a. I_a is verzameling van alle items waarvoor a een rating gaf:

$$\bar{r}_a = \frac{1}{\mid I_a \mid} \sum_{j \in I_a} r_{a,j}$$

Belangrijk hierbij is op te merken dat deze correlatiecoëfficient enkel berekend kan worden op overlappende items. Indien twee gebruikers één item gemeenschappelijk beoordeeld hebben, dan is de coëfficient niet betekenisvol, aangezien er dan telkens door 0 gedeeld wordt. Daarom is het voor een gebruiker enkel mogelijk om de correlatiecoëfficient te berekenen indien de gebruikers minstens twee items gemeenschappelijk beoordeeld hebben. Dit is meestal slechts een klein percentage, zoals beschreven in [35]. Doordat de formule ook het gemiddelde gebruikt van de ratings, is het onmogelijk om gebruik te maken van binaire waarden. Dit zou namelijk telkens leiden tot een onbepaaldheid (0/0).

Hieronder zal getoond worden hoe de similariteit berekend wordt tussen twee gebruikers met minstens twee gemeenschappelijk beoordeelde producten. Er wordt gestart met

de interactiematrix **R** die de ratings bevat van de verschillende gebruikers. Vervolgens wordt er per gebruiker het gemiddelde berekend wat leidt tot **G**.

$$\mathbf{R} = \begin{bmatrix} \perp & 5 & \perp & 2 \\ \perp & 4 & 3 & 5 \\ 4 & \perp & 3 & \perp \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} 3.5 \\ 4 \\ 3.5 \end{bmatrix} \tag{4.5}$$

De nodige gegevens zijn nu allemaal beschikbaar, dus moet enkel de similariteit berekend worden. Als voorbeeld wordt de similariteit genomen tussen gebruiker 1 en 2. Deze gebruikers hebben twee gemeenschappelijke producten beoordeeld.

$$s_{1,2} = \frac{(5-3.5)(4-4)+(2-3.5)(5-4)}{\sqrt{((5-3.5)^2+(2-3.5)^2)((4-4)^2+(5-4)^2)}} = -0.71$$

Er worden enkele veronderstellingen gemaakt over de data. De relaties tussen gebruikers moet lineair zijn. Ook zijn enkele voorwaarden verbonden aan de fouten: ze moeten onafhankelijk zijn en ze moeten een waarschijnlijkheidsverdeling met een gemiddelde van 0 hebben [18]. Deze veronderstellingen zijn vaak maar gedeeltelijk voldaan in Collaborative Filteringdatasets. In dat geval wordt Pearsoncorrelatie minder betrouwbaar en kan er eventueel overgeschakeld worden op Spearmancorrelatie.

Spearmancorrelatie

Deze correlatie is gelijkaardig aan de Pearsoncorrelatie met het verschil dat het geen veronderstellingen maakt over de data. De correlatie wordt niet tussen de beoordelingen zelf maar tussen de posities van beoordelingen van een gebruiker a berekend. Het item i met de laagste beoordeling krijgt positie 1, $positie_{a,i} = 1$. Daarna volgt een item met gelijke of hogere beoordeling die positie 2 krijgt, ... De Spearmancorrelatie wordt als volgt berekend:

$$s_{a,b} = \frac{\sum_i^m (positie_{a,i} - \overline{positie_a})(positie_{b,i} - \overline{positie_b})}{\sigma_a \sigma_b} \tag{4.6}$$

waarin σ de standaardafwijking is. Indien er gewerkt wordt met een kleine beoordelingsschaal, bijvoorbeeld discrete waarden van 0 tot en met 5, dan krijgt één bepaalde beoordeling veel verschillende posities. Dit zorgt voor een minder juist resultaat. Spearman is dan ook niet toe te passen voor loggebaseerde Collaborative Filtering. Toch kiest Spearman in vele gevallen nagenoeg dezelfde buren als Pearson, waardoor de werking even goed is als Pearsoncorrelatie [18].

Uitbreidingen

Inverse user frequency Een uitbreiding van vectorsimilariteit en Pearsoncorrelatie is het gebruik van inverse user frequency. Uit onderzoek blijkt dat de door iedereen geliefde items niet zo bruikbaar zijn als de minder gekende items om similariteiten te berekenen. Om hiermee rekening te houden wordt f_i gedefinieerd als $\log \frac{n}{n_i}$ waarbij n_i het aantal gebruikers is die item i beoordeeld hebben en n het totale aantal gebruikers. Merk op, als iedereen het item beoordeeld heeft, dan is f_i gelijk aan nul.

Om inverse user frequency toe te passen bij het vectorsimilariteitsalgoritme, wordt een getransformeerde rating gebruikt bij formule 4.2. De getransformeerde rating is simpelweg de originele rating vermenigvuldigd met de f_i factor.

$$s_{a,b} = \sum_i \frac{f_i r_{a,i}}{\sqrt{\sum_{k \in I_a} f_k r_{a,k}^2}} \frac{f_i r_{b,i}}{\sqrt{\sum_{k \in I_b} f_k r_{b,k}^2}}$$

In het geval van Pearsoncorrelatie, wordt formule 4.4 aangepast zodat f_i behandeld wordt als frequentie en een item met een hogere f_i meer gewicht krijgt in de correlatieberekening. Het relevante correlatiegewicht met de inverse frequentie is:

$$s_{a,b} = \frac{\sum_i f_i \sum_i f_i \; r_{a,i} \; r_{b,i} - (\sum_i f_i \; r_{a,i})(\sum_i f_i \; r_{b,i})}{\sqrt{(UV)}}$$

waarbij

$$U = \sum_i f_i \; (\sum_i f_i \; r_{a,i}^2 - (\sum_i f_i \; r_{a,i})^2)$$
$$V = \sum_i f_i \; (\sum_i f_i \; r_{b,i}^2 - (\sum_i f_i \; r_{b,i})^2)$$

Het onderzoek uitgevoerd in [6] heeft aangetoond dat het gebruik van inverse user frequency een statistisch significante verbetering met zich meebracht in vergelijking met de gewone correlatieberekening en vectorsimilariteit. Hieronder volgt een voorbeeld om duidelijk te maken hoe alles in zijn werk gaat. Het voorbeeld werkt met vectorsimilariteit.

Gegeven is de interactiematrix $\mathbf{R}$ die reeds eerder vermeld werd (Figuur 4.3). Deze matrix wordt, zoals bij gewone vectorsimilariteit met binaire ratings, eerst genormaliseerd per gebruiker, waardoor matrix $\mathbf{R}'$ bekomen wordt (Figuur 4.3). Per item wordt vervolgens de frequentie berekend. Zo is dit voor het eerste item:

$$f_1 = \log \frac{3}{1} = 0.47712$$

om te komen tot:

$$\begin{bmatrix} 0.47712 & 0.17609 & 0.17609 & 0.17609 \end{bmatrix}$$

Hierna wordt in matrix $\mathbf{R}'$ rekening gehouden met de frequenties, waardoor matrix $\mathbf{R}''$ ontstaat. Ten slotte kan de gebruikerssimilariteitsmatrix $\mathbf{M}$ berekend worden aan de hand van $\mathbf{R}''$ en via formule 4.2.

$$\mathbf{R}'' = \begin{bmatrix} 0 & 0.08805 & 0 & 0.08805 \\ 0 & 0.05869 & 0.05870 & 0.05870 \\ 0.23856 & 0 & 0.08805 & 0 \end{bmatrix} \quad \mathbf{M} = \begin{bmatrix} 1 & 0.82 & 0 \\ 0.82 & 1 & 0.2 \\ 0 & 0.2 & 1 \end{bmatrix} \tag{4.7}$$

Constrained Pearson correlation Een variant op het Pearsonalgoritme is het gebruik maken van het positieve en het negatieve van ratings. Wanneer de schaal van ratings bijvoorbeeld ligt tussen 0 en 5, kan er aangenomen worden dat de waarden boven 4 positief aanvaard worden en die eronder negatief [54, 62]. Het Pearsonschema kan dan aangepast worden zodat enkel wanneer beide personen een item positief beoordeeld hebben of negatief, de correlatiecoëfficiënt stijgt tussen die twee personen. De aangepaste Pearsonvergelijking wordt hieronder gegeven waarbij 4 kan veranderd worden in een waarde naar keuze.

$$s_{a,b} = \frac{\sum_i (r_{a,i} - 4)(r_{b,i} - 4)}{\sqrt{\sum_i (r_{a,i} - 4)^2 \sum_i (r_{b,i} - 4)^2}}$$

In artikel [54] wordt beweerd dat de gelijkaardigheid berekenen aan de hand van bovenstaande formule performanter is dan met de standaard Pearsonberekening. Het spreekt voor zich dat dit niet mogelijk is voor binaire waarden.

Default voting Default voting is een andere uitbreiding op het Pearsoncorrelatiealgoritme [6]. Het is ontstaan uit de observatie dat wanneer er relatief weinig beoordelingen zijn voor zowel de actieve gebruiker of de vergeleken gebruiker, het correlatiealgoritme niet goed zal functioneren aangezien dit enkel de ratings gebruikt die beide gebruikers gemeenschappelijk beoordeeld hebben ($I_a \cap I_b$). Wanneer een standaardwaarde aangenomen wordt als beoordeling voor items waarvoor nog geen expliciete beoordelingen aanwezig zijn, dan kan er gekeken worden naar de unie van items die beide gebruikers beoordeeld hebben, niet per se gemeenschappelijk ($I_a \cup I_b$). Hierbij wordt de standaardwaarde telkens ingevoegd in de formule voor de onbeoordeelde items.

Ook kan dezelfde standaardwaarde d aangenomen worden voor een aantal bijkomende items k die geen enkele gebruiker heeft beoordeeld. Dit zorgt voor het effect dat er een aantal extra items zijn die geen enkele gebruiker beoordeelde, maar waarvoor beide gebruikers toch zouden overeenkomen. In de meeste gevallen zal de waarde voor d een neutrale, ietwat negatieve voorkeur uitdrukken. De vergelijking wordt gegeven als:

$$s_{a,b} = \frac{(n+k)\sum_i(r_{a,i}r_{b,i} + kd^2) - (\sum_i r_{a,i} + kd)(\sum_i r_{b,i} + kd)}{\sqrt{((n+k)\sum_i(r_{a,i}^2 + kd^2) - (\sum_i r_{a,i} + kd)^2)((n+k)\sum_i(r_{b,i}^2 + kd^2) - (\sum_i r_{b,i} + kd)^2)}}$$

Hierbij stelt i alle items voor uit de unie van de items die gebruiker a of b beoordeeld heeft ($I_a \cup I_b$) en is $n =\mid I_a \cup I_b \mid$. Belangrijk hierbij op te merken is dat enkel gewichten berekend worden voor gebruikers die minstens één item gemeenschappelijk beoordeeld hebben met de actieve gebruiker.

4.2.3 Buren bepalen

Het selecteren van de buren probeert twee doelen te bevredigen:

1. Buren zouden sterk gecorreleerd moeten zijn met de actieve gebruiker.
2. Buren zouden in staat moeten zijn om het gedrag te voorspellen voor vele items die nog niet beoordeeld werden door de actieve gebruiker.

Aan de hand van de methoden hiervoor beschreven, wordt voor m klanten de $m \times m$ gelijkaardigheidsmatrix opgesteld. De volgende taak is om de eigenlijke neighbourhood of omgeving te vormen. Ook hier zijn er verschillende mogelijkheden, maar er is één veelgebruikte en gemakkelijke methode: de centrumgebaseerde methode. Hierbij wordt een neighbourhood gevormd met grootte k voor een gebruiker c door de k dichtste andere klanten te selecteren. Zo zal bij **M** (Figuur 4.7) voor gebruiker 1 de dichtste buur gebruiker 2 zijn met een gelijkaardigheidsfactor van 0.82.

Een andere manier is het geaggregeerdeneighbourhoodschema, waarbij een neighbourhood gevormd wordt van grootte k voor klant c door eerst de dichtste buur te zoeken van c. De andere k-1 buren worden als volgt geselecteerd. Laat er op een bepaald moment j buren zijn in de neighbourhood N, waarbij $j < k$. Het algoritme berekent dan het centrum van de neighbourhood. Het midden van N wordt gedefinieerd als $\vec{C}$ en berekend als $\vec{C} = \frac{1}{j}\sum_{\vec{V} \in N} \vec{V}$. Een klant w, met $w \notin N$, wordt enkel geselecteerd als de j + 1-ste buur als w de dichtste is bij het midden $\vec{C}$. Dan wordt het centrum herberekend voor j+1 buren en het proces gaat verder totdat $\mid N \mid$=k. Dus dit algoritme laat toe dat de dichtste buren de vorming van de neighbourhood beïnvloeden, wat

voordelig kan zijn voor heel erg schaarse datasets. Toch is gebleken uit experimenten dat dit niet het geval is [51]. Aangezien deze methode veel ingewikkelder is dan de vorige, wordt er voorkeur gegeven aan de vorige methode.

4.2.4 Aanbevelingen genereren

De laatste stap bij CF-gebaseerde aanbevelingssystemen is om de top-N van aanbevelingen af te leiden uit de neighbourhood van de gebruikers. Opnieuw zijn er verschillende manieren om dit te bereiken.

Meest frequente aangekocht

Meest frequente-itemaanbeveling kijkt in de neighbourhood N en scant voor elke buur door de aankoopdata of beoordeelde items en voert hierbij een frequentietelling uit van de producten. Nadat alle buren bekeken zijn, sorteert het systeem de producten volgens frequentietelling en geeft vervolgens enkel de N frequentste producten terug als aanbeveling. Hierbij werden natuurlijk wel alle producten weggelaten die de gebruiker reeds bezit of kent.

Gewogen gemiddelde van de afwijking

Een andere veelgebruikte methode is het berekenen van voorspellingen aan de hand van het gewogen gemiddelde van de afwijkingen van het gemiddelde van de buren:

$$p_{a,i} = \bar{r}_a + \frac{\sum_u (r_{u,i} - \bar{r}_u) \times s_{a,u}}{\sum_u s_{a,u}}$$

Hierbij is $p_{a,i}$ de voorspelling voor de actieve gebruiker a voor item i; $s_{a,u}$ de gelijkaardigheid tussen de gebruikers a en u en loopt u over alle gebruikers in de neighbourhood.

Potentiëlescorematrix

In artikel [6] wordt gebruik gemaakt van een potentiëlescorematrix, die later ook zal gebruikt worden bij item-based CF. Hierbij wordt de bekomen similariteitsmatrix $\mathbf{M}$ vermenigvuldigd met de originele interactiematrix $\mathbf{R}$.

$$\mathbf{M}.\mathbf{R} = \begin{bmatrix} 1 & 0.82 & 0 \\ 0.82 & 1 & 0.2 \\ 0 & 0.2 & 1 \end{bmatrix} . \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1.82 & 0.82 & 1.82 \\ 0.2 & 1.82 & 1.2 & 1.82 \\ 1 & 0.2 & 1.2 & 02 \end{bmatrix}$$

4.2.5 Efficiëntie

User-based CF gebruiken om aan te bevelen is niet zo efficiënt. Het is $O(mn)$ in het slechtste geval, aangezien het m gebruikers bekijkt en tot n producten voor elke gebruiker. Maar, omdat de gemiddelde gebruiker slechts weinig producten beoordeelt, ligt de performantie van het algoritme dichter bij $O(m + n)$. Elke gebruiker scannen is ongeveer $O(m)$, niet $O(mn)$, aangezien bijna alle klanten een kleine voorkeurvector hebben in vergelijking met de grootte van de catalogus. Maar er zijn een klein aantal gebruikers die een groot percentage van de catalogus beoordeeld of aangekocht hebben, wat $O(n)$ vraagt. Dus is de finale performantie van deze algoritmen $O(m + n)$. Voor grote datasets, met 10 miljoen gebruikers en 1 miljoen producten in de catalogus, zal dit algoritme grote performantie- en schalingsproblemen hebben [31].

4.2.6 Problemen

Ondanks de populariteit van user-based CF-aanbevelingssystemen, zijn er toch een aantal limieten gerelateerd aan schaalbaarheid en real-time werking [25, 34, 36].

Schaalbaarheid. De computationele complexiteit van deze methoden groeit lineair met het aantal klanten. Op het moment dat er aanbevelingen online opgevraagd worden, moet er gezocht worden naar alle gebruikersprofielen om de beste buren te vinden. Dit probleem betekent dat deze aanbevelingssystemen niet kunnen schalen naar grote omgevingen met miljoenen gebruikers en producten, wat het geval is in veel commerciële applicaties. Deze stap is ook heel traag, op die manier dat het soms seconden tot minuten kan duren om buren te vinden voor de actieve gebruiker. Hierdoor is het dus niet haalbaar om de buren te bepalen wanneer er een aanvraag is door een gebruiker, waardoor dit best periodiek offline berekend wordt. Hieruit volgt wel dat aanbevelingen niet altijd up to date zijn en dat nieuwe beoordelingen van de gebruiker niet direct effect hebben.

Sparsity of ijlheid. Gebruikersgelijkaardigheid is enkel berekenbaar ten opzichte van enkele andere gebruikers. Om het mogelijk te maken om aanbevelingen te creëren van goede kwaliteit, moet het aanbevelingssysteem de actieve gebruiker kunnen vergelijken met alle andere gebruikers met het doel die buren te selecteren met de meer relevante ratings voor items. Deze stap is verplicht en de accuraatheid ervan beïnvloedt de accuraatheid van het hele systeem: falen in het vinden van goede buren leidt tot aanbevelingen met een slechte kwaliteit. Doordat de ratingsmatrix $\mathbf{R}$ meestal heel ijl is, want gebruikers hebben de neiging om slechts een miniem aantal items te beoordelen, komt het veel voor dat twee gebruikers niet het minimum aantal items gemeenschappelijk

hebben beoordeeld nodig voor metrieken om similariteit te berekenen. Daarom wordt het systeem geforceerd om die buren te kiezen in de kleine portie vergelijkbare gebruikers en zal daardoor waarschijnlijk andere niet-vergelijkbare maar relevante gebruikers missen. Dit komt normaal niet voor bij gebruikers met honderden ratings, maar wel voor gebruikers met slechts enkele ratings. Hoe dan ook kan het betwist worden dat het belangrijker is voor een aanbevelingssysteem om goede aanbevelingen te doen aan een gebruiker met weinig ratings om deze uit te nodigen om meer ratings in te brengen en het systeem te blijven gebruiken dan aan een gebruiker met veel ratings die waarschijnlijk het systeem al regelmatig gebruikt.

Ook is het zo dat, hoewel geweten is dat een paar gebruikers geen items gemeenschappelijk beoordeeld hebben, deze toch nog altijd dezelfde smaak kunnen hebben. Veronderstel bijvoorbeeld dat gebruikers a en b een gelijkaardige smaak hebben, evenals gebruikers b en c. Deze relaties geeft ook informatie over de similariteit tussen gebruikers a en c. Wanneer a en c geen items gemeenschappelijk beoordeelden, dan zou een correlatiegebaseerde similariteitsberekening geen relatie ontdekken tussen de twee gebruikers. Potentieel nuttige informatie gaat zo verloren als deze transitieve similariteitsrelatie niet ontdekt kan worden [5]. Er zijn reeds verscheidene oplossingen onderzocht om dit probleem op te lossen. Enkele hiervan zullen later in deze tekst besproken worden.

Belangrijk hierbij op te merken is dat deze sparsity of ijlheid de motivatie is om te filteren: de meeste mensen willen niet alle beschikbare informatie verwerken. Anderzijds zorgt de ijlheid voor een computationele uitdaging omdat het moeilijker wordt buren te vinden en items aan te raden aangezien weinig personen deze beoordeeld hebben.

Cold-start problem. Wanneer nieuwe gebruikers zich aanmelden bij het systeem, kunnen er nog geen items aanbevolen worden (ook het early-rater problem genoemd). Hetzelfde geldt voor een nieuw product dat toegevoegd wordt. Dit probleem staat bekend als het 'cold-start problem' [23]. CF kan geen betekenisvolle aanbevelingen doen voor een nieuwe gebruiker aangezien er te weinig of zelfs geen ratings zijn. Op dezelfde manier is het zo dat als een nieuw product toegevoegd wordt, het onmogelijk is dat dit wordt aangeraden aangezien weinig gebruikers dit al beoordeeld hebben [53]. Het 'cold-start problem' kan dus eigenlijk aanzien worden als een speciaal geval van het ijlheidsprobleem. CF-systemen steunen vooral op het altruïsme van een aantal gebruikers die veel producten willen beoordelen zonder veel aanbevelingen te krijgen. Economisten vermoeden dat zelfs indien beoordelen geen inspanning vereist van de gebruiker, gebruikers zouden wachten om items in aanmerking te nemen totdat de buren ze aanbieden via aanbevelingen [4]. Er zijn reeds een aantal oplossingen gezocht voor dit probleem, maar niet elke oplossing is even gemakkelijk toepasbaar op elke dataset [61, 46, 52]. De meest gebruikte oplossing, is het presenteren van een startset

die iedereen moet beoordelen samen met enkele strategisch gekozen items.

Aanvallen. Een ander gekend probleem bij Collaborative Filteringsystemen zijn de gemakkelijke aanvallen van kwaadaardige gebruikers. Aanbevelingssystemen worden veel gebruikt bij e-commerce sites (bv. Amazon.com). In deze context is het heel aantrekkelijk om de aanbevelingen te beïnvloeden, zoals bijvoorbeeld iemand die forceert dat het boek dat hij/zij geschreven heeft altijd wordt aanbevolen. Hoe dan ook is het ondermijnen van CF-technieken heel gemakkelijk [41]. De eenvoudigste aanval is gekend als de 'copy-profile attack' waarbij de aanvaller de ratings van een doelgebruiker kan kopiëren en het systeem hierbij denkt dat de aanvaller de meest gelijkaardige gebruiker is als de doelgebruiker. Zo zal elk extra item dat de aanvaller beoordeelt met grote waarschijnlijkheid aanbevolen worden aan de doelgebruiker. Omdat de huidige aanbevelingssystemen vooral gecentraliseerde servers zijn, is het creëren van valse identiteiten een tijdsconsumerende activiteit en daardoor zijn deze aanvallen nog niet echt bestudeerd [34]. Zo is er een gekend voorval van een aanval op Amazon.com[1]. Bij het boek van de katholieke televangelist Pat Robertson stond zo onder 'Customers who shopped for this item also shopped for these items' een seksueel getinte handleiding. Een andere gekende manier van aanvallen is een *nuke attack*. Dit heeft als doel de populariteit van een bepaald item te laten dalen. Verschillende antwoorden zijn hier reeds op gevonden, zoals [38].

Synoniemen. In het echte leven kunnen verschillende productnamen verwijzen naar hetzelfde object. Aanbevelingssystemen die op correlaties gebaseerd zijn, kunnen deze verborgen associatie niet vinden en behandelen deze producten dan ook verschillend, zodat ze ook verschillend kunnen aangeraden worden [49].

4.3 Item-based

De grootste e-commercewebsites werken op een schaal die rechtstreeks invloed heeft op de implementatie van CF. In user-based CF-systemen blijkt het burenformatieproces en daarbij vooral de berekening van de gebruikerssimilariteiten, de grootste bottleneck die op zijn beurt het hele proces ongeschikt maakt voor het genereren van realtime recommandaties. Een manier om hiermee om te gaan is item-based werken. Het grote verschil met user-based CF is dat de berekeningen uitgevoerd worden in de productruimte. In een typisch e-commercescenario zijn de items meestal statisch vergeleken met het aantal gebruikers dat constant verandert [50].

[1] `http://www.news.com/2100-1023-976435.html`

De intuïtie achter deze benadering is dat een gebruiker geïnteresseerd zal zijn in producten die gelijkaardig zijn als die die hij/zij eerder kocht en die producten zal vermijden die hem/haar vroeger niet interesseerden. Deze technieken moeten de buren van gelijkaardige gebruikers niet zoeken wanneer een aanbeveling gevraagd wordt. Als resultaat hiervan zullen ze veel sneller aanbevelingen produceren.

De statische eigenschappen van items leidt tot het idee om itemsimilariteiten op voorhand te berekenen. Een mogelijke manier om dit te doen is om offline alle similariteiten te berekenen tussen de verschillende items om vervolgens snel online iets op te zoeken in de tabel om er de gelijkaardigheidswaarden uit te halen. Deze methode vereist wel $O(n^2)$ ruimte voor n producten, maar bespaart heel wat tijd.

Omdat er slechts een klein aantal gelijkaardige items nodig zijn om voorspellingen te doen, kan er een alternatief model opgesteld worden. Hierin wordt telkens slechts een klein aantal gelijkaardige items bijgehouden. Voor elk item j worden de k meest gelijkaardige items berekend, waarbij $k << n$ en deze worden opgeslagen samen met hun similariteitswaarden. k wordt ook de *modelgrootte* genoemd.

Item-based CF-algoritmen bestaan hoofdzakelijk uit twee componenten [50, 10] die berekend kunnen worden op verschillende tijdstippen:

1. **Gelijkaardigheid berekenen.** De eerste component bepaalt de gelijkheden tussen alle n items onderling. Deze kan offline berekend worden: dit is één keer per dag of per week.
2. **Aanbevelingen genereren.** De tweede component maakt gebruik van de vooraf berekende gegevens om voorspellingen te doen. Deze component wordt online berekend, met andere woorden, het moment waarop de gebruiker suggesties verwacht.

Hoe dit precies in zijn werk gaat wordt hieronder uitgelegd. Merk op, er zijn nog andere manieren om item-based te werken zonder de voorgaande twee componenten. Dit zal aangetoond worden via het popalgoritme.

4.3.1 Gelijkaardigheid berekenen

Een kritieke stap in een item-based algoritme is het berekenen van de correlatie tussen items. De gelijkaardigheid berekenen van twee items i en j, $s_{i,j}$, gebeurt door de gebruikers die beide items beoordeeld hebben te selecteren en daarna met die gegevens een graad van gelijkenis te bepalen. Er zijn verschillende manieren bestudeerd om dit te berekenen, hieronder volgen de meest gebruikte [50, 10]. Deze zijn meestal gelijkaardig aan die van de user-based systemen, met wat kleine aanpassingen.

Vectorsimilariteit

Net zoals bij het berekenen van de correlatie bij user-based aanbevelingssystemen, kan ook hier gebruik gemaakt worden van vectorsimilariteit. Hier worden twee items a en b aanzien als vectoren in een m-dimensionale gebruikersruimte en wordt de dichtheid bepaald via de cosinus van de hoek tussen die vectoren. Er zijn slechts enkele aanpassingen nodig aan de berekening bij user-based systemen. Aangezien deze miniem zijn, zal dit verduidelijkt worden aan de hand van een voorbeeld.

De kolommen van een eenvoudige interactiematrix $\mathbf{R}$ worden genormaliseerd wanneer er binaire ratings gebruikt worden. Het verschil met het normaliseren op rij is dat er hier op product genormaliseerd wordt in plaats van op gebruiker: een product dat meer gekocht wordt, zal hierdoor niet meer doorwegen dan een product dat weinig gekocht wordt. Op die manier wordt $\mathbf{R}'$ bekomen.

$$\mathbf{R} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \quad \mathbf{R}' = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 0 & 1/2 & 1/2 & 1/2 \\ 1 & 0 & 1/2 & 0 \end{bmatrix}$$

De gelijkheid tussen items wordt berekend met onderstaande formule waarbij $s_{i,j}$ de similariteitswaarde is en $\vec{i}$ de vectoriële voorstelling is van item i (bij user-based was de vector de voorstelling van een gebruiker).

$$s_{i,j} = \cos(\vec{i}, \vec{j}) = \frac{\vec{i}.\vec{j}}{\|\vec{i}\|_2 \times \|\vec{j}\|_2}$$

$$s_{1,2} = \frac{(1/2, 1/2, 0).(0, 1/2, 1/2)}{\sqrt{(1/2)^2 + (1/2)^2} \times \sqrt{(1/2)^2 + (1/2)^2}} = \frac{1/4}{1/2} = 0,5$$

Op die manier wordt de volgende productcorrelatiematrix $\mathbf{M}$ berekend. Deze matrix heeft logischerwijze een andere dimensie (4×4) dan die van de gebruikerscorrelatiematrix (3×3).

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 1/2 & 0 \\ 0 & 1 & 1/2 & 1 \\ 1/2 & 1/2 & 1 & 1/2 \\ 0 & 1 & 1/2 & 1 \end{bmatrix}$$

Pearsoncorrelatie

Er is opnieuw de mogelijkheid om de correlatie te berekenen via de Pearsoncorrelatie. Ook hier zijn er geen grote verschillen met het user-based aanbevelingssysteem. Enkel wanneer er gebruik gemaakt wordt van genormaliseerde interactiematrices is er een verschil: het normaliseren gebeurt zoals hiervoor geschreven op item. De algemene formule wordt gegeven hieronder. Hierbij loopt p over alle individuen die beide items a en b hebben beoordeeld en is $\bar{r}_a$ de gemiddelde beoordeling die gegeven wordt aan item a.

$$s_{a,b} = \frac{\sum_{p \in P}(r_{p,a} - \bar{r}_a)(r_{p,b} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{p,a} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{p,b} - \bar{r}_b)^2}}$$

Gebaseerd op voorwaardelijke kans

Een alternatieve manier om de correlatie te berekenen tussen elk itempaar is gebaseerd op voorwaardelijke kans [50]: de kans dat item j gekocht wordt als item i al gekocht werd. $P(j|i)$ wordt berekend door het aantal klanten dat items i en j heeft gekocht ($freq(i,j)$) te delen door het totaal aantal klanten dat i heeft aangekocht ($freq(i)$):

$$P(j|i) = \frac{freq(i,j)}{freq(i)}$$

Merk op dat $P(j|i) \neq P(i|j)$. Het gaat hier dus om een asymmetrische relatie. Een nadeel van een asymmetrische relatie is dat elk item een hoge correlatie gaat hebben met een veel aangekocht item[2]. Dit probleem kan opgevangen worden door $P(j|i)$ te delen door bepaalde factor $P(j)$. Uit onderzoek [10] blijkt dat dit herschalen de precisie van een aanbevelingssysteem ten goede komt. Hoeveel invloed het herschalen best heeft is afhankelijk van de dataset. De invloed van $P(j)$ wordt dan ook best geparametriseerd:

$$s_{i,j} = \frac{freq(i,j)}{freq(i) \times freq(j)^\alpha} \quad , \quad \alpha \in [0,1]$$

Is $\alpha = 0$ dan valt de vergelijking terug tot de eerste. Is $\alpha = 1$ dan is de vergelijking gelijk aan $P(j|i)/P(j)$.

De correlatievergelijking ziet nog geen verschil tussen klanten die een verschillend aantal items gekocht hebben. Om dit verschil te laten doorwegen en om een hoger gewicht

[2]Dit is het zogenaamde "Harry Potter probleem": bij bijna elk boek kan een Harry Potter boek aangeraden worden omdat ze ooit samen werden aangekocht. Meestal is die aanbeveling echter niet gewenst.

te geven aan klanten die weinig items gekocht hebben, kan de vergelijking uitgebreid worden:

$$s_{i,j} = \frac{\sum_{\forall q: r_{q,i}>0} r_{q,j}}{freq(i) \times freq(j)^{\alpha}}$$

Hiervoor moeten de rijen in de user-itemmatrix genormaliseerd worden indien er sprake is van binaire ratings. Elke rij moet de eenheidslengte krijgen zodat elke gebruiker evenveel meetelt ongeacht het aantal gegeven beoordelingen. Hieronder wordt een klein voorbeeld uitgewerkt. Er wordt gebruik gemaakt van de niet-nulelementen in matrix $\mathbf{R}$ en niet meer van de frequentie van een item.

De rijen van een eenvoudige interactiematrix $\mathbf{R}$ worden genormaliseerd. Op die manier wordt $\mathbf{R}'$ bekomen.

$$\mathbf{R} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \quad \mathbf{R}' = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 0 & 1/3 & 1/3 & 1/3 \\ 1/2 & 0 & 1/2 & 0 \end{bmatrix}$$

Daarna wordt de correlatie tussen elk item berekend, α is in dit voorbeeld 0.3. De correlatie tussen twee gelijke items is 0. Dit resulteert in een similariteitsmatrix $\mathbf{M}$ met n^2 elementen.

$$s_{1,3} = \frac{1/2 + 1/3}{2 \times 2^{0,3}} = 0.338$$

$$\mathbf{M} = \begin{bmatrix} 0 & 0 & 0.406 & 0 \\ 0 & 0 & 0.135 & 0.338 \\ 0.25 & 0.135 & 0 & 0.135 \\ 0 & 0.338 & 0.135 & 0 \end{bmatrix}$$

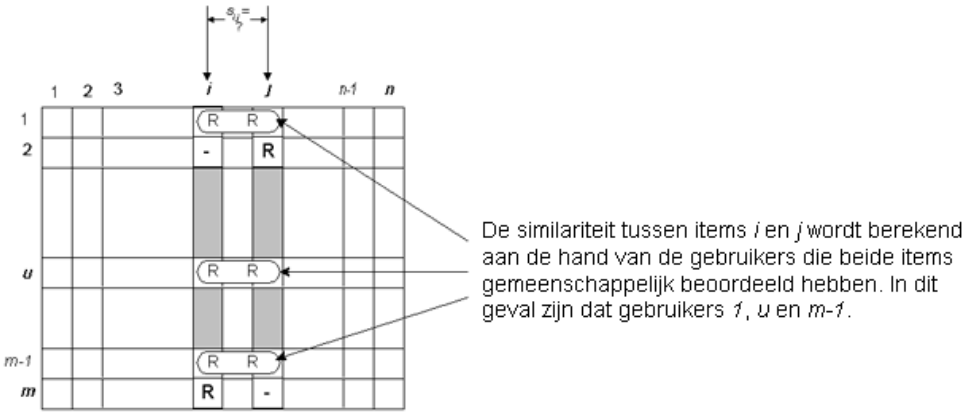
Uitbreidingen

Inverse item frequency Een uitbreiding van vectorsimilariteit en Pearsoncorrelatie is het gebruik van inverse item frequency. Opnieuw is de uitwerking hiervan gelijkaardig aan die van *inverse user frequency*. De kleine aanpassingen worden getoond aan de hand van een voorbeeld. Eerst en vooral wordt de interactiematrix $\mathbf{R}$ genormaliseerd op product, wat $\mathbf{R}'$ oplevert (zoals hiervoor). Vervolgens wordt matrix $\mathbf{R}''$ berekend. Deze houdt rekening met de frequentie van de voorkeur voor de items door een gebruiker. Bij de eerste gebruiker is dit dus $f_0 = \log \frac{4}{2} = 0.30103$, het logaritme van het totaal aantal items gedeeld door het aantal door die gebruiker beoordeelde items. Tenslotte

wordt de correlatiematrix **M** berekend via Pearsoncorrelatie of via vectorsimilariteit (zoals in het voorbeeld).

$$\mathbf{R}'' = \begin{bmatrix} 0 & 0,15051 & 0 & 0,15051 \\ 0 & 0,06247 & 0,06247 & 0,06247 \\ 0,3103 & 0 & 0,15052 & 0 \end{bmatrix} \quad \mathbf{M} = \begin{bmatrix} 1 & 0 & 0,92 & 0 \\ 0 & 1 & 0,15 & 1 \\ 0,92 & 0,15 & 1 & 0,15 \\ 0 & 1 & 0,15 & 1 \end{bmatrix}$$

Aangepaste cosinussimilariteit Een groot verschil bij de berekening van de gelijkaardigheid in user-based CF en item-based CF is dat in het geval van user-based CF de gelijkaardigheid berekend wordt over de rijen van de matrix terwijl dit bij item-based CF over de kolommen berekend wordt, met andere woorden elk paar in de gemeenschappelijk beoordeelde set behoort tot verschillende gebruikers (Figuur 4.1).



Figuur 4.1: Aangepaste cosinussimilariteit: isolatie van de gemeenschappelijk beoordeelde items en similariteitsberekening

Het berekenen van similariteiten gebruik makend van de gewone Pearsoncorrelatie heeft in het item-based geval een groot nadeel: het verschil in de beoordelingsschaal tussen verschillende gebruikers wordt niet meegerekend [50]. De aangepaste cosinussimilariteit houdt rekening met dit nadeel door het corresponderende gebruikersgemiddelde af te trekken van elk gemeenschappelijk beoordeeld paar. Daardoor wordt de similariteit tussen items a en b berekend door onderstaande formule waarbij $\bar{r}_u$ het gemiddelde is van de beoordelingen van de gebruiker.

$$s_{a,b} = \frac{\sum_{p \in P}(r_{p,a} - \bar{r}_u)(r_{p,b} - \bar{r}_u)}{\sqrt{\sum_{p \in P}(r_{p,a} - \bar{r}_u)^2}\sqrt{\sum_{p \in P}(r_{p,b} - \bar{r}_u)^2}}$$

4.3.2 Aanbevelingen genereren

M is de matrix die gebruikt wordt om voorspellingen te doen. Het voorbeeld wordt uitgewerkt voor de tweede gebruiker p_1, de gebruiker waarvoor de transactiegegevens op de tweede rij van **R** staan.

$$\mathbf{X} = \begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0.406 & 0 \\ 0 & 0 & 0.135 & 0.338 \\ 0.25 & 0.135 & 0 & 0.135 \\ 0 & 0.338 & 0.135 & 0 \end{bmatrix} = \begin{bmatrix} 0.25 & 0.437 & 0.27 & 0.473 \end{bmatrix}$$

Daarna worden de items die al door de gebruiker aangekocht werden op 0 gezet. Die items mogen of moeten niet meer aanbevolen worden.

$$\begin{bmatrix} 0,25 & 0 & 0 & 0 \end{bmatrix}$$

In dit eenvoudige voorbeeld wordt er slechts één item aanbevolen, namelijk het eerste.

4.3.3 Popalgoritme

Zoals eerder vermeld werd, zijn er nog andere manieren om item-based te werken en hiervan is het popalgoritme een voorbeeld. Het is een heel eenvoudig algoritme, maar zal dan ook niet altijd als goed geëvalueerd worden. Dit algoritme zal telkens de hoogst beoordeelde items aanraden.

Bij dit algoritme worden alle gebruikers aanzien als afkomstig van eenzelfde globale cluster en worden de aanbevelingen voor alle gebruikers gebaseerd op globale gemiddelden van de ratings [15]. De items die gemiddeld de beste beoordeling krijgen van alle gebruikers, worden eerst aangeraden.

4.3.4 Efficiëntie

De efficiëntie van item-based top-N-aanbevelingsalgoritmen hangt af van de tijd nodig om de similariteiten te berekenen tussen de verschillende items en van de tijd nodig om de aanbeveling te berekenen, gebruik makend van de similariteiten. Tijdens de similariteitsberekening wordt de gelijkaardigheid berekend tussen elk product en alle andere producten en vervolgens worden de k meest gelijkaardige geselecteerd. De complexiteit is dus $O(mn^2)$ aangezien $n \times (n-1)$ similariteiten berekend moeten worden, elk potentieel m operaties vragend. Hoe dan ook is de echte efficiëntie heel wat beter aangezien

de item-itemsimilariteitsmatrix heel ijl is doordat elke klant slechts weinig aankopen doet. Door dus gebruik te maken van kleine datastructuren om de interactiematrix op te slaan en de similariteiten enkel te berekenen tussen paren items die samen werden gekocht door ten minste één klant, kan de efficiëntie substantieel verkleind worden tot $O(mn)$ [31].

De tijd nodig om de top-N-aanbevelingen te berekenen voor de actieve gebruiker p wordt gegeven door $O(k\|p\|)$ doordat de k meest gelijkaardige items moeten benaderd worden voor alle voorkeuren van p. Hierbij worden de N meest gelijkaardige items bepaald voor gebruiker p.

Zoals bij de inleiding van item-based CF vermeld, kan het geheugengebruik herleid worden tot $O(kn)$, aangezien bij elk item slechts die k items bijgehouden worden die het meest relevant zijn.

4.3.5 Problemen

Een aantal van de problemen die besproken werden bij user-based Collaborative Filtering zijn reeds opgelost door item-based te werken. Toch zijn er enkele die blijven voorkomen en een nieuw probleem.

Tijd. Item-based CF-systemen hebben heel wat tijd nodig om hun model op te bouwen. Dit probleem is echter gemakkelijk op te lossen door het model offline te construeren, waardoor de gebruiker hier nooit last van heeft. De server zal natuurlijk wel geruime tijd berekeningen moeten doen, waardoor deze ondertussen minder bruikbaar is voor andere doeleinden. Toch zal de gebruiker overdag nog steeds nieuwe aanbevelingen verkrijgen, doordat er nog een online berekening is, die heel wat minder tijd inneemt.

Sparsity of ijlheid. Ook hier is er nog altijd sprake van het ijlheidsprobleem dat bij user-based CF werd besproken. In hoofdstuk 4.4 worden hier oplossingen voor gegeven.

Aanvallen. Opnieuw is het mogelijk om een 'copy profile attack' uit te voeren. Het systeem zal er wel anders op reageren, maar het is opnieuw zo dat verschillende personen dezelfde ratings ingegeven hebben, waardoor ook de itemsimilariteiten vergroten voor de gemeenschappelijke items.

Synoniemen. Ook hier kunnen er problemen ontstaan door synoniemen. Meer uitleg hierover kan gevonden worden bij 4.2.6.

4.4 Andere benaderingen

4.4.1 Eigentaste

Het doel van eigentaste is accurate en efficiënte recommandaties te doen aan gebruikers in een constante online tijd [15]. Het algoritme profileert de smaak van een individu met universele vragen: elk item wordt voorgesteld met een korte, onpartijdige omschrijving (bijvoorbeeld een samenvatting van een boek of een film) zodat de gebruikers zich een mening kunnen vormen en op elke vraag kunnen antwoorden. Eigentaste vraagt aan elke gebruiker dezelfde testset van items te beoordelen in de profileringsfase. Dit heeft het voordeel dat de subset van de interactiematrix die de testset bevat dicht is. Universele ratings laten ook het systeem toe om directe feedback te hebben op alle aangeraden items.

Eigentaste maakt gebruik van een continue schaal, in tegenstelling tot de meeste andere algoritmen. Om items te beoordelen wordt aan gebruikers gevraagd om met hun muis op een horizontale beoordelingsbalk te klikken. Terwijl het technisch niet mogelijk is om continu te werken (denk maar aan de limieten van granulariteit bij HTML image maps), kunnen ongeveer 200 niveaus van ratings onderscheiden worden. Continue beoordelingen vermijden discretisatie-effecten bij matrixberekeningen.

Eigentaste splitst de berekeningen op in online en offline fases. Offline gebruikt eigentaste 'principal component analysis' voor optimale dimensionaliteitsreductie en clustert dan gebruikers in de kleinere dimensionale subruimte. De online fase maakt gebruik van eigenvectoren om nieuwe gebruikers te projecteren in clusters en van een opzoekingstabel om geschikte items aan te raden, zodat de runtime onafhankelijk is van het aantal gebruikers in de database.

Het algoritme bestaat uit een aantal aparte delen die hieronder verduidelijkt worden.

Normaliseren van de ratings

Uit de interactiematrix $\mathbf{R}$ worden k producten geselecteerd om de testset te vormen (de geldige gebruikers zijn dus die die alle items in de testset hebben beoordeeld). Deze subset van $\mathbf{R}$ wordt genormaliseerd om $\mathbf{A}$ te bekomen, een $k \times m$ submatrix van alle ratings gegeven aan de testset.

Elke rating wordt genormaliseerd door er de gemiddelde rating voor dat item van af te trekken en het vervolgens te delen door de standaarddeviatie van de ratings voor dat item. Aangezien geldige gebruikers alle items uit de testset beoordeeld hebben, zijn er geen 'null'-ratings. De gemiddelde rating van het j^{de} item in de testset is:

$$\mu = \frac{1}{m} \sum_{i \in U_j} r_{i,j}$$

en de variantie ervan is:

$$\sigma_j^2 = \frac{1}{m-1} \sum_{i \in U_j} (r_{i,j} - \mu_j)^2$$

In $\mathbf{A}$, wordt de genormaliseerde rating $r'_{i,j}$ bepaald door $\frac{r_{i,j} - \mu_j}{\sigma_j}$

Pearson correlatiematrix

Wanneer een continue ratingschaal en een lineaire relatie tussen variabelen wordt verondersteld, kan de globale correlatiematrix $\mathbf{C}$ gedefinieerd worden over alle gebruikers. Hierbij is $\mathbf{C}$ symmetrisch en positief gedefinieerd.

$$\mathbf{C} = \frac{1}{m-1} \mathbf{A^T A}$$

Principal component analysis

Principal component analysis werd voor het eerst voorgesteld in 1901 door Karl Pearson [42]. Hotelling generaliseerde het voor random variabelen in 1933 [21]. Eigenanalyse wordt hier toegepast om op te lossen naar matrices $\mathbf{E}$ en $\mathbf{\Lambda}$ zodat

$$\mathbf{C} = \mathbf{E^T \Lambda E}$$

en

$$\mathbf{E C E^T} = \mathbf{\Lambda}$$

Laat $\mathbf{B} = \mathbf{A E^T}$ een lineaire transformatie zijn voor $\mathbf{A}$ zodat de getransformeerde punten ongecorreleerd zijn (de correlatiematrix is diagonaal):

$$\mathbf{C_B} = \frac{1}{m-1} \mathbf{B^T B} = \mathbf{E C E^T} = \mathbf{\Lambda}$$

Elke kolom j van $\mathbf{B}$ heeft variantie λ_j. Het idee is om enkel de hoofdzakelijke eigenvectoren te behouden. Het aantal eigenvectoren dat weerhouden wordt hangt af van de

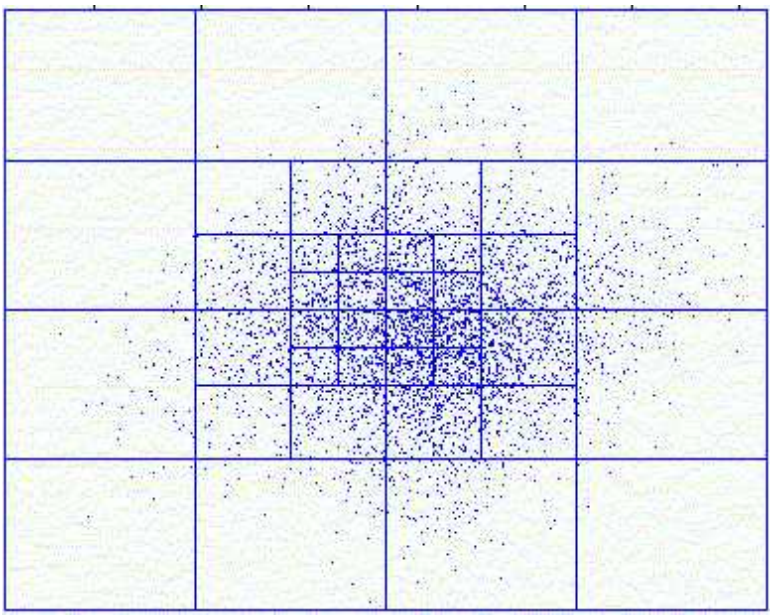
varianties (eigenwaarden) maar is meestal klein. Als v eigenvectoren behouden blijven, wordt de data geprojecteerd over de eerste v eigenvectoren:

$$x = \mathbf{RE_v^T}$$

Recursief rechthoekig clusteren

Er zijn veel verschillende manieren om de geprojecteerde data te clusteren. Bij recursief rechthoekig clusteren neemt de celgrootte af bij de oorsprong. Dit kan veralgemeend worden voor hogere dimensies en een variëteit aan alternatieve clusteringmethoden kan gebruikt worden bij eigentaste. Het clusteren gaat hier als volgt:

1. Start met de minimale rechthoekige cel die alle punten (gebruikersprojecties) van het eigenvlak bevat. Dit vormt de buitenste rechthoekige subdivisie.
2. Deel deze cel op over de x- en y-as om vier rechthoekige subcellen te bekomen.
3. Voor elke nieuwe subcel die de oorsprong als één van zijn punten heeft, voer opnieuw vorige stap uit om subcellen te genereren op het volgende hiërarchische niveau.
4. Herhaal stap drie voor elk niveau totdat een gewenste diepte bereikt wordt.



Figuur 4.2: Eigentaste: 4 niveaus van recursie die zorgen voor 40 clusters.

Dit principe wordt aangeduid in Figuur 4.2. Elke cel wordt behandeld als een cluster van buren in het eigenvlak. Per cluster wordt per item dat zich niet in de testset bevindt, het gemiddelde berekend, gebaseerd op het aantal gebruikers die het beoordeeld hebben. Door de items die niet in de testset staan dalend te sorteren voor gemiddelde ratings, wordt een opzoekingstabel van aanbevelingen opgesteld voor die cluster.

Online berekening van aanbevelingen

De vorige stappen gebeurden allemaal offline. Maar wanneer een nieuwe gebruiker in het systeem komt, gebeurt het volgende:

1. Verzamel ratings voor alle items in de testset.
2. Gebruik de hoofdzakelijke eigenvectoren om de k-vector te projecteren op het eigenvlak.
3. Zoek de representatieve cluster.
4. Zoek geschikte aanbevelingen, presenteer ze aan de nieuwe gebruiker en verzamel ratings.

4.4.2 Dimensionaliteitsreductiealgoritme

Dit algoritme condenseert de originele interactiematrix $\mathbf{R}$ en genereert aanbevelingen op basis van de gecondenseerde, minder ijle matrix om het tekort aan gegevens op te lossen [49, 5]. Hierbij wordt gebruik gemaakt van de standaard Singular Value Decomposition (SVD) om de interactiematrix $\mathbf{R}$ op te delen in $\mathbf{U}.\mathbf{Z}.\mathbf{V}^T$ waarbij $\mathbf{U}$ en $\mathbf{V}$ twee orthogonale matrices zijn met respectievelijk groottes $m \times r$ en $n \times r$ en r is de rang van matrix $\mathbf{R}$. $\mathbf{Z}$ is een diagonale matrix met grootte $r \times r$ waarbij alle enkelvoudige waarden van matrix $\mathbf{R}$ op de diagonaal staan. Het algoritme zal vervolgens $\mathbf{Z}$ reduceren waarbij enkel de k grootste waarden overblijven om zo $\mathbf{Z}_k$ te vormen. $\mathbf{U}$ en $\mathbf{V}$ worden vervolgens ook zo gereduceerd om $\mathbf{U}_k$ en $\mathbf{V}_k$ te bekomen. $\mathbf{U}_k.\mathbf{Z}_k.\mathbf{V}_k^T$ levert dan de beste laagsterangsbenadering van de originele interactiematrix $\mathbf{R}$, waarbij de primaire datapatronen behouden blijven nadat de ruis verwijderd werd. Gebruikersgelijkaardigheden kunnen dan afgeleid worden van de compacte representatie gebaseerd op $\mathbf{U}_k$ en $\mathbf{Z}_k^{1/2}$. Vervolgens zal het algoritme aanbevelingen genereren op dezelfde manier als de user-based algoritmen.

Op de reeds gebruikelijke interactiematrix $\mathbf{R}$ toegepast levert dit onderstaande matrices op. Eerst en vooral wordt SVD toegepast waaruit matrices $\mathbf{U}, \mathbf{Z}$ en $\mathbf{V}^T$ volgen. Vervolgens wordt de compacte gebruikersrepresentatiematrix berekend. Hierop wordt (in

dit voorbeeld) de vectorsimilariteit berekend waardoor de gebruikerssimilariteitsmatrix ontstaat. Vervolgens wordt via het matrixproduct tussen **R** en de gebruikerssimilariteitsmatrix berekend wat de potentiële scores zijn.

$$\mathbf{R} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} =$$

$$\begin{bmatrix} 0.56 & 0.45 & -0.70 \\ 0.78 & 0 & 0.63 \\ 0.28 & -0.89 & -0.35 \end{bmatrix} \cdot \begin{bmatrix} 2.19 & 0 & 0 \\ 0 & 1.41 & 0 \\ 0 & 0 & 0.46 \end{bmatrix} \cdot \begin{bmatrix} 0.13 & 0.61 & 0.48 & 0.61 \\ -0.6 & 0.32 & -0.6 & 0.32 \\ -0.8 & -0.2 & 0.6 & -0.2 \end{bmatrix}$$

Singulaire vectordecompositie: $A = U.Z.V^T$

$$\begin{bmatrix} 0.83 & 0.53 & 0 \\ 1.15 & 0 & 0 \\ 0.41 & -1.06 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0.84 & -0.2 \\ 0.84 & 1 & 0.36 \\ -0.2 & 0.36 & 1 \end{bmatrix} \quad \begin{bmatrix} -0.2 & 1.84 & 0.64 & 1.84 \\ 0.36 & 1.84 & 1.36 & 1.84 \\ 1 & 0.16 & 1.36 & 0.16 \end{bmatrix}$$

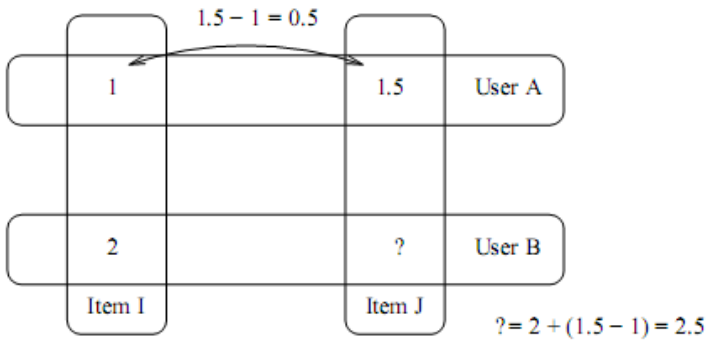
Compacte gebruikers-representatie $\mathbf{U}.\mathbf{Z}_2^{1/2}$	Gebruikersgelijkaardigheids-matrix **M**	Potentiëlescore-matrix

Het is bewezen dat dit algoritme heel wat slechter werkt dan de traditionele Collaborative Filteringsystemen wanneer er gesproken wordt over een heel ijle e-commerce dataset. Bij dichtere datasets levert het algoritme echter heel goede resultaten op [49].

4.4.3 Slope one

Slope one-algoritmen werken via het intuïtieve principe van een 'populariteitsdifferentiaal' tussen items voor individuen [28]. Op een paarsgewijze manier wordt onderzocht hoeveel beter het ene item is dan het andere. Eén manier om die differentiaal te berekenen is simpelweg door de gemiddelde rating van de twee items af te trekken. Dit verschil kan dan op zijn beurt gebruikt worden om de rating van een andere gebruiker te voorspellen voor één van die items, gegeven de rating van de andere.

Neem twee gebruikers a en b in beschouwing en twee items i en j (Figuur 4.3). Er kan uit afgeleid worden dat item j hoger beoordeeld wordt dan item i met $1.5 - 1 = 0.5$ punten, dus kan er voorspeld worden dat gebruiker b aan item j $2 + 0.5 = 2.5$ punten zal geven. Gebruiker b is hier de actieve gebruiker en item j is het voorspelde item. Veel van die differentialen bestaan in een trainingset voor elke ongekende rating en van die differentialen wordt het gemiddelde genomen.



Figuur 4.3: Slope one: voorbeeld

De grootste sterkte van slope one is het feit dat er niet met alle data rekening gehouden wordt: enkel die ratings van gebruikers die gemeenschappelijke ratings hebben met de actieve gebruiker en enkel die ratings van items die de actieve gebruiker heeft ingegeven. Dit algoritme is echter niet bruikbaar voor binaire ratings.

Gegeven de twee evaluatietabellen r_v en r_w die alle ratings bevatten van personen v en w. Er wordt gezocht naar de beste voorspeller van de vorm $f(x) = x + b$ om de rating voor w te bepalen uit v door $\sum_i (v_i + b - w_i)^2$ te minimaliseren. Afleiden naar b en gelijk stellen aan nul geeft $b = \frac{\sum_i w_i - v_i}{n}$. Met andere woorden, de constante b moet zo gekozen worden dat ze gelijk wordt aan het gemiddelde verschil tussen de twee tabellen. Deze gedachtegang wordt gebruikt in onderstaand schema.

Gegeven een trainingset χ en elke twee items j en i met ratings $r_{u,j}$ en $r_{u,i}$ respectievelijk voor een gebruiker u (geschreven als $u \in S_{j,i}(\chi)$), dan is de gemiddelde afwijking van item i ten opzichte van item j:

$$dev_{j,i} = \sum_{u \in S_{j,i}(\chi)} \frac{r_{u,j} - r_{u,i}}{card(S_{j,i}(\chi))}$$

Hierbij wordt geen rekening gehouden met alle gebruikers u die geen ratings hebben voor items i of j. $card(S_{j,i}(\chi)$ duidt de grootte aan van de verzameling. De symmetrische matrix gedefinieerd door $dev_{j,i}$ kan eenmalig berekend worden en snel geüpdatet worden wanneer nieuwe data beschikbaar zijn, zoals geïmplementeerd in [29].

Gegeven dat $dev_{j,i} + r_{u,i}$ een voorspelling is voor $r_{u,j}$ gegeven $r_{u,i}$, dan is een acceptabele voorspeller mogelijk door het gemiddelde te nemen van al die voorspellingen:

$$P(u,j) = \frac{1}{card(R_j)} \sum_{i \in R_j} (dev_{j,i} + r_{u,i})$$

waarbij $R_j = \{i | i \in S(u), i \neq j, card(S_{j,i}(\chi)) > 0\}$ de set van alle relevante items is. Er is een benadering die de berekening van deze voorspelling kan vereenvoudigen. Voor een dichte dataset waarbij bijna alle paren van items ratings hebben, dus waarbij $card(S_{j,i}(\chi)) > 0$ voor bijna alle i en j, zal meestal $R_j = S(u)$ voor $j \notin S(u)$ en $R_j = S(u) - \{j\}$ als $j \in S(u)$. Aangezien $\bar{u} = \sum_{i \in S(u)} \frac{r_{u,i}}{card(S(u))} \approx \sum_{i \in R_j} \frac{r_{u,i}}{card(R_j)}$ voor de meeste j, kan de voorspellingsformule vereenvoudigd worden tot het slope oneschema:

$$P^{S1}(u,j) = \bar{u} + \frac{1}{card(R_j)} \sum_{i \in R_j} dev_{j,i}$$

Het is interessant op te merken dat deze implementatie van slope one niet afhankelijk is van hoe de gebruiker individuele items heeft beoordeeld, maar enkel van de gemiddelde rating van de gebruiker.

Een van de grootste nadelen van slope one is dat het aantal bekeken ratings niet meegerekend wordt. Intuïtief kan gezien worden dat om gebruiker a's rating voor item l te voorspellen, gegeven gebruiker a's ratings voor items j en k, als 2000 individuen het paar items j en l hebben beoordeeld en slechts 20 individuen k en l, dan zal gebruiker a's rating voor item j een betere voorspeller zijn voor item l dan gebruiker a's rating voor k. Daarom werd de gewogen slope one ontwikkeld waarbij $c_{j,i} = card(S_{j,i}(\chi))$:

$$P^{wS1}(r_{u,j}) = \frac{\sum_{i \in S(u)-\{j\}} (dev_{j,i} + r_{u,i}) c_{j,i}}{\sum_{i \in S(u)-\{j\}} c_{j,i}}$$

Van dit algoritme is bewezen dat het gemakkelijk te implementeren is, maar toch nog steeds even goed werkt als duurdere schema's. Met dit algoritme zijn vijf doelen bereikt [28]:

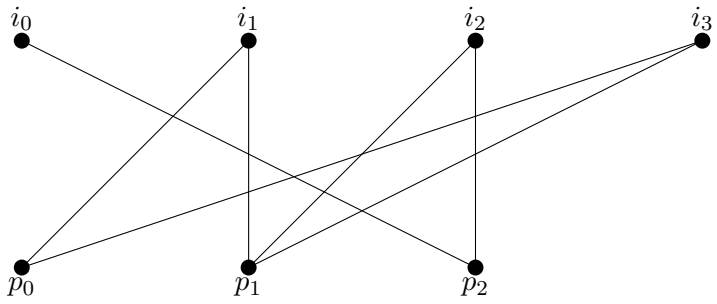
- gemakkelijk implementeerbaar en onderhoudbaar,
- on the fly updateable,
- korte querytijd,
- er wordt weinig verwacht van bezoekers met weinig ratings,
- accuraat.

Het gemiddelde ratingverschil berekenen voor elk paar items heeft $O(n(n-1)/2)$ plaats nodig en $O(mn^2)$ tijd. Deze grens is opnieuw heel pessimistisch aangezien gebruikers meestal niet zoveel items hebben beoordeeld. Stel dat dit maximaal y is, dan wordt het ratingverschil berekenen niet meer dan $O(n^2 + my^2)$. Wanneer een gebruiker x beoordelingen gaf, dan zal een rating voorspellen $O(x)$ tijd vragen en alle ontbrekende ratings voorspellen $O((n-x)x)$.

4.4.4 Spreading-activationalgoritme

Het spreading-activationalgoritme wordt hier vooral vermeld ter inleiding op het link-analysisalgoritme dat hierop volgt, aangezien het het eerste CF-algoritme is dat gebruik maakt van een graaf. Het doel is het ijlheidsprobleem en daarbij horend het cold-start problem aan te kaarten door de interactiematrix $\mathbf{R}$ meningsvol op te vullen, zodat deze dicht wordt. Dit kan bekomen worden door transitieve verbindingen tussen gebruikers en items te ontdekken in een bipartite gebruiker-productgraaf die overeenkomt met de interactiematrix $\mathbf{R}$. Stel dat gebruikers p_0 en p_1 het product i_1 kochten en gebruikers p_1 en p_2 het product i_2 kochten. Standaard CF-benaderingen die geen rekening houden met transitieve verbindingen, zullen p_0 associëren met p_1 en ook p_1 met p_2, maar nooit p_0 met p_2. Een benadering die rekening houdt met transitieve interacties zal, hoe dan ook, deze associatieve relatie tussen p_0 en p_2 herkennen en zal deze transitieve verbindingen in de interactiematrix $\mathbf{R}$ toevoegen. Dit algoritme is wel enkel toepasbaar voor binaire ratings.

Het spreading-activationalgoritme ontwikkeld in de associatieve information retrieval werd aangepast om efficiënt te wandelen door de gebruiker-productgraaf en de transitieve verbindingen te ontdekken. Het verkennen van transitieve verbindingen in de context van aanbevelingssystemen, wordt uitgericht in een graafgebaseerd aanbevelingsmodel en dit omwille van twee redenen. Eerst en vooral is een graaf gemakkelijk te begrijpen en geeft dit een natuurlijk en algemeen framework voor verschillende types applicaties, waaronder aanbevelingssystemen. Ten tweede zijn reeds een hele set van graafgebaseerde algoritmen toepasbaar wanneer de aanbevelingstaak geformuleerd wordt als een graaftheoretisch probleem.



Veronderstel bovenstaande bipartite graaf en dat het aanbevelingssysteem producten moet aanraden voor de eerste gebruiker (p_0). Standaard CF-benaderingen zullen enkel rekening houden met paden van lengte 3 (bijvoorbeeld: $p_2 - i_2 - p_1 - i_1$, waarbij i_1 zal aan geraden worden aan p_2). Associatie tussen p_0 en i_0 bestaat niet aangezien er geen pad tussen is van lengte 3. Intuïtief kan gezien worden dat hoe hoger het aantal afzonderlijke paden is dat de productnode en de gebruikersnode verbindt, hoe hoger de

associatie is tussen die twee nodes. Dus is het zeker de moeite waard om dit product aan te raden aan die gebruiker.

De input voor het algoritme is de interactiematrix $\mathbf{R}$ waaruit de bipartite graaf kan geconstrueerd worden. Gegeven een gebruikersnode p_t en een productnode i_j, dan is de associatie r_{p_t,i_j} gedefinieerd als de som van de gewichten van alle afzonderlijke paden die p_t en i_j verbinden. In deze berekening wordt enkel rekening gehouden met de paden waarvan de lengte kleiner is dan of gelijk aan lengte M. Deze paramater kan gecontroleerd worden door de ontwerper, maar is best oneven aangezien de transitieve associaties gerepresenteerd worden in een bipartite graaf. Voor een pad van lengte x (met $x \leq M$) wordt het gewicht berekend als α^x, waarbij α een constante is gelegen tussen 0 en 1, zodat er verzekerd wordt dat langere paden een kleinere impact hebben. De ontwerper kan deze α kiezen: een hoge α wordt best gekozen als transitieve relaties een sterke voorspeller zijn voor interesse van een gebruiker. In het voorbeeld toegepast levert dit, met $M = 3$ (standaard CF), $r_{p_0,i_2} = 0.5^3 + 0.5^3 = 0.25$ en $r_{p_0,i_0} = 0$.

Voor gebruiker p_t wordt bovenstaande associatieberekening herhaald voor alle items $i_j \in I$. De producten in I worden vervolgens gesorteerd in dalende volgorde volgens r_{p_t,i_j}. De eerste k items (behalve die die p_t eerder beoordeelde) worden vervolgens aangeraden aan p_t.

Het bovenstaande proces wordt nu beschreven aan de hand van de matrixnotatie. Gegeven de interactiematrix $\mathbf{R}$, het padgewicht α en de maximum padlengte M, dan worden de transitieve associaties tussen producten en gebruikers gegeven in matrix $\mathbf{R}_\alpha^M$.

$$\mathbf{R}_\alpha^M = \begin{cases} \alpha R & \text{als } M = 1 \\ \alpha^2 R.R^T.R_\alpha^{M-2} & \text{als } M = 3, 5, 7, \ldots \end{cases}$$

In bovenstaand numeriek voorbeeld met interactiematrix $\mathbf{R}$ en $\alpha = 0.5$, zijn de transitieve associaties voor $M = 3$ en $M = 5$:

$$\mathbf{R}_{0.5}^3 = \begin{bmatrix} 0 & 0.5 & 0.25 & 0.5 \\ 0.125 & 0.625 & 0.5 & 0.625 \\ 0.25 & 0.125 & 0.375 & 0.125 \end{bmatrix} \quad \mathbf{R}_{0.5}^5 \begin{bmatrix} 0.625 & 0.5625 & 0.375 & 0.5625 \\ 0.15625 & 0.75 & 0.59375 & 0.75 \\ 0.15625 & 0.21875 & 0.3125 & 0.21875 \end{bmatrix}$$

De grootste uitdaging om de bovenstaande benadering te implementeren, is het berekenen van $\mathbf{A}^*$, aangezien dit uitgebreide bronnen nodig heeft om te berekenen, vooral wanneer er veel gebruikers- en productnodes zijn en als M groot is. Hiervoor wordt het Hopfieldnetalgoritme gebruikt, dat zorgt voor een competitieve performantie in aanbevelingssystemen [23]. Het graafmodel van Collaborative Filtering mapt de verbonden neuronen en synapsen in het Hopfieldnet met de neuronen die de gebruikers en de items

voorstellen en de synapsen die de interactie voorstellen tussen gebruikers en items. Het Hopfieldnetactivatiealgoritme werkt als volgt:

Initialisatie. De gebruikersnode die de actieve gebruiker voorstelt wordt geïnitialiseerd met activatieniveau 1. Alle andere nodes krijgen activatieniveau 0.

Activatie en activatieniveauberekening. Een vast aantal nodes (die met het grootste activatieniveau) worden geactiveerd. Het activatieniveau voor elke node wordt berekend als:

$$\mu_j(t+1) = f_s[\sum_{i=0}^{n-1} t_{ij}\mu_i(t)] \text{ met } 0 \leq j \leq n-1$$

met f_s de continue sigmoïdetransformatiefunctie

$$f_s(x) = \frac{1}{1+\exp((\theta_1 - x)/\theta_2)},$$

$\mu_j(t+1)$ het activatieniveau van node j bij iteratie $t+1$ en t_{ij} is het gewicht van de verbinding die node i en j connecteert. Elke nieuwe geactiveerde node berekent zijn activatieniveau gebaseerd op de som van de producten van de activatieniveaus van de buren en hun synapsen. De controleparameters θ_1 en θ_2 van de sigmoïdefunctie kunnen heuristisch bepaald worden.

Stopvoorwaarde. Het bovenstaande proces wordt herhaald totdat

$$\sum_j \mu_j(t+1) - \sum_j \mu_j(t) < \epsilon \times t$$

gehaald wordt, aangevend dat er geen significante verandering is tussen de laatste twee iteraties. In deze voorwaarde is ϵ een klein positief getal. De toelaatbare verandering is proportioneel met het aantal iteraties die uitgevoerd worden om de convergentie te versnellen. De beste productnodes met de hoogste activatieniveaus zullen uiteindelijk aangeraden worden nadat alle items eruit gehaald zijn die de gebruiker reeds kent of beoordeelde.

Dit algoritme zal efficiënt de verbondenheid onderzoeken van een gebruiker-productpaar in de gebruiker-productgraaf. Verbondenheid verhoudt zich met het aantal paden tussen de paren en de lengtes en is de voorspeller van het voorkomen van een toekomstige interactie.

4.4.5 Link-analysisalgoritme

Het link-analysisalgoritme bij aanbevelingssystemen is ontstaan als reactie op het ijlheidsprobleem bij traditionele CF-systemen. Ze kunnen enkel gebruikt worden bij binaire ratings. Link-analysisalgoritmen hebben hun nut bewezen in webpage ranking en social-network analyse, zoals HITS en PageRank. Onderstaand algoritme [24] past het HITS-algoritme (Hypertext-Induced Topic Selection) aan aan de aanbevelingscontext. Het originele HITS-algoritme maakt onderscheid tussen twee types webpagina's die over een bepaalde topic gaan:

- *Authorative* pagina's bevatten informatie met hoge kwaliteit.
- *Hub*pagina's bevatten hoofdzakelijk links naar authorative pagina's.

Volgens dit model is er een sterk verband tussen hubs en authoraties: goede hubs verwijzen naar vele goede authoraties en er wordt naar goede authoraties verwezen vanuit vele goede hubs. Hubs en authoraties zouden dus communities moeten vormen die kunnen gezien worden als dichte bipartite delen van het Web, waarbij de hubs linken aan nabije authoraties [22].

In het aanbevelingsalgoritme vormt de user-itemrelatie een bipartite graaf bestaande uit gebruikers- en productnodes. Een link tussen een gebruikersnode p en een productnode i betekent dat p een interesse kan voorstellen voor i en dat i gedeelte kan uitmaken van p's gebruikersbasis. Vergeleken met webpage ranking, zal een recommender eerder items moeten identificeren waarin interesse is, dan items die populair zijn.

Volgende gegevens worden gedefinieerd:

- $pr(i, p_0)$ de productrepresentativiteitscore van product i voor gebruiker p_0. Dit kan gezien worden als een meting voor de grootte van interesse die gebruiker p_0 heeft voor product i.
- $cr(p, p_0)$ de gebruikersrepresentativiteitscore van gebruiker p voor gebruiker p_0. Dit meet hoe goed p, als 'hub' voor p_0, overeenkomt met de interesses van p_0.

In plaats van gebruik te maken van een vectorrepresentatie, zoals bij het originele HITS-algoritme, wordt hier gebruik gemaakt van $\mathbf{PR} = (pr_{ip})$ om een $m \times n$ productrepresentativiteitsmatrix voor te stellen, waarbij $pr_{ip} = pr(i, p)$ de productrepesentativiteit voorstelt van product i voor gebruiker p. Een $m \times m$ matrix $\mathbf{CR} = (cr_{it})$ wordt gebruikt om alle gebruikersrepresentativiteiten voor te stellen. Gebruik makend van de recursieve definities van authoraties en hub scores, kunnen de product- en gebruikersrepresentativiteitscores gedefinieerd worden als $\mathbf{PR} = \mathbf{R}^T.\mathbf{CR}$ en $\mathbf{CR} = \mathbf{R}.\mathbf{PR}$.

Intuïtief kan gezien worden dat de som van de representativiteitscores van producten gelinkt aan een gebruiker de gebruikersscore oplevert en vice versa. Dit zal uiteindelijk convergeren.

Deze uitbreidingen op de originele scoredefinities zorgen voor twee inherente problemen. Eerst en vooral, als een gebruiker verbindingen heeft met alle producten, zal die gebruiker de hoogste representativiteitscores hebben voor elke doelgebruiker. Maar, het gedrag van zo'n gebruiker levert maar weinig informatie om het gedrag van de doelgebruiker te voorspellen. Een fundamenteler probleem is dat van de convergentie-eigenschap: **PR** en **CR** moeten in principe convergeren naar matrices met identieke kolommen. Dit zorgt voor scores die productrankings voorstellen die onafhankelijk zijn van bepaalde gebruikers, dus enkel gelimiteerde waarde hechten aan de aanbeveling.

Om voormelde problemen te adresseren wordt **CR** herdefinieerd: $\mathbf{CR} = \mathbf{B}.\mathbf{PR} + \mathbf{CR_0}$ waarbij $\mathbf{B} = (b_{ij})$ een $m \times n$ matrix is, afgeleid van interactiematrix **R**:

$$b_{ij} = \frac{r_{ij}}{(\sum_j r_{ij})^\gamma} \tag{4.8}$$

en $\mathbf{CR_0}$ is de oorspronkelijke gebruikersrepresentatiematrix,

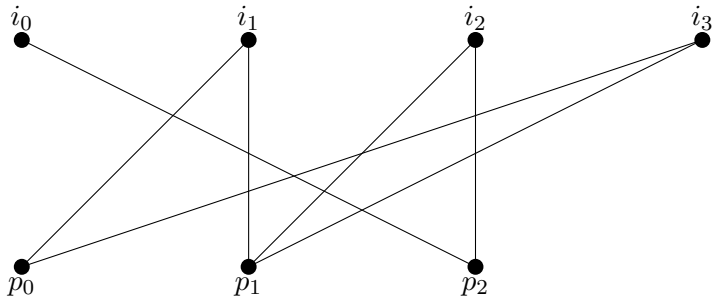
$$cr_{ij}^0 = \begin{cases} 1 & \text{als } i = j \\ 0 & \text{anders} \end{cases}$$

Deze definitie leent ideeën gelijkaardig aan die die succesvol werden gebruikt in netwerkspreidingsmodellen. De introductie van **B** normaliseert de representativiteitscores die een klant krijgt van gelinkte producten door het te delen door het totaal aantal producten waaraan hij/zij gelinkt wordt. Of met andere woorden, om even representatief te worden gezien als een andere gebruiker, zal een gebruiker die meer producten heeft aangekocht of beoordeeld, meer overlappende aankopen/beoordelingen moeten hebben met de doelgebruiker.

De parameter γ controleert de omvang van het *straffen* van een bepaalde gebruiker omdat hij/zij teveel aankopen/beoordelingen heeft gedaan. In het onderzoek naar spreidingsmodellen is zo'n aanpassing grondig bestudeerd. Er werden reeds experimenten uitgevoerd met verschillende γ in het bereik van 0 tot 1. De oorspronkelijke matrix $\mathbf{CR_0}$ wordt erbij betrokken om de hoge representativiteitscores voor de doelgebruikers zelf te behouden en om de score updates voor elke gebruiker aan te passen. Om consistente niveaus van de gebruikers zelfrepresentativiteit te behouden, wordt de matrixvermenigvuldiging $\mathbf{PR}.\mathbf{B^T}$ genormaliseerd. De normalisatie gebeurt hierbij zo dat elke kolom van $\mathbf{PR}.\mathbf{B^T}$ een totaal heeft van 1. Zo'n normalisatie helpt ook numerieke problemen te vermijden tijdens de herhaalde vermenigvuldigingen van grote matrices.

Eén van de grote voordelen van het link-analysisalgoritme is dat het kan werken met heel weinig gegevens en toch nog altijd heel performant blijft [22]. Met heel weinig gegevens betekent eigenlijk dat er wel nog altijd veel gebruikers en items zijn, maar de verhouding van het aantal beoordelingen die een gebruiker geeft ten opzicht van het aantal items is enorm klein.

Hieronder wordt het algoritme samengevat in een aantal stappen, telkens uitgelegd aan de hand van een voorbeeld.



Aan de hand van bovenstaande bipartite graaf wordt de interactiematrix $\mathbf{R}$ opgesteld. Vervolgens wordt associatiematrix $\mathbf{B}$ berekend aan de hand van formule 4.8 met hier als voorbeeld $\gamma = 0.7$:

$$\mathbf{R} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 0.62 & 0 & 0.62 \\ 0 & 0.46 & 0.46 & 0.46 \\ 0.62 & 0 & 0.62 & 0 \end{bmatrix} \quad \mathbf{CR_0} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Hierna wordt de oorspronkelijke gebruikersrepresentatiematrix $\mathbf{CR_0}$ geconstrueerd die tevens overeenkomt met de $\mathbf{CR(0)}$. Dit komt neer op het zetten van 1'en op de diagonaal van linksboven naar rechtsbeneden. Vervolgens wordt per iteratie t het volgende uitgevoerd:

1. $\mathbf{PR(t)} \ = \ \mathbf{CR(t-1)} \ . \ \mathbf{R}$
2. $\mathbf{CR(t)} \ = \ \mathbf{PR(t)} \ . \ \mathbf{B}^T$
3. Normaliseer $\mathbf{CR(t)}$ zodat $\sum_{i=1}^{m} cr_{ij} \ = \ 1$
4. $\mathbf{CR(t)} \ = \mathbf{CR(t)} \ + \ \mathbf{CR_0}$
   Voer voorgaande stappen uit totdat de matrices convergeren of na een vastgelegd aantal iteraties t.

Deze stappen zijn dus telkens het matrixproduct uitvoeren tussen twee verschillende matrices (met $\mathbf{B}^T$ de getransponeerde matrix). Bij $\mathbf{CR}$ is er echter nog het verschil dat

deze na het matrixproduct nog moet genormaliseerd worden, zoals eerder vermeld en hierbij vervolgens nog $\mathbf{R}$ moet opgeteld worden. In het voorbeeld werd dit normaliseren echter weggelaten om de matrixvermenigvuldiging gemakkelijk te kunnen volgen. Na drie iteraties worden onderstaande resultaten bekomen. Uit $\mathbf{PR}(\mathbf{3})$ blijkt dat aan de eerste gebruiker producten 3 en 1 worden aanbevolen (in die volgorde), aangezien producten 2 en 4 reeds gekend waren. Uit $\mathbf{CR}(\mathbf{3})$ volgt dat de eerste gebruiker het meest lijkt op de tweede gebruiker met een score van 9.46 en dan pas op de derde.

$$\mathbf{PR}(\mathbf{1}) \qquad \mathbf{PR}(\mathbf{2}) \qquad \mathbf{PR}(\mathbf{3})$$

$$\begin{bmatrix} 0.00 & 1.00 & 0.00 & 1.00 \\ 0.00 & 1.00 & 1.00 & 1.00 \\ 1.00 & 0.00 & 1.00 & 0.00 \end{bmatrix} \quad \begin{bmatrix} 0.00 & 3.16 & 0.93 & 3.16 \\ 0.62 & 3.62 & 3.01 & 3.62 \\ 2.23 & 0.46 & 2.69 & 0.46 \end{bmatrix} \quad \begin{bmatrix} 0.57 & 8.24 & 3.93 & 8.24 \\ 2.23 & 10.2 & 7.98 & 10.2 \\ 4.03 & 2.25 & 5.71 & 2.25 \end{bmatrix}$$

$$\begin{bmatrix} 2.23 & 0.93 & 0 \\ 1.23 & 2.39 & 0.62 \\ 0 & 0.46 & 2.23 \end{bmatrix} \quad \begin{bmatrix} 4.89 & 3.36 & 0.57 \\ 4.46 & 5.75 & 2.23 \\ 0.57 & 1.68 & 4.03 \end{bmatrix} \quad \begin{bmatrix} 11.2 & 9.46 & 2.77 \\ 12.6 & 14.2 & 6.28 \\ 2.77 & 4.73 & 7 \end{bmatrix}$$

$$\mathbf{CR}(\mathbf{1}) \qquad \mathbf{CR}(\mathbf{2}) \qquad \mathbf{CR}(\mathbf{3})$$

4.5 Samennemen van verschillende algoritmen

Er werden hiervoor een groot aantal algoritmen besproken. Eén van de mogelijkheden om de aanbevelingen nog te verbeteren, is om de beste algoritmen met specifieke parameterwaarden samen te nemen voor een bepaalde dataset en zo een nieuw aanbevelingssysteem te vormen. Een schoolvoorbeeld hiervan is de inzending van BellKor voor de Netflixprijs[3]. BellKor gebruikt 107 verschillende algoritmen die in een bepaalde mix samengenomen worden.

De vraag is natuurlijk hoe het samennemen van algoritmen het best gebeurt. In de literatuur is hier echter weinig over te vinden.

In [9] wordt een lineaire methode voorgesteld. De zekerheidswaarden van aanbevelingen gegeven door de verschillende algoritmen worden gecombineerd als:

$$\sum_a w_a r_i^a$$

[3]De Netflixprijs zoekt naar verbeteringen van de accuraatheid van de voorspellingen over hoeveel iemand van een film zal gaan houden gebaseerd op hun filmvoorkeuren. Wanneer een organisatie die voorspellingen substantieel kan verbeteren kunnen prijzen gewonnen worden. Meer informatie op: `http://www.netflixprize.com`

Hierbij is w_a het gewicht gegeven aan algoritme a en is r_i^a de zekerheidswaarde gegeven door algoritme a aan de aanbeveling van een bepaald item i. Het spreekt voor zich dat algoritmen met een gewicht $w_a = 0$ niet meetellen in de combinatie. De beste aanbevelingen worden dan gekozen uit de gerangschikte lijst om te tonen aan de gebruiker. De gewichten gebruikt bij het combineren van de individuele zekerheidswaarden (w_a) kunnen geleerd worden door een aantal voorgeselecteerde discrete parameterwaarden te onderzoeken en voor elke parameterwaarde de algoritmen te laten lopen met een evaluatiemethode.

Aangezien niet alle zekerheidswaarden bij een aanbeveling tussen 0 en 1 liggen, is een andere manier van werken per aanbevelingssysteem de waarden te normaliseren. Dit kan verwezenlijkt worden door alle waarden te delen door de grootste waarde van dat bepaald aanbevelingssysteem. Dit impliceert wel dat er evenveel aanbevelingen met zekerheidswaarde 1 zullen zijn als er aanbevelingsalgoritmen gecombineerd worden. Vervolgens wordt in een gegevensstructuur alle aangeraden items bijgehouden met de maximumwaarde waarmee ze ooit werden aangeraden. Stel dus dat een item door het ene aanbevelingsalgoritme wordt aangeraden met waarde 0.83 en vervolgens door een ander aanbevelingsalgoritme met 0.67, dan zal de waarde 0.83 bij dat aanbevelen item behouden blijven.

4.6 Verklaring van de aanbeveling

Een verklaring waarom een bepaald item aanbevolen wordt, zorgt voor transparantie naar de gebruiker toe van de werking van het Collaborative Filteringsysteem. Gebruikers zullen sneller aanbevelingen vertrouwen wanneer ze de reden weten waarom iets aanbevolen wordt. De verklaringen zullen de gebruikers helpen het CF-proces te verstaan en ook te begrijpen waar de sterkten en zwakten van het systeem liggen [19].

Verklaringen van de aanbevelingen zorgen voor een mechanisme om fouten af te handelen die voorkomen bij aanbevelingen. Veronderstel hoe mensen omgaan met suggesties verkregen door anderen. Er wordt erkend dat andere mensen imperfecte aanbevelers zijn. In het beslissingsproces of een aanbeveling aanvaard wordt van een vriend, wordt de kwaliteit van voorafgaande aanbevelingen van die vriend nagegaan of worden de hoofdzakelijke interesses vergeleken. Hoe dan ook, als er twijfel is, wordt snel de vraag "waarom?" gesteld en kan de vriend uitleggen waarom hij/zij dit aanraadt. Vervolgens kan de logica van de aanbeveling bekeken worden en zal deze overtuigend zijn als het bewijs sterk genoeg blijkt. Dezelfde redenering kan toegepast worden voor Collaborative Filteringsystemen.

Het is altijd mogelijk dat iemand bijvoorbeeld een ticket gekocht heeft op zijn/haar naam voor één of andere voorstelling. Met deze transactie wordt echter nog altijd reke-

ning gehouden in het CF-systeem. Wanneer de gebruiker vervolgens items aangeraden krijgt op basis van deze voorstelling, weet hij/zij dat die er niet toe doen.

Een verklaringsfaciliteit invoegen in het aanbevelingssysteem zorgt voor vele voordelen voor de gebruiker:

- **Rechtvaardiging.** De gebruiker begrijpt de redenering hoe de aanbeveling tot stand kwam, zodat deze kan beslissen hoeveel vertrouwen er in de aanbeveling kan geplaatst worden.
- **Gebruikersbetrokkenheid.** De gebruiker wordt betrokken in het aanbevelingsproces, waarbij zijn/haar kennis en besluitvaardigheidsvermogen gebruikt wordt om het beslissingsproces te vervolledigen.
- **Vorming.** Er is vorming van de gebruiker nodig voor de processen om de aanbevelingen te genereren, zodat de gebruiker beter kan verstaan wat de sterktes en limieten zijn van het systeem.
- **Aanvaarding.** Het aanbevelingssysteem wordt makkelijker aanvaard gezien de limieten en sterkten volledig zichtbaar zijn en de aanbevelingen gerechtvaardigd zijn.

Er zijn verschillende manieren hoe een aanbeveling verklaard kan worden. Er werden verschillende soorten onderzocht in [19] en daaruit kwamen volgende als het meest interessant:

- histogram met groepering per ratingswaarde,
- hoe correct het systeem in het verleden was voor dit item,
- histogram met de ratings van de buren,
- tabel van ratings van de buren,
- gelijkaardigheid met andere items.

Het is ook zo dat niet alle verklaringen een positief effect hebben, integendeel. Sommige zullen het suggereren zelfs tegenwerken.

Deel II

Methodologie

Hoofdstuk 5

Invoergegevens

Afhankelijk van de beschikbare datasets reageren de verschillende Collaborative Filteringalgoritmen anders. Daarom worden de gebruikte datasets hieronder beschreven samen met hun eigenschappen.

5.1 Datasets

5.1.1 Onderzoeksdatasets

Er zijn enkele datasets vrijgegeven voor onderzoeksdoeleinden. Verschillende van die datasets worden vaak gebruikt in wetenschappelijke artikels aangezien ze vrij beschikbaar zijn of waren op het internet.

- **Jester.** Jester is een dataset met 100 grappen die door een groot aantal gebruikers een beoordeling kregen. De beoordeling gebeurt op een beoordelingsbalk en zorgt voor ratings met reële waarden. Elke nieuwe gebruiker is verplicht om 40 grappen te beoordelen met een waarde tussen -10 en 10.
- **EachMovie.** De EachMovieservice maakte deel uit van een onderzoeksproject aan het Systems Research Center of Digital Equipment Corporation. Deze service was 18 maanden beschikbaar en werd gesloten in september 1997. Tijdens deze tijd groeide de database snel en bevatte uiteindelijk ratings van 72916 gebruikers op 1628 films. De ratings bevatten discrete waarden tussen nul en vijf (inclusief) [37].
- **MovieLens.** Dit is veruit de meest gebruikte dataset. Deze dataset werd samengesteld door het GroupLensonderzoeksproject en bestaat uit actuele ratings van een groep gebruikers. De kern van deze dataset is een lijst filmratings gegroepeerd

per persoon. Elk persoon heeft ten minste 20 films beoordeeld. Alle MovieLens-beoordelingen bestaan uit een rating gelegen tussen één en vijf (inclusief).

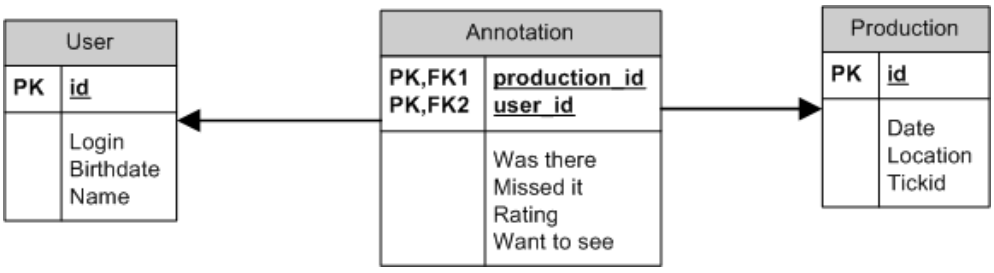
- **BookCrossing.** Deze dataset bevat heel wat boeken en beoordelingen van gebruikers op boeken. Deze dataset werd verzameld door Cai-Nicolas Ziegler in 4 weken tijd via de BookCrossingcommunity. Het bevat 278858 anonieme gebruikers met 1149780 ratings (expliciete en impliciete) over 271379 boeken. De ratings liggen tussen één en tien (inclusief).
- **Audioscrobbler.** Deze dataset bevat heel wat artiesten en liedjes samen met de mensen die ze beluisteren en bevat binaire ratings. Deze dataset is niet meer vrij verkrijgbaar.

In onderstaande experimenten wordt enkel gebruik gemaakt van de MovieLens- en de BookCrossingdatasets. De Jesterdataset is geen realistische dataset door de verhouding items op beoordelingen.

5.1.2 Vooruitdataset

De dataset van Vooruit wordt samengesteld uit twee databases. Enerzijds is er de database van de communitywebsite, waar gebruikers producties op een verlanglijstje kunnen zetten, kunnen aanduiden welke producties ze gezien hebben, ... Anderzijds is er de database van het ticketingsysteem.

Websitedatabase



Figuur 5.1: Vereenvoudigde voorstelling gegevens in websitedatabase

De website van Vooruit is in gebruik sinds april 2007. Doordat de website relatief recent is, bevat de *MySQL* database nog niet veel gegevens. De database bevat gegevens over gebruikers, producties en de relatie daartussen. Een sterk vereenvoudigde voorstelling van de database is te vinden in Figuur 5.1.

Tabel 5.1: Websitedatabase-evolutie

	10/08/2007	18/10/2007	12/02/2008	12/03/2008	9/04/2008
Kalender	1837	2700	3589	3926	4159
Gezien	1026	1238	1607	1676	1787
Gemist	276	351	507	556	615
Beoordelingen	201	279	455	506	554

De evolutie van de gegevens in de websitedatabase is te zien in Tabel 5.1. Het is duidelijk dat de community in volle ontwikkeling is: steeds meer mensen worden lid en nemen deel.

Ticketingdatabase

De ticketinggegevens worden bewaard in een Oracle $10g$ database. Deze database bevat verkopen sinds 2001. Een vereenvoudigde voorstelling van de database is te vinden in Figuur 5.2.

Figuur 5.2: Vereenvoudigde voorstelling gegevens in ticketingdatabase

Bruikbaarheid gegevens

Doordat er met twee databases gewerkt wordt, moeten gebruikers uit de ene database samengevoegd worden met gebruikers uit de andere. Hetzelfde moet gebeuren voor producties. Op die manier kan één profiel per gebruiker opgesteld worden met daarin alle gegevens. Helaas zijn er geen goede (automatische) verbindingen tussen de databases.

Samenvoegen van gebruikers De gegevens in de ticketingdatabase zijn niet goed onderhouden. De gegevens van een gebruiker zijn vaak onvolledig, incorrect of erger nog, gedupliceerd:

- Aantal gedupliceerde namen: 18306 (20% van de gebruikers in het ticketingsysteem).

- Aantal gebruikers met gedupliceerde naam waarbij ook de adresgegevens exact dezelfde zijn: 4926 (5%).
- Aantal gebruikers met onjuiste geboortedatum: dit is het aantal gebruikers zonder gekende geboortedatum samen met het aantal geboortedatums buiten het interval 1907-1998: 61613 (67%).

De recentere websitedatabase is heel wat netter. Doordat het veld e-mail verplicht is en tevens gecontroleerd wordt, heeft het altijd een correcte waarde. Van 23% van de gebruikersgegevens in de websitedatabase zijn enkel de gegevens gekend om aan te melden, dus geen voornaam, familienaam of geboortedatum.

Enkel de volgende gegevens kunnen dus gebruikt worden om gebruikers uit de verschillende databases met elkaar te linken:

- E-mailadres: maakt een goede koppeling mogelijk. Ongeveer 30% van de e-mailadressen die te vinden zijn in de websitedatabase zijn ook terug te vinden in de ticketingdatabase. Indien verschillende gebruikers hetzelfde e-mailadres gebruiken, bijvoorbeeld een getrouwd stel, worden ze toch samengevoegd in één profiel. Er wordt ervan uitgegaan dat ze geïnteresseerd zijn in dezelfde voorstellingen.
- Voor- en familienaam samen met geboortedatum. Doordat de geboortedatum niet vaak of correct is ingevuld in zowel de ticketing- als de websitedatabase, voldoen niet veel gebruikers aan deze voorwaarde. In de ticketingdatabase werden er 250 namen gevonden die ook in de websitedatabase van 12/02/2008 aanwezig zijn en waarbij het e-mailadres niet overeen komt. Het is dus redelijk zeker dat het om dezelfde personen gaat, maar bijkomende controle bleek niet mogelijk door de onvolledige gegevens. Om privacyredenen is een link op naam alleen niet gewenst.

Samenvoegen van producties. Het samenvoegen van producties is iets eenvoudiger. De gegevens van producties waarvoor op de website tickets verkocht worden, bevatten het veld `tickid`, dit is een verwijzing naar het sleutelveld in de ticketingdatabase. Bij de producties waarvoor geen `tickid` beschikbaar is, wordt getracht een link te leggen via de startdatum en de locatie of via de startdatum en de naam. Enkel de naam van een productie is niet genoeg omdat sommige producties jaarlijks en onder dezelfde naam georganiseerd worden.

Verschillende Vooruitdatasets. Aangezien de gegevens uit verschillende bronnen komen, is er geopteerd om de algoritmen te testen met drie datasets afkomstig van de Vooruitdata:

- **Volledige dataset.** Eerst en vooral is er de volledige dataset waarbij alle gegevens gebruikt worden die beschikbaar zijn. Niet alle algoritmen kunnen hierop uitgevoerd worden omdat de interactiematrix erg grote dimensies heeft.
- **Websitedataset.** Een andere mogelijkheid is enkel die producties en gebruikers uit de websitedatabase in de dataset te steken. Deze gegevens worden wel aangevuld met gegevens uit de ticketingdatabase. Als bijvoorbeeld een gebruiker op de website niet heeft vermeld dat hij/zij naar een concert is geweest, zal dit zo aangevuld worden en zal een vollediger profiel beschikbaar zijn.
- **Minimumdataset.** Ten slotte is het ook handig wanneer er een minimaal aantal ratings opgelegd wordt die een gebruiker moet hebben om aanbevelingen te krijgen. Hier wordt tien genomen als minimum aantal.

5.2 Eigenschappen

Elke dataset heeft enkele voor de hand liggende eigenschappen:

- m: het aantal gebruikers.
- n: het aantal producten of items.
- De dichtheid wordt in percent uitgedrukt en is gedefineerd als

$$dichtheid = 100 \times (\frac{\text{aantal beoordelingen}}{n \times m})$$

  Een interactiematrix ($\mathbf{R}$) is meestal erg ijl, een dichtheid van minder dan één procent komt vaak voor.
- De representatie van de beoordelingen. Wordt er gebruik gemaakt van discrete, continue of zelfs binaire waarden? Hier werd dieper op in gegaan in sectie 4.2.1.

Het is duidelijk dat elk van deze eigenschappen een sterke invloed heeft op de werking van een algoritme. Deze eigenschappen zijn makkelijk te bepalen en worden meestal vermeld in wetenschappelijke artikels. Andere eigenschappen zijn iets subtieler, maar de invloed ervan mag niet worden onderschat:

- De verdeling van de beoordelingen. Wordt de volledige beoordelingsschaal gebruikt door alle gebruikers of zijn er afwijkingen? Het normaliseren van beoordelingen kan op vele verschillende manieren gebeuren. Enkele werden reeds besproken in de literatuurstudie.

- De verdeling van het aantal beoordelingen per gebruiker. De dichtheid van een dataset kan een vertekend beeld geven indien er enkele gebruikers enorm veel beoordelen en veel gebruikers erg weinig.

Tabel 5.2: Datasetstatistieken

Dataset	m	n	ratings	sparsity (%)	ratings/gebruiker
MovieLens	6040	3706	1000209	4,468	165
BookCrossing	105283	340555	1149780	0,003	11
Volledige dataset	81208	2067	133050	0,079	2
Websitedataset	1857	917	9940	0.584	5
Minimumdataset	1996	1319	62027	2.356	31

In Tabel 5.2 wordt een overzicht gegeven van de verschillende eigenschappen voor de beschikbare datasets. Opvallend is het verschil in ijlheid of sparsity. De ijlheid bij MovieLens is heel groot, terwijl die bij BookCrossing heel klein is. Alle datasets van Kunstencentrum Vooruit liggen hiertussen. Toch is het aantal beoordelingen per gebruiker het kleinst bij Vooruit, behalve als er gekeken wordt naar de dataset waarbij iedereen minstens 10 ratings heeft. De volledige dataset is niet echt bruikbaar aangezien deze alle gebruikers bevat van de vroegere ticketingdatabase, terwijl er geen verkoopgegevens voor hen beschikbaar zijn.

De MovieLensdataset en de minimumdataset zijn ideale datasets omdat er heel wat ratings zijn per gebruiker en omdat de interactiematrix relatief dicht is. Deze datasets zijn echter niet realistisch en zullen bijna nooit voorkomen in de praktijk, behalve wanneer men mensen verplicht om items te beoordelen. Hierdoor wordt echter de zekerheid verloren dat de ratings naar behoren zijn gegeven en kan foute informatie in het systeem sluipen. Vele algoritmen worden toch getest met deze datasets, waardoor heel goede aanbevelingen gegenereerd worden. Toch moet men dit met een korreltje zout nemen. De BookCrossingdataset is daarentegen een realistische dataset. Wanneer een algoritme hierop goed reageert, zal dit het doorgaans ook doen op andere realistische datasets.

5.3 Audioscrobbler

Het Audioscrobblersysteem voedt hoofdzakelijk de website Last.fm (reeds kort besproken in Hoofdstuk 1), maar biedt daarnaast nog data aan via webservices zodat andere projecten interessante dingen kunnen verwezenlijken met de beschikbare data en aanbevelingen. Aangezien Kunstencentrum Vooruit het interessant vond om ook *cultuur* buiten het centrum aan te bieden, zal de webservice van Audioscrobbler[1] gebruikt wor-

[1]Meer info is te vinden op: `http://www.audioscrobbler.net/`.

den om per gebruiker en per voorstelling andere artiesten aan te raden. Dit is echter enkel mogelijk voor concerten, aangezien Audioscrobbler enkel werkt voor muzikanten.

Hoofdstuk 6

Implementatie

Het implementeren van Collaborative Filteringalgoritmen is uitdagend: het geheugengebruik moet binnen de perken gehouden worden, er moeten heel wat suggesties per seconde gegenereerd kunnen worden en de algoritmen moeten bovenal goede resultaten opleveren. Er werd gezocht naar open-source implementaties als basis. De volgende bibliotheken werden onderzocht:

- **COFE** is een Java softwarebibliotheek die beschikbaar is onder een licentie van de Oregon State University. De naam staat voor COllaborative Filtering Engine. Er is weinig documentatie over beschikbaar en het bevat enkel een implementatie voor een nearest-neighbour algoritme waarbij de similariteiten berekend worden met Pearson. De bibliotheek wordt ook al meer dan 4 jaar niet meer onderhouden.
- **ColFi** is geschreven in Java voor een masterproef [7]. De documentatie in de thesis is duidelijk maar ook deze softwarebibliotheek wordt niet meer actief ontwikkeld. ColFi bevat enkel nearest-neighbouralgoritmen, zowel item- als user-based.
- **Consensus** is geschreven in Python, een objectgeörienteerde en platformonafhankelijke scripttaal. De softwarebibliotheek is relatief goed gedocumenteerd en bevat nearest-neighbouralgoritmen met verschillende afstandsmetrieken. Er wordt nog actief gewerkt aan deze bibliotheek.
- **RACOFI** biedt ondersteuning voor *rule applying* Collaborative Filtering [3] en voor meerdimensionale waarderingen. Het bouwt verder op COFI, een Javabibliotheek uitgegeven onder GPL-voorwaarden. Er is geen documentatie voor beschikbaar.
- **Taste** is eveneens geschreven in Java en implementeert verschillende algoritmen en talloze afstandsmetrieken. Zowel item- als user-based nearest-neighbouralgoritmen als het Slope-One-algoritme zijn geïmplementeerd. Er wordt

actief aan ontwikkeld en de documentatie is goed. Het is vrijgegeven onder de voorwaarden verbonden aan de Apache License V2.0.

Er werden ook bibliotheken gevonden waarvan de broncode niet beschikbaar is of die geschreven zijn in een programmeertaal die niet geschikt leek. Zo is bijvoorbeeld SUGGEST een in C geschreven bibliotheek waarvan de bron niet beschikbaar is en is "Fast Maximum Margin Matrix Factorization" geschreven in Matlab en GNU Octave. Die bibliotheken werden meteen aan de kant geschoven omdat er moeilijk mee gewerkt kan worden. Taste bleek de beste keuze.

6.1 Taste

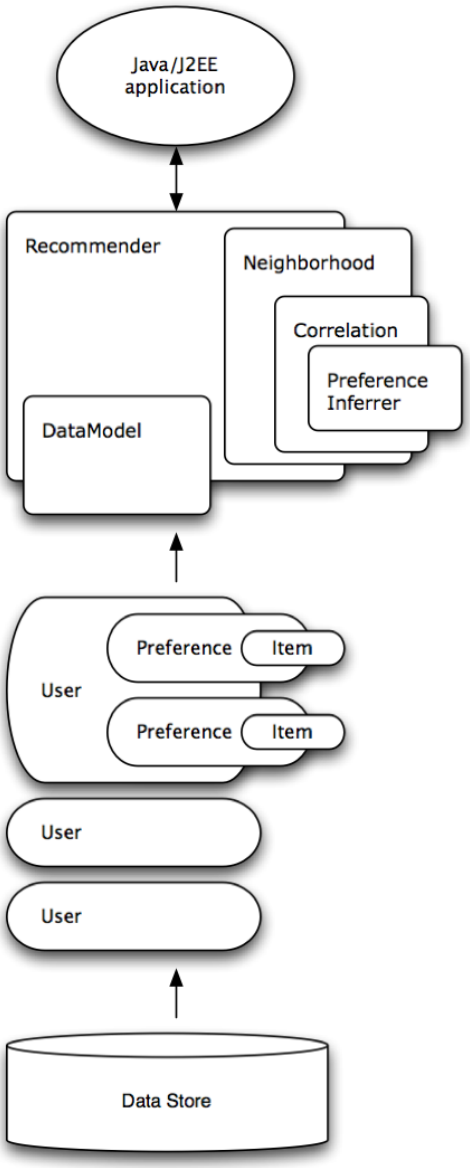
Taste werd gekozen als basis voor het werk omdat er goede documentatie beschikbaar is en omdat er al heel wat bruikbare componenten in aanwezig zijn. Ook wordt het nog steeds onderhouden. Er is dus al veel werk verzet, onder andere tijdens Google's Summer Of Code, en daarop kan verder gewerkt worden. Hieronder wordt de structuur van Taste besproken, daarna wordt dieper ingegaan op de implementatie van de algoritmen en andere componenten die werden toegevoegd.

6.1.1 Structuur

Taste voorziet interfaces waarmee een eigen aanbevelingssysteem gemaakt kan worden. De interfaces `DataModel`, `UserCorrelation`, `ItemCorrelation` en `Recommender` zijn abstracties voor componenten die veel in aanbevelingssystemen voorkomen. Subpackages van `comp.planetj.taste.impl` bevatten implementaties van deze interfaces die samen gebruikt kunnen worden. Figuur 6.1 toont de relaties tussen de verschillende Tastecomponenten voor een user-based systeem. De structuur van een item-based systeem is gelijkaardig.

`Recommender` definieert de interface van een aanbevelingsalgoritme. Aan de hand van een bepaald `DataModel` kan het aanbevelingen genereren. Twee belangrijke implementaties van de interface zijn `GenericUserBasedRecommender` en `GenericItemBasedRecommender` die gebruik maken van respectievelijk een `UserCorrelation` of een `ItemCorrelation`.

Het `DataModel` beschrijft de manier waarop informatie over voorkeuren van gebruikers opgevraagd kan worden. Een implementatie haalt de informatie van om het even welke bron: tekstbestanden, webservices, databases, ... Enkele `DataModel`implementaties



Figuur 6.1: Onderdelen van Taste

worden met Taste meegeleverd. `MySQLJDBCDataModel` kan gebruikt worden om voorkeuren via JDBC uit een MySQL-database te halen, terwijl `FileDatamodel` de informatie haalt uit tekstbestanden. Meestal is een eigen aangepaste implementatie van `DataModel` nodig.

Bij een `DataModel` horen `User`s, `Item`s en `Preference`s. Dit zijn abstracties die respectievelijk gebruikers, producten en voorkeuren voorstellen. Een zelf geschreven `DataModel` levert implementaties op van die interfaces afhankelijk van de context van het aanbevelingssysteem. Zo kan bijvoorbeeld een klant van Kunstencentrum Vooruit voorgesteld worden door `VooruitUser`, een implementatie van `User`. Een productie kan voorgesteld worden door `VooruitProduction`, een implementatie van `Item`.

`UserCorrelation` definieert in hoeverre gebruikers op elkaar lijken om vervolgens daarmee een `Neighbourhood` te construeren. `ItemCorrelation` bepaalt de similariteit tussen `Items` maar is voor de rest analoog. Voordat de correlaties berekend worden, kan het zijn dat er nog een transformatie nodig is van de ratings. Hiervoor dienen implementaties van `PreferenceTransform2`. Aanbevelingen worden gegenereerd door te zoeken naar een `Neighbourhood` van gelijkaardige gebruikers aan een actieve gebruiker.

6.1.2 Recommenderklassen

De belangrijkste klassen van Taste zijn de `Recommender`klassen. Daarom wordt hier even dieper op ingegaan om aan te tonen wat een `Recommender` juist moet kunnen, afhankelijk van het soort: user-based of item-based.

De basisvereiste is dat er items kunnen aangeraden worden via `recommend(...)`. De methode `refresh()` zorgt ervoor dat de `Recommender` zichzelf herbouwt. Wanneer gebruik gemaakt wordt van een `UserBasedRecommender` is het eveneens mogelijk om de meest gelijkaardige gebruikers op te vragen, terwijl bij `ItemBasedRecommender` de meest gelijkaardige producten kunnen opgevraagd worden. Een `ItemBasedRecommender` kan ook verklaren waarom iets aangeraden wordt. Zoals in hoofdstuk 4.6 vermeld, is dit een belangrijke eigenschap.

```
public interface Recommender {
   List<RecommendedItem> recommend(Object userID, int howMany)
      throws TasteException;
   DataModel getDataModel();
   void refresh();
}

public interface UserBasedRecommender extends Recommender {
   List<User> mostSimilarUsers(Object userID, int howMany)
      throws TasteException;
}

public interface ItemBasedRecommender extends Recommender {
   List<RecommendedItem> mostSimilarItems(Object itemID, int howMany)
      throws TasteException;
   List<RecommendedItem> recommendedBecause(Object userID, Object itemID,
      int howMany) throws TasteException;
}
```

6.1.3 Werking

Met de daarnet besproken componenten is het eenvoudig om een item-based algoritme te construeren:

```
01 DataModel model = new FileDataModel(new file(‘‘data.txt’’));
02 ItemCorrelation itemCorrelation =  new PearsonCorrelation(model);
03 Recommender recommender =
      new GenericItemBasedRecommender(model, itemCorrelation);
04 recommender.recommend(‘‘Mathias’’,10);
```

Op de eerste regel wordt een `DataModel` aangemaakt dat op de tweede regel meegegeven wordt aan `PearsonCorrelation`, een implementatie van interface `ItemCorrelation`. Daarna wordt een `Recommender` aangemaakt die van die correlatie gebruik maakt om te bepalen hoe sterk items op elkaar lijken. Op regel vier worden voor de gebruiker met identificatie “Mathias” tien aanbevelingen gevraagd.

Aan het gebruik van Taste zijn ook enkele nadelen verbonden. Zo zijn er bijvoorbeeld erg veel klassen gemarkeerd met `final`. Dit zorgt er voor dat er geen overerving van die klasse meer mogelijk is hoewel het net handig kan zijn om functionaliteit toe te voegen via overerving. Om dit probleem te omzeilen moest af en toe code worden gekopieerd.

De structuur van Taste is optimaal voor de aangeboden functionaliteit maar als er extra zaken toegevoegd moeten worden is het moeilijk om het in de structuur van Taste in te passen. Er moest bijvoorbeeld veel code worden aangepast om het gewenste gedrag van `SmartUser` te verkrijgen.

6.2 Toegevoegde componenten

Er werden enkele componenten toegevoegd aan Taste om het zowel bruikbaar te maken voor Kunstencentrum Vooruit als om nog meer algoritmen te kunnen gebruiken. Ook werden enkele klassen aangemaakt die het testen van willekeurige algoritmen eenvoudig maken. Daarover is meer te lezen in hoofdstuk 7.2. Het verkleinde klassendiagram kan gevonden worden in bijlage C.

6.2.1 Datamodel

Eerst en vooral werden enkele `DataModel`implementaties toegevoegd:

- `BookCrossingDataModel` bevat de gegevens van de BookCrossingdataset die hiervoor beschreven werd.
- `VooruitDataModel` bevat gegevens van Kunstencentrum Vooruit en heeft enkele varianten. Met `VooruitJDBCDataModel` wordt direct uit de databases gelezen en `VooruitFileDataModel`, die zijn gegevens haalt uit tekstbestanden, is beschikbaar voor testdoeleinden, aangezien dit veel sneller gaat. De beschikbare datasets bevatten dan ofwel:
  - alle gegevens,
  - alle gegevens zonder de gebruikers die geen voorkeuren hebben en zonder items waarvoor geen voorkeuren zijn,
  - alle gegevens van de websitedatabase aangevuld met gegevens uit de ticketingdatabase.
- Er is voor elk `DataModel` een versie beschikbaar die een limiet op het aantal items legt, ook voor het standaard aanwezige `GroupLensDataModel`. Zo kan bijvoorbeeld een `BookCrossingDataset` opgevraagd worden met exact 2531 willekeurige items. Dit is praktisch voor testdoeleinden: een algoritme werkt sneller met minder items.

Een `VooruitProduction` is een `Item` met heel wat extra eigenschappen: websiteidentificaties, ticketingidentificaties, ... Ook geeft het aan `isRecommendable` een implementatie aangezien enkel die `VooruitProduction`s mogen aangeraden worden die in de toekomst liggen.

Ook een bezoeker van Kunstencentrum Vooruit beschikt over meer gegevens dan enkel een id en daarom werd een `VooruitUser` aangemaakt. Deze erft over van het toegevoegde `SmartUser` die als voornaamste extra functionaliteit `isRecommendable` heeft. Het doel hiervan is dat de gebruiker kan bepalen of een bepaalde `VooruitProduction` aangeraden mag worden aan hem/haar. Een gebruiker moet steeds de mogelijkheid hebben om iets als een 'onaanvaardbare' suggestie te declareren. Deze `item`s mogen dan niet meer aangeraden worden. Hiervoor worden twee verschillende tabulijsten bijgehouden: `tabulistProductions` en `tabulistArtists`.

Een `VooruitRating` is de `Preference` van een `VooruitUser` voor een bepaalde `VooruitProduction`. Deze voorkeur kan uitgedrukt zijn in een numerieke rating of aan de hand van binaire gegevens die de gebruiker uitdrukkelijk heeft gegeven. Wanneer een gebruiker minder dan 5 ratings gaf, wordt ook rekening gehouden met de webpaginabezoeken. Dit wordt echter voor minder waarde meegerekend.

6.2.2 Algoritmen

Er werd geopteerd om niet specifiek voor de gegevens van Kunstencentrum Vooruit nieuwe algoritmen toe te voegen, maar om alle algoritmen generiek te maken. Zo kunnen de geïmplementeerde algoritmen bijvoorbeeld later gebruikt worden in een andere context, zonder hier iets te moeten aanpassen. Hierdoor is het ook mogelijk om alle algoritmen te testen met bijvoorbeeld de MovieLensdataset en te kijken of de uitkomsten gelijkaardig zijn aan die aangegeven in de artikels.

Om te kunnen werken met verschillende algoritmen, werd een `VooruitRecommender` aangemaakt. Deze `Recommender` zal op zijn beurt verschillende andere `Recommender`s instantiëren en hun aanbevelingen tonen. Figuur C.1 is een klassendiagram die duidelijk maakt hoe dit in zijn werk gaat.

User-based

In Taste zijn reeds een aantal implementaties beschikbaar om correlaties te berekenen, zoals `PearsonCorrelation` en `SpearmanCorrelation`. Hieraan werd een implementatie toegevoegd van vectorsimilariteit, namelijk `CosineCorrelation`.

Er werd niet echt diep ingegaan op user-based aanbevelen door de verschillende gekende problemen die reeds werden vermeld in hoofdstuk 4.2.6. Schaalbaarheid is een cruciaal punt aangezien de websitecommunity nog steeds groeit. Toch is het de bedoeling dat aanbevelingen direct getoond worden, zonder enige vertraging. Een oplossing hiervoor is werken met een online en offline component. Dit impliceert echter dat de aanbevelingen overdag niet meer kunnen veranderen, waardoor de gebruiker na het aanpassen van zijn profiel de dag erna pas zijn nieuwe suggesties te zien krijgt.

Item-based

In Taste isr een implementatie aanwezig van een generiek item-based algoritme. Er kan een `ItemCorrelation` aan meegegeven worden die bepaalt hoe de afstanden tussen items berekend worden. Hoe goed een item-based algoritme presteert, is grotendeels van die afstandsmaat afhankelijk. Er zit echter geen ondersteuning in deze implementatie om berekende similariteiten op schijf te bewaren, wat echter praktisch is voor testdoeleinden. Ook is het niet mogelijk om de gegevens vooraf te normaliseren. Daarnaast gaat de implementatie er van uit dat de similariteiten berekend worden op het moment dat een aanbeveling gedaan wordt, wat veel tijd in beslag neemt. Daarom werd ervoor gekozen om zelf een implementatie van `GenericItemBasedRecommender` te voorzien die deze eigenschappen wel heeft.

Er werden ook enkele `ItemCorrelation`implementaties toegevoegd die gebruikt kunnen worden met de `GenericItemBasedRecommender`, aangezien in Taste enkel een implementatie aanwezig is van Pearsoncorrelatie. Het popalgoritme werd niet geïmplementeerd aangezien dit sowieso geen meerwaarde zal brengen aan Taste. De volgende correlaties, waarover meer uitleg te vinden is in hoofdstuk 4.3.1, werden geïmplementeerd:

- `CosineCorrelation` is een implementatie om correlatie via vectorsimilariteit te berekenen.
- `AdjustedCosineCorrelation` implementeert aangepaste cosinussimilariteit en is een uitbreiding op het voorgaande algoritme.
- `ConditionalProbability` berekent de correlatie gebaseerd op voorwaardelijke kans.

Verder werd een implementatie van `PreferenceTransform2` toegevoegd om `InverseItemFrequency` te bekomen.

Andere benaderingen

Hieronder volgt een overzicht van de verschillende andere benaderingen besproken in hoofdstuk 4.4. Telkens wordt hierbij vermeld of dit algoritme werd geïmplementeerd of net niet en waarom.

- Eigentaste is een heel interessant algoritme, maar niet toepasbaar voor Kunstencentrum Vooruit. De grootste reden hiervoor is dat elke gebruiker dezelfde testset voorgeschoteld krijgt met vragen wat hij/zij ervan vindt. Dit is niet mogelijk voor Kunstencentrum Vooruit, aangezien er niet genoeg voorstellingen zijn die alle bezoekers kennen en er dus een externe bron zou moeten betrokken worden. Dit is echter niet de bedoeling.
- Het dimensionaliteitsreductiealgoritme is niet van toepassing voor Kunstencentrum Vooruit aangezien de interactiematrix veel te ijl is. Bij MovieLens levert dit algoritme echter goede resultaten op volgens [49].
- Slope one moet niet meer geïmplementeerd worden aangezien het reeds in het Tasteframework aanwezig is als `SlopeOneRecommender`.
- Het spreading-activationalgoritme werd niet geïmplementeerd omdat het link-analysisalgoritme betere resultaten oplevert [24].

- Het link-analysisalgoritme werd geïmplementeerd in `LinkAnalysisRecommender` omdat het volgens [24] veel beter werkt dan de andere gekende algoritmen. Het grote probleem met dit algoritme is omgaan met het nodige geheugen. Er worden enkele grote, dichte matrices gebruikt waarop bewerkingen uitgevoerd worden die performant moeten blijven.

Samennemen van algoritmen

Er werden ook twee klassen geschreven die intern meerdere andere `Recommender`s aanspreken om suggesties te doen. De twee manieren zijn beschreven in hoofdstuk 4.5 en zijn geïmplementeerd in `MixedWeightedRecommender` en `MixedNormalizedRecommender`.

6.2.3 Aanbevelingen via Audioscrobbler

Naast de `VooruitRecommender` is er ook nog de `LastFmRecommender`. Deze `Recommender` heeft als doel artiesten of groepen aan te raden aan de gebruikers. Er wordt gebruik gemaakt van de webservice van Audioscrobbler die per artiest een XML-bestand teruggeeft van gelijkaardige artiesten, telkens met een percentage van overeenkomst en andere nuttige gegevens. Deze `Recommender` werkt ook volgens het online- en offlineprincipe: alle gegevens worden bijvoorbeeld 's nachts opgehaald en de aanbevelingen zelf worden gegenereerd op het moment dat erom gevraagd wordt, waardoor er overdag ook nieuwe aanbevelingen mogelijk zijn.

Om aanbevelingen te genereren voor een gebruiker wordt er gekeken welke concerten de gebruiker beoordeelde. Vervolgens worden bij elke concerterende artiest alle gelijkaardige groepen of artiesten opgehaald en worden de beste overeenkomsten bijgehouden. Wordt een groep meerdere keren aangeraden bij een concerterende artiest, dan wordt het rekenkundig gemiddelde genomen van de mate van overeenkomst. De beste overeenkomstige groepen of artiesten worden vervolgens aangeraden aan de gebruiker.

Om dit alles mogelijk te maken zijn er nog een `LastFmArtist` en een `LastFmRecommendedItem` nodig, respectievelijk implementaties van `Item` en `RecommendedItem`.

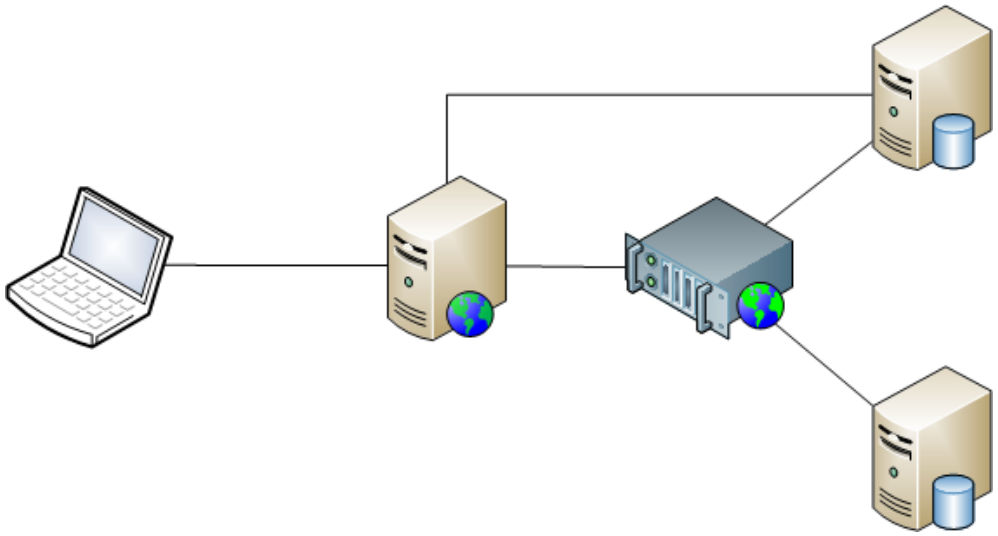
6.3 Webservice

Aangezien het programma op verschillende plaatsen moet kunnen gebruikt worden (bijvoorbeeld op de website en aan het bespreekbureau), werd geopteerd om een webservice

te schrijven. Deze webservice kan dan aangesproken worden waar nodig. De interface van de webservice biedt grotendeels dezelfde mogelijkheden aan zoals een `Recommender`, maar zal ook extra mogelijkheden aanbieden, zoals artiesten aanraden. Het grootste verschil met een `Recommender` is dat er kan opgegeven worden vanaf hoeveel ratings er suggesties mogen gegenereerd worden voor een bepaalde gebruiker. Meer informatie over deze webservice kan gevonden worden in bijlage A.2.

6.4 Gebruikte Technologieën

De verschillende gebruikte technologieën zullen hieronder niet in detail uitgelegd worden, enkel een korte beschrijving wordt gegeven.



Figuur 6.2: Verantwoordelijkheden servers

Er werd gebruik gemaakt van een hele reeks technologieën om een praktisch gedistribueerd aanbevelingssysteem te implementeren. De verantwoordelijkheden van de verschillende servers zijn opgesplitst zoals in Figuur 6.2.

De actieve gebruiker op de laptop links vraagt een pagina op van de website. De webserver bevraagt de webserviceserver die op zijn beurt gebruik maakt van de databases. De database met de gegevens van de website staat rechtsboven, de website zelf kan er mee communiceren, de ticketingdatabase staat rechtsonder. De technologieën die gebruikt worden op de verschillende servers zijn:

- **MySQL**: de websitedatabase is een MySQL 5.1 database. MySQL is een multithreaded, multi-user SQL DBMS. De broncode ervan is beschikbaar onder de

voorwaarden van de GNU General Public License en onder commerciële licenties. MySQL is eigendom van Sun Microsystems, het bedrijf achter Java. Meer informatie is te vinden op `http://www.mysql.com`.

- **Oracle**: de ticketingdatabase is een Oracle10*g* database. Het is een commercieel en robuust DBMS die met grote hoeveelheden gegevens performant blijft werken. Meer informatie is te vinden op `http://www.oracle.com`.

- **Ubuntu Server**: de webserviceserver gebruikt de laatste versie van Ubuntu Server, versie 8.04. Het is een linuxdistributie die gebaseerd is op Debian. Versie 8.04, ook gekend als "Hardy Heron", is een LTS-release wat staat voor Long Term Support. De serverediti van Hardy Heron wordt tot 2013 ondersteund. Meer info is te vinden op `http://ubuntu.com`.

- **Lighttpd** is een webserver die lichter, sneller en eenvoudiger te configureren is dan Apache. Het wordt gebruikt op de server links in Figuur 6.2. Lighttpd wordt gebruikt door onder ander YouTube en Wikipedia. Meer info is te vinden op `http://www.lighttpd.net`.

- **R** wordt gebruikt bij het evalueren van algoritmen. R is een open softwareomgeving waarmee statistiek bedreven kan worden en waarmee gafieken gemaakt kunnen worden. Het is bruikbaar op verschillende UNIX-platforms en op Windows [45]. Het wordt gebruikt om ROC-curves te tekenen en de oppervlakte onder die curve te bepalen. ROCR [55], een plug-in voor R, doet daarbij het zware werk. Daarover volgt uitleg in hoofdstuk 7.2. Meer info is te vinden op `http://www.r-project.org`.

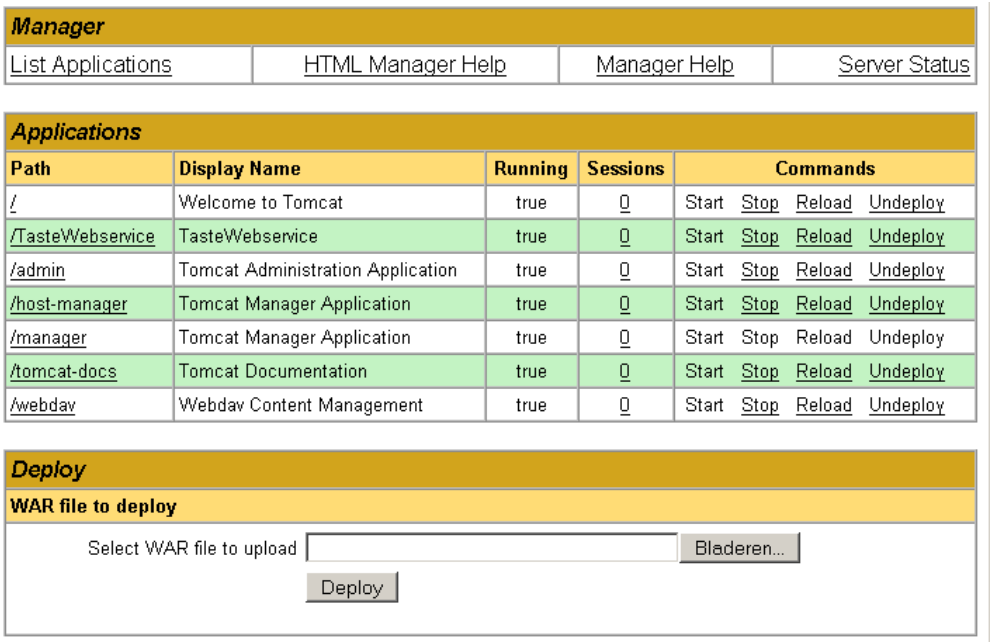
Naast de bovenstaande technologieën werd gebruik gemaakt van twee programmeertalen: Java en Ruby. Hieronder worden de verschillende gebruikte technologieën beknopt besproken die met die talen te maken hebben.

6.4.1 Java

De volgende technologieën die gebaseerd zijn op Java, een platformonafhankelijke, objectgeoriënteerde programmeertaal, werden gebruikt:

- **Taste** is zoals besproken een Collaborative Filteringbibliotheek voor Java. Sinds kort, 04/04/2008, is de volledige codebase van Taste geschonken aan Apache Mahout, een bibliotheek met verschillende schaalbare *Machine Learning*algoritmen. Meer informatie is te vinden op `http://lucene.apache.org/mahout`.

- **Apache Tomcat 5.5** is een webapplicatieplatform dat wordt gebruikt om de webservice op te hosten. Het kan werken met webapplicaties die gebruik maken van Java Servlets. Er worden samen met Tomcat webapplicaties meegeleverd die het beheer ervan eenvoudig maken. Zo kan met de Tomcat Manager eenvoudig een WAR-bestand[1] ontplooid worden op een server (Figuur 6.3). Meer informatie is te vinden op `http://tomcat.apache.org`.



Figuur 6.3: Tomcat Manager: installeren van WAR-bestand

- **Apache Axis2** is een webserviceplatform dat de herwerkte versie is van de veel gebruikte Apache Axis SOAP stack. Er zijn twee implementaties van Axis2, één in Java en één in C en is vergelijkbaar met JAX-WS. Hieronder volgen enkele voordelen aan Axis2 in vergelijking met de vorige versie:
  - Snelheid: Axis2 gebruikt een eigen objectmodel (AXIOM) en StAX (Streaming API for XML) om snel en efficiënt XML-documenten te parsen.
  - Geheugengebruik: Axis2 werd ontworpen om weinig geheugen te gebruiken.
  - WSDL-ondersteuning: Axis2 ondersteunt de Web Service Description Language, zowel versie 1.1 als 2.0. Dit zorgt ervoor dat het eenvoudiger wordt om proxyklassen te genereren in bijvoorbeeld Ruby.

  Een volledige lijst is te vinden op `http://ws.apache.org/axis2`.

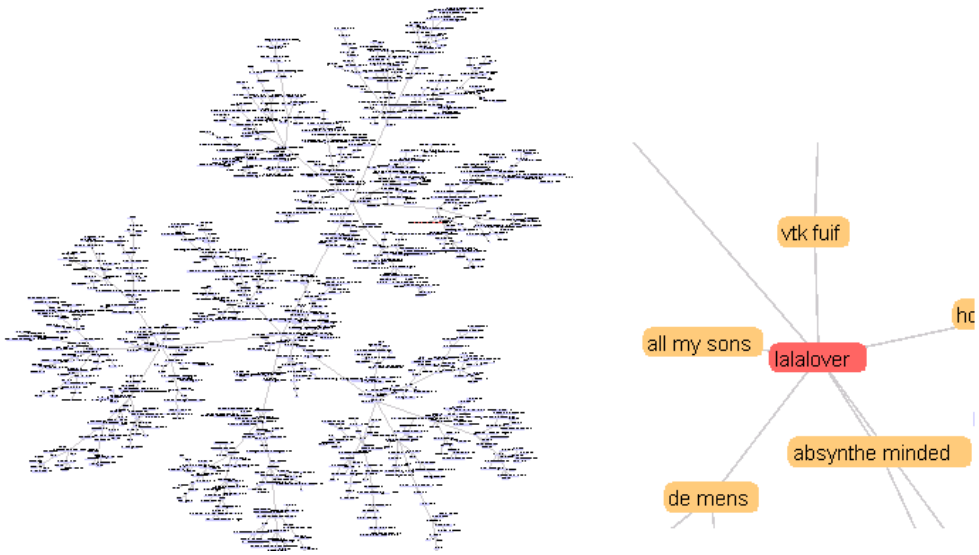
[1] Een WAR-Bestand bevat alle nodige bestanden nodig om een Java webapplicatie uit te voeren. Het is vergelijkbaar met een ZIP- of JAR-bestand.

- **JUnit** is een unit testing framework voor Java dat Test Driven Development (TDD) mogelijk maakt. Het werd onder andere gebruikt om algoritmen te controleren en om correlaties te testen. Meer informatie is te vinden op `http://junit.org`.

- **Prefuse** is een visualisatietoolkit voor Java. Het beschikt over een hele reeks mogelijkheden om graafachtige structuren grafisch voor te stellen. De omschrijving op hun website (`http://prefuse.org`) is duidelijk:

  > *Prefuse is an extensible software framework for helping software developers create interactive information visualization applications using the Java programming language. It can be used to build standalone applications, visual components embedded in larger applications, and web applets. Prefuse intends to greatly simplify the processes of representing and efficiently handing data, mapping data to visual representations (e.g., through spatial position, size, shape, color, etc), and interacting with the data[11].*

  Het werd gebruikt om clusters met producties te visualiseren zoals in Figuur **??**.



Figuur 6.4: Prefuse in werking: clusters van voorstellingen gevisualiseerd

6.4.2 Ruby

Ruby is een general-purpose, high-level programmeertaal. Ruby ondersteunt verschillende paradigma's, het haalt het merendeel van zijn mosterd bij objectgeoriënteerde programmeertalen, maar er zijn ook elementen aanwezig van functioneel- en metaprogrammeren. Ruby houdt automatisch zijn geheugen schoon en heeft een dynamisch

typesysteem, ook wel "duck typing"genoemd. Het is vergelijkbaar met onder meer Smalltalk, Python, Perl en Lisp. Ruby werd gebruikt in de Vooruitwebsite en voor allerhande scripts.

- **Ruby-on-Rails** is een webapplicatieframework dat ontwikkelen voor het web sneller, eenvoudiger en efficiënter tracht te maken. Het framework gebruikt het MVC-patroon om de applicatie te structureren. Het is opgebouwd uit verschillende componenten:

  - ActiveRecord is een Object-Relational-Mappper (ORM). Het wordt gebruikt om Rubyobjecten te bewaren in een database. Het grote voordeel ervan is dat nagenoeg alle SQL gegenereerd wordt, het is dan ook databaseonafhankelijk. Activerecord biedt onder meer ondersteuning voor MySQL, SQLLite, PostgreSQL en Oracle.
  - ActionController bevat ondersteunende functionaliteit om bijvoorbeeld om te gaan met sessies of caching.
  - ActionView wordt gebruikt om views te renderen. Dit zijn onder meer functies om eenvoudig webforms te creëren of om interactie via Ajax te realiseren.

  Meer info is te vinden op `http://www.rubyonrails.org/`.

- **Soap4R** wordt gebruikt om SOAP 1.1-berichten[2] te lezen in Ruby. De werking ervan is vergelijkbaar aan die in .NET of Java: er worden proxyklassen gegenereerd aan de hand van een WSDL-bestand. Die klassen kunnen dan gebruikt worden om eenvoudig te communiceren met een webservice. Meer info is te vinden op `http://dev.ctor.org/soap4r`.

- **Mongrel** is een performante HTTP-server voor Ruby. Het is bedoeld om webapplicaties in Ruby te hosten. Het maakt geen gebruik van FastCGI of CGI, zoals bijvoorbeeld Apache, maar enkel van HTTP. Het kan gebruikt worden in combinatie met Ruby on Rails of met andere Rubywebapplicatieframeworks. Meer info is te vinden op `http://mongrel.rubyforge.org/`.

- **Andere bibliotheken** worden gebruikt door de Vooruitwebsite om allerhande zaken eenvoudig te houden:

  - Hpricot: een HTML-parser.
  - RedCloth: ondersteuning voor een eenvoudige opmaaktaal.
  - Ferret: een bibliotheek die full-text search eenvoudig maakt.
  - RMagick: maakt het bewerken, verkleinen, converteren van figuren eenvoudig.

[2]Meer info over SOAP 1.1 is te vinden op `http://www.w3.org/TR/soap/`

Hoofdstuk 7

Resultaten

7.1 Metrieken

Het evalueren van een CF-algoritme is een kunst op zich. De literatuur staat vol met vele, onderling onvergelijkbare methoden om algoritmen te evalueren. Naast een hele reeks aan kwalitatieve evaluatietechnieken zijn er een paar dozijn kwantitatieve metrieken beschreven [20]. Bijna in elke paper over CF-algoritmen zijn meetresultaten te vinden die aantonen dat de voorgestelde benadering in die specifieke opstellingen, voor die specifieke metrieken superieur is tegenover de andere geteste algoritmen. Tot op heden zijn er nog geen richtlijnen die kunnen vertellen welk algoritme het best gebruikt kan worden in een bepaalde situatie [24].

Er zijn enkele elementen die het evalueren van CF-systemen moeilijk maken. Dit zijn de zaken die er voor zorgen dat een vergelijkende studie moeilijk is. In [20] wordt een mooi overzicht gegeven:

- **Dataset:** een algoritme kan beter of slechter zijn voor een bepaalde dataset. De eigenschappen van verschillende datasets: *sparsity*, aantal gebruikers, aantal items, ... kunnen sterk verschillen.
- **Doel:** het doel van de aanbeveling kan verschillen. Bij een beslissingsondersteunend systeem is het aantal foute beslissingen van belang. Bij andere systemen is het belangrijker om een rating te voorspellen: de "accuraatheid" van de voorspellingen.
- **Metingen:** welke metingen er uitgevoerd moeten worden in vergelijkende studies blijft tot op heden een open vraag.

Hoewel er veel discussie rond is zijn er toch enkele metrieken die veel terug keren in wetenschappelijke artikels. Hieronder volgt een reeks metrieken die in de praktijk goed

weergeven hoe een systeem zich gedraagd. De bruikbaarheid van elke metriek wordt besproken.

7.1.1 Gemiddelde absolute fout

De eerste metriek meet hoe goed een systeem de beoordeling van een gebruiker voorspelt. De meest gebruikte voorstelling is de gemiddelde absolute fout of *Mean Absolute Error*:

$$MAE = \frac{1}{k}\sum_{i=1}^{k}|p_i - q_i| \tag{7.1}$$

Voor elk beoordeling-voorspellingspaar (p_i, q_i) wordt de absolute fout berekend. Het totaal aantal paren k is gelijk aan het totaal aantal gegeven beoordelingen. Hoe lager de MAE, hoe accurater het schema. Er kan wel niet verwacht worden dat de MAE lager wordt dan de fout gemaakt via de beoordelingen van de individuen. Wanneer men namelijk dezelfde lijst producten voorlegt aan een persoon op verschillende tijdstippen, dan zal de resulterende data verschillen. De grootte van deze fout werd echter nog niet berekend [54].

Een voorbeeld: een interactiematrix $\mathbf{R}$, waarbij $\perp$ staat voor geen beoordeling gegeven, en een potentiëlescorematrix $\mathbf{P}$ zien er zo uit:

$$\mathbf{R} = \begin{bmatrix} \perp & 5 & \perp & 2 \\ \perp & 4 & 0 & 5 \\ 4 & \perp & 3 & \perp \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} 1 & 4 & 4 & 3 \\ 0 & 3 & 2 & 5 \\ 4 & 1 & 3 & 1 \end{bmatrix}$$

$$\begin{aligned} MAE &= \frac{|5-4| + |2-3| + |4-3| + |0-2| + |5-5| + |4-4| + |3-3|}{7} \\ &= 0.71 \end{aligned}$$

In dit voorbeeld ligt een beoordeling in het interval $[0, 5]$ en zijn er 7 beoordelingen. Er wordt een voorspelling gedaan voor elke beoordeling en het absolute verschil met de voorspelling wordt bepaald. De gemiddelde absolute fout is 0.71. Dit betekent dat een voorspelling gemiddeld 0.71 boven of onder de werkelijke beoordeling van een gebruiker ligt. Bij een gemiddelde absolute fout moet altijd verteld worden welke beoordelingsschaal (hier $[0, 5]$) gebruikt wordt. Als het resultaat genormaliseerd wordt naar percentages van de oorspronkelijke beoordelingsschaal dan is dit niet meer nodig. Deze metriek heet *Normalized Mean Absolute Error* of kortweg NMAE [15]. NMAE is

gelijk aan de oorspronkelijke MAE gedeeld door het verschil tussen het maximum en minimum van de beoordelingsschaal:

$$NMAE = \frac{MAE}{\text{maximum} - \text{minimum}} \tag{7.2}$$

De gemiddelde absolute fout heeft twee voordelen. Eerst en vooral is de metriek eenvoudig te berekenen. Daarnaast heeft het gekende statistische eigenschappen: of een verschil in gemiddelde absolute fout tussen twee systemen statistisch significant is, kan bewezen worden. Het grote nadeel is het feit dat het enkel de *juistheid* van een systeem meet. De juistheid van een voorspelling voor een item waar een gebruiker toch geen interesse in heeft is meestal niet belangrijk, maar dit wordt wel gemeten. Er zijn uitbreidingen die hier rekening mee houden en enkel de extremen meten [20]. Indien de dataset bestaat uit binaire voorkeuren (loggebaseerde CF) is de gemiddelde absolute fout geen correcte metriek. Hiervoor worden metrieken uit *Information Retrieval* gebruikt.

7.1.2 Metrieken uit Information Retrieval

Information retrieval is de wetenschap die zich bezig houdt met het zoeken van informatie in documenten, naar documenten zelf, in databases, ... De meest gekende toepassing ervan zijn zoekmachines zoals Google. In dit onderzoeksdomein worden enkele metrieken gebruikt die aanduiden hoe goed een systeem relevante documenten van niet relevante weet te onderscheiden. Deze zijn ook bruikbaar voor aanbevelingssystemen. De juistheid van een voorspelling wordt niet meer direct gemeten. Afwijkingen worden toegelaten zolang ze niet voor een foute classificatie van items zorgen. Er wordt gemeten hoe goed het systeem is in het classificeren van items in een groep relevante en niet relevante voor een actieve gebruiker. We bespreken de metrieken *precision* en *recall* daarna wordt de *ROC*-curve besproken die uit die twee metrieken volgt.

Precision en recall

Precision en recall worden berekend uit de Tabel 7.1. Alle voorkeuren van een gebruiker worden, als dit nog niet zo was, omgezet in een binaire schaal: ofwel is een item relevant ofwel niet. Om precision en recall te kunnen berekenen moeten de items verdeeld worden in een reeks aanbevolen en een reeks niet aanbevolen items. Er wordt verondersteld dat de gebruiker de volledige set aanbevolen items bekijkt.

Precision is de verhouding tussen het aantal relevante aanbevolen items en het totale aantal aanbevolen items. Het stelt de waarschijnlijkheid voor dat een aanbevolen item

	Aanbevolen	Niet aanbevolen
Relevant	True Positive (T_p)	False Negative (F_n)
Niet Relevant	False Positive (F_p)	True Negative (T_n)

Tabel 7.1: Mogelijke classificaties van aanbevelingen

relevant is. Recall is de waarschijnlijkheid dat een relevant item aanbevolen zal worden [58, 6].

$$precision = \frac{T_p}{T_p + F_p} \tag{7.3}$$

$$recall = \frac{T_p}{T_p + F_n} \tag{7.4}$$

De definitie van de term relevantie en de manier waarop relevantie berekend wordt is een bron van discussie. Bij het zoeken naar documenten kan een objectieve relevantie gedefinieerd worden. Bij aanbevelingssystemen is dit moeilijker. De relevantie is dan een maat voor de voorkeur van de actieve gebruiker voor een bepaald item en is, per definitie, subjectief [20]. Enkel de actieve gebruiker zelf kan bepalen of een item relevant is of net niet.

Recall en precision zijn allebei afhankelijk van het aantal aanbevolen items. Als het aantal aanbevolen items groter wordt, stijgt de recall en daalt de precision. Net omdat ze omgekeerd evenredig zijn moeten beide metrieken samen bekeken worden. Enkel een precision-en-recallpaar zegt iets over een systeem. Er zijn pogingen gedaan om precision en recall te combineren in één metriek. De voor aanbevelingssystemen meest gebruikte [51, 24, 6] is de F_1-maat:

$$F_1 = \frac{2 \times precision \times recall}{precision + recall} \tag{7.5}$$

De F_1-maat is dus een gewogen combinatie van de precision en recall die uitkomsten genereert tussen nul en één. Het geeft net als precision en recall enkel weer hoe goed een systeem is in classificeren. De kwaliteit van de volgorde van de relevante aanbevelingen wordt niet gemeten.

In het volgende voorbeeld wordt de interactiematrix $\mathbf{R}$ omgezet in een binaire interactiematrix $\mathbf{R}'$. Elke beoordeling boven 2.5 is relevant, een lagere beoordeling of een ontbrekende beoordeling ($\perp$) wordt als niet relevant beschouwd.

$$\mathbf{R} = \begin{bmatrix} \perp & 5 & \perp & 2 \\ \perp & \mathbf{4} & \mathbf{0} & \mathbf{5} \\ 4 & \perp & 3 & \perp \end{bmatrix} \quad \mathbf{R}' = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 1 & 0 & 1 & 0 \end{bmatrix} \tag{7.6}$$

Een systeem beveelt item 1, item 2 en item 3 aan voor gebruiker 2. De verschillende categorieën worden geteld:

- Item 1 wordt, net als item 3, aanbevolen en is niet relevant, dit is een *False Positive*.
- Item 2 wordt aanbevolen en is relevant, dit is een *True Positive*.
- Item 4 wordt niet aanbevolen en is relevant, dit is een *False Negative*.
- Alle niet relevante items worden aanbevolen, dus er is geen enkel dat niet aanbevolen wordt: geen enkele *True Negative*.

	Aanbevolen	Niet aanbevolen
Relevant	1 (T_p)	1 (F_n)
Niet Relevant	2 (F_p)	0 (T_n)

Tabel 7.2: Voorbeeld classificaties van aanbevelingen

In Tabel 7.2 worden deze gegevens voorgesteld. Uit deze tabel kunnen de precision, recall en F_1-maat berekend worden.

$$precision = \frac{T_p}{T_p + F_p} = \frac{1}{1+2} = 1/3$$

$$recall = \frac{T_p}{T_p + F_n} = \frac{1}{1+1} = 1/2$$

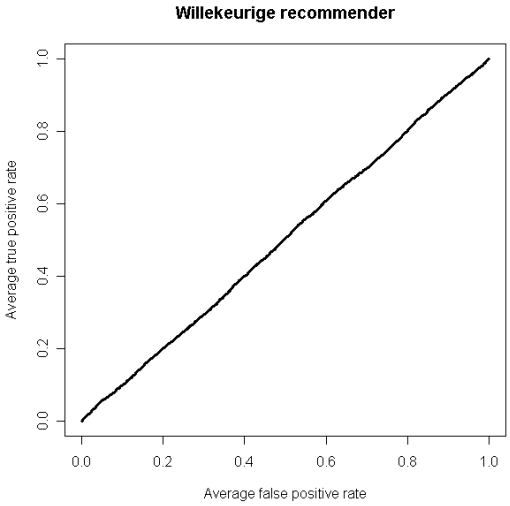
$$F_1 = \frac{2 \times precision \times recall}{precision + recall} = \frac{2 \times 1/3 \times 1/2}{1/3 + 1/2} = \frac{2/6}{5/6} = 2/5$$

ROC-curve

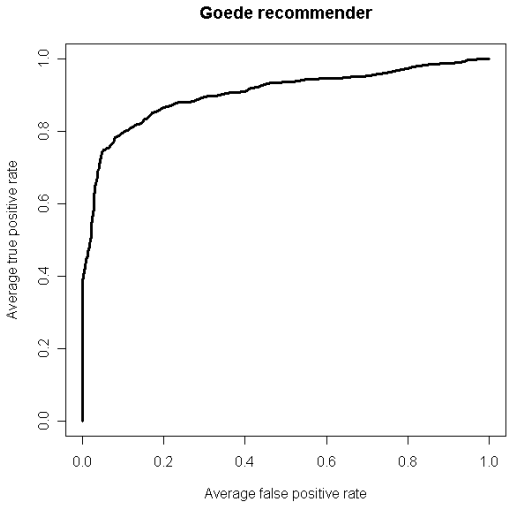
Een Receiver Operating Characteristic curve of ROC-curve kan het gedrag van een aanbevelingssysteem grafisch voorstellen. Er is onenigheid over de afkorting ROC, sommige bronnen hebben het over Relative Operating Characteristic. Wat wel zeker is, is dat het oorspronkelijk gebruikt werd in signaaltheorie en daarna via Information Retrieval bij het Recommender Systemsonderzoeksdomein terecht kwam [12]. Het is de bedoeling van een ROC-curve om te bepalen hoe goed een systeem het onderscheid kan maken tussen signaal (relevante items) en ruis (niet relevante items).

De ROC-curve beeldt sensitiviteit en specificiteit af van de test. Sensitiviteit is de kans dat een random geselecteerd relevant item wordt geaccepteerd door de filter. De specificiteit is de kans dat een random geselecteerd niet relevant item wordt afgewezen door de filter. De ROC-curve beeldt sensitiviteit (van 0 to 1) en 1-specificiteit (van 0 to 1) af, waardoor een set punten bekomen worden door de drempel van de voorspelde score te variëren waarboven het item aanvaard wordt [18].

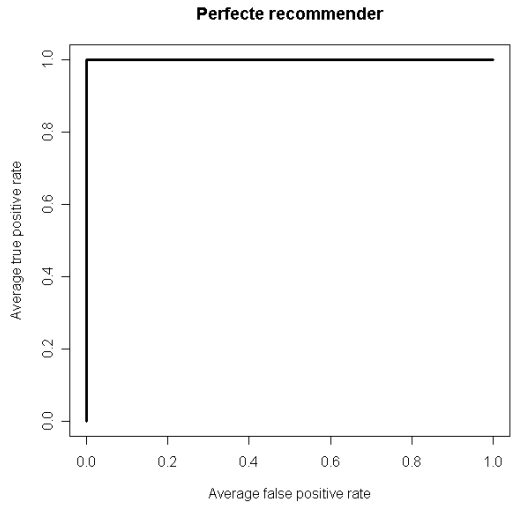
Net zoals bij precision en recall worden binaire relevanties gebruikt. Een item is relevant of is het niet. De volgorde van de aanbevelingen wordt buiten beschouwing gelaten.



Figuur 7.1: Willekeurige ROC, AUC=0,5



Figuur 7.2: Goede ROC, AUC=0,8



Figuur 7.3: Perfecte ROC, AUC=1

Een aanbevelingssysteem dat items willekeurig aanbeveelt, zorgt voor een ROC-curve met daarop enkel de rechte $x = y$, zoals in Figuur 7.1. De oppervlakte onder de curve is 0,5. Een betere ROC-curve is te zien in Figuur 7.2, er worden duidelijk meer relevante items aanbevolen dan niet relevante. Bij een perfect systeem worden eerst alle relevante

items aanbevolen en daarna volgen alle niet relevante. De oppervlakte onder de curve is één (Figuur 7.3).

Het is duidelijk dat de oppervlakte onder de ROC-curve (AUC) een goede maat is voor de performantie van een systeem. De oppervlakte stelt de waarschijnlijkheid voor waarmee een systeem een goede keuze kan maken tussen twee items, een item uit de set niet relevante items en een item uit de set relevante items. Men moet zich echter niet blindstaren op enkel maar de AUC: het geeft een maat voor de performantie samengenomen over hele systeem, het geeft niet noodzakelijk weer hoe goed het systeem werkt in een praktische omgeving. Het moet samen bekeken worden met andere metrieken die wel meten hoe praktisch het systeem is, bijvoorbeeld de rank score.

7.1.3 Rank score

Als het Collaborative Filteringalgoritme een lijst met aanbevelingen genereert, is het handig om te meten hoe bruikbaar die lijst is. De bruikbaarheid van een lijst voor een gebruiker is de waarschijnlijkheid dat een item in de lijst bekeken wordt vermenigvuldigd met de relevantie van het item. Bij binaire classificatie is een item ofwel relevant of niet. Deze metriek werd voorgesteld in [6] en wordt tegenwoordig regelmatig gebruikt [22, 20, 10, 24].

Er moet ook een schatting gemaakt worden hoe waarschijnlijk het is dat een gebruiker een item zal bekijken. We veronderstellen dat de waarschijnlijkheid dat een aanbeveling op de lijst bekeken wordt exponentieel daalt. De rank score voor gebruiker p kan als volgt worden bepaald:

$$RS_p = \sum_{j=1}^{N} \frac{q_j}{2^{(j-1)/(\alpha-1)}} \tag{7.7}$$

waarin

- N het aantal gegenereerde aanbevelingen,
- j de plaats is op de lijst, $j \in [1..N]$
- α is de positie op de lijst waarbij de kans dat het item bekeken wordt 0.5 is. Als $N = 10$ is 5 een redelijke keuze [6].
- $$q_j = \begin{cases} 1 & \text{als het product op plaats } j \text{ relevant is voor } p \\ 0 & \text{anders} \end{cases}$$

De verschillende rank scores worden voor elke gebruiker berekend en nadien op een speciale manier bij elkaar opgeteld:

$$RS = 100 \frac{\sum_p RS_p}{\sum_p RS_p^{max}} \tag{7.8}$$

Waarin RS_p^{max} de maximaal haalbare rank score is voor een bepaalde persoon: een lijst met alle toekomstige aankopen. Vergelijking 7.8 zorgt ervoor dat de uiteindelijke rank score onafhankelijk is van N en van de grootte van de testset.

7.2 Experimentele opstelling

De methode die gebruikt werd om de performantie van de algoritmen te meten, splitst de dataset op in twee delen. Het ene deel bevat 80% van de informatie en wordt gebruikt als invoer voor een algoritme. De overige 20% wordt gebruikt om de aanbevelingen van het systeem te controleren en verschillende metrieken te berekenen. De verhouding tussen controle- en testset kan ingesteld worden. De testprocedure ziet er zo uit:

```
   Input: algoritme,dataset
   Output: metrieken
 1 begin
 2     testdataset,controledataset = verdeel(dataset)
 3     algoritme.initialiseer(testdataset)
 4     foreach gebruiker in testdataset do
 5         aanbevelingen = algoritme.geefAanbevelingen(gebruiker)
 6         foreach aanbeveling in aanbevelingen do
 7             if aanbeveling ∈ controledataset then
 8                 berekenMetriekenVoorTruePositive()
 9             else
10                 berekenMetriekenVoorFalsePositive()
11             end
12         end
13     end
14     return berekendeMetrieken();
15 end
```

Algoritme 1: evaluatiealgoritme

Het algoritme schrijft automatisch R-code waarmee ROC-grafieken gemaakt kunnen worden en waarmee de oppervlakte onder de kromme bepaald kan worden. Alle me-

trieken die hierboven beschreven staan, behalve de gemiddelde absolute fout, worden berekend. Bij het teruggeven van de berekende metrieken, lijn 10, worden de gemiddelden bepaald over alle gebruikers. In de praktijk worden alle resultaten samengevoegd in één Excel-document via een Rubyscript om de interpretatie ervan eenvoudig te maken.

7.3 Interpretatie resultaten

Er werden experimenten gedaan op individuele algoritmen en op combinaties van de beste algoritmen. Hieronder volgt een beknopt verslag van de resultaten.

7.3.1 Individuele algoritmen

Per algoritme kunnen er één of meerdere parameters getest worden. Dit gaat van modelgrootte tot de parameter gamma bij link-analysis. Ook zijn er verschillende datasets waarop het algoritme losgelaten kan worden. Deze werden besproken in hoofdstuk 5.1. De verschillende parameters gecombineerd met de datasets zorgen per algoritme voor een grote hoeveelheid aan meetresultaten. Hier wordt dan ook een overzicht gegeven van de beste prestaties van de verschillende algoritmen. De BookCrossing- en GroupLensdataset werden gebruikt ter vergelijking.

Het opsplitsen van de dataset in een test- en controledeel gebeurt willekeurig waarbij gemiddeld 80% van de gegevens gebruikt wordt als invoer voor het algoritme en 20% als controle. Doordat de selectie willekeurig gebeurt moeten alle evaluaties meermaals herhaald worden om toevallige resultaten uit te sluiten. In Tabel 7.3 zijn de gemiddelden te vinden over enkele tests. Er werd vastgesteld dat de resultaten van de verschillende tests dicht bij elkaar liggen. Het doet er blijkbaar niet toe welke 20% van de gegevens weggelaten worden. Link-analysis werd niet uitgevoerd voor het volledige Vooruitdatamodel, aangezien het geheugen van de testpc veel te klein was.

Bij het evalueren is het belangrijk rekening te houden met alle evaluatiemetrieken. Het uiteindelijke doel is ze samen groot krijgen in plaats van individueel. De beste resultaten werden telkens in vet gezet. Uit de resultaten blijkt dat alle item-based algoritmen over het algemeen goed presteren met voorwaardelijke kans als beste similariteitsmaat. Link-analysis is een goede tweede.

Er werd gekozen om enkel verder te werken met de vooruitwebsitedataset omdat dan voor alle leden van de Vooruitcommunity aanbevelingen gegenereerd kunnen worden, niet enkel voor de leden met meer dan 5 voorkeuren. Als dit nodig blijkt te zijn, kan deze ondergrens toch nog opgelegd worden via de Vooruitwebservice.

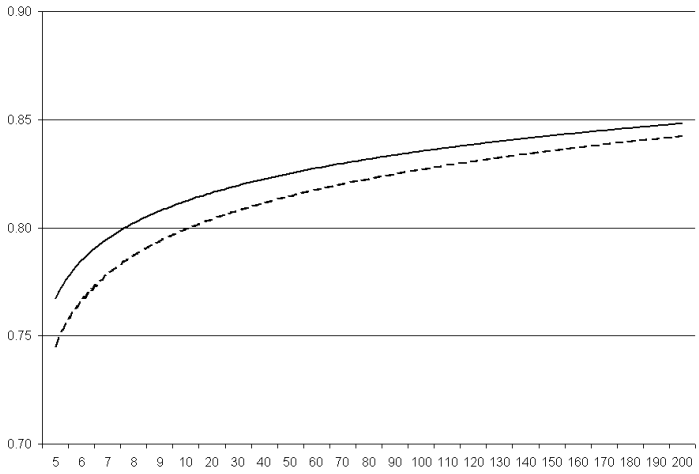
Dataset	Algoritme	F1	Rank score	AUC
BookCrossing	AdjustedCosine	0.0128	9.67	**0.64**
BookCrossing	Conditional: $\alpha = 0.3$	**0.0296**	**21.39**	0.59
BookCrossing	Cosine: inverse item	0.0051	5.63	0.62
BookCrossing	Link-analysis: $\gamma = 0.7$	0.0151	4.20	0.60
GroupLens	AdjustedCosine	0.0451	13.88	0.65
GroupLens	Conditional: $\alpha = 0.1$	**0.1624**	**63.25**	0.69
GroupLens	Cosine	0.0303	9.71	0.66
GroupLens	Link-analysis: $\gamma = 0.2$	0.1573	24.80	**0.75**
Vooruit reduced 5	AdjustedCosine	0.0280	9.40	**0.65**
Vooruit reduced 5	Conditional: $\alpha = 0.2$	**0.0788**	**23.60**	0.60
Vooruit reduced 5	Cosine: inverse item	0.0220	12.94	0.59
Vooruit reduced 5	Link-analysis: $\gamma = 0.1$	**0.0788**	15.86	0.56
Vooruit website	AdjustedCosine	0.0130	8.14	**0.81**
Vooruit website	Conditional $\alpha = 0.2$	**0.0446**	**29.80**	0.78
Vooruit website	Cosine: inverse item	0.0129	8.09	0.79
Vooruit website	Link-analysis $\gamma = 0.7$	0.0069	6.71	0.77
Vooruit	AdjustedCosine	0.0120	12.4	**0.62**
Vooruit	Conditional $\alpha = 0.2$	**0.0210**	**17.5**	0.59
Vooruit	Cosine: inverse item	0.0203	6.2	0.57

Tabel 7.3: Resultaten van individuele algoritmen

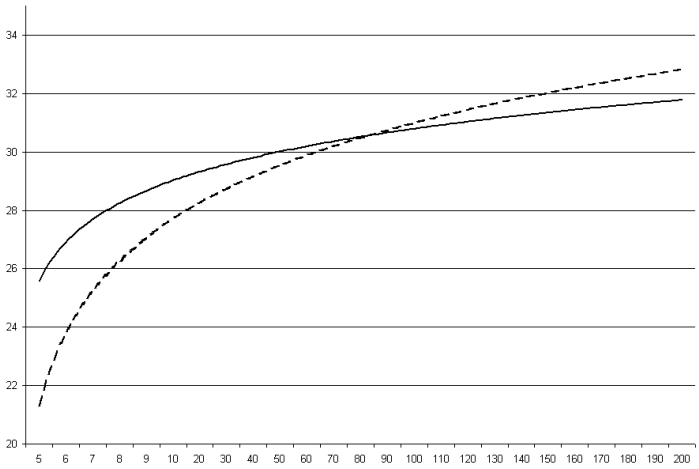
De beste resultaten voor voorwaardelijke kans bij de Vooruitwebsitedataset werden bekomen voor een $\alpha = 0.1$ en $\alpha = 0.2$. Hiervoor werd vervolgens gezocht naar een optimale modelgrootte. Om dit gemakkelijk te vinden werden enkele grafieken (figuren 7.4, 7.5 en 7.6) opgesteld. De grafieken van de andere algoritmen geven vergelijkbare resultaten.

In de grafieken wordt de invloed van de modelgrootte op de verschillende metrieken afgebeeld. De volle lijn stelt voorwaardelijke kans met $\alpha = 0.1$ voor, de stippellijn $\alpha = 0.2$. Voor alle metrieken presteert het algoritme, zoals verwacht, beter als de modelgrootte toeneemt. Het is dan ook aangeraden om de modelgrootte te maximaliseren, totdat er geen significante verbeteringen meer merkbaar zijn, dit aangezien de hoeveelheid aanwezig geheugen een beperkende factor is. Zoals te zien is in de grafieken doet voorwaardelijke kans het beter met $\alpha = 0.2$ dan met $\alpha = 0.1$ bij toenemende modelgrootte, AUC is hierop een uitzondering.

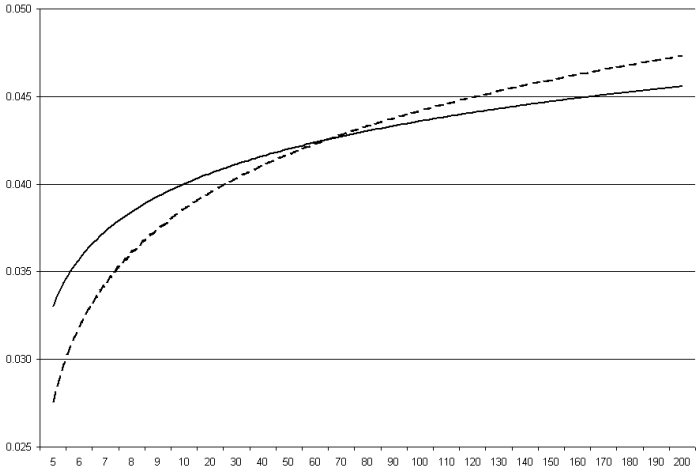
Figuur 7.7 bevat twee ROC-grafieken. De volle lijn stelt opnieuw $\alpha = 0.1$ voor en de stippellijn $\alpha = 0.2$, de modelgrootte is 190. De grafiek geeft weer welke invloed α heeft op een ROC-grafiek. Er kan vastgesteld worden dat, net zoals in Figuur 7.4, de AUC-



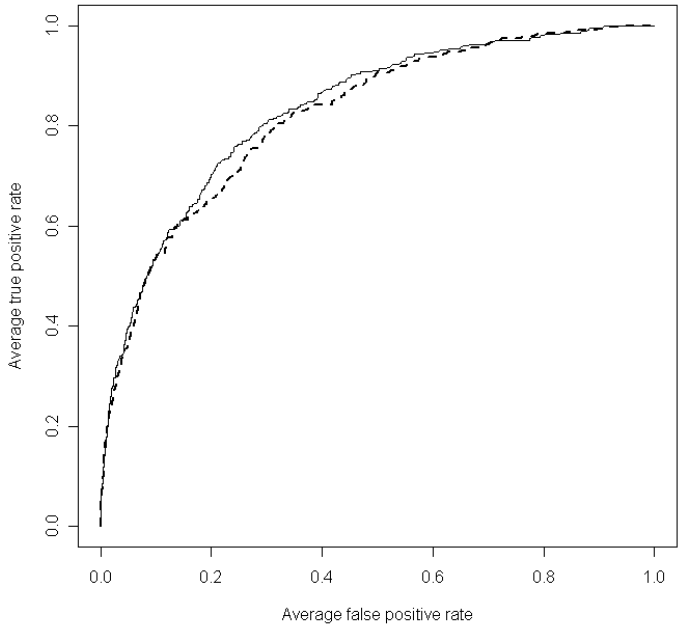
Figuur 7.4: Invloed modelsize op AUC



Figuur 7.5: Invloed modelsize op rank score



Figuur 7.6: Invloed modelsize op F1



Figuur 7.7: ROC voor voorwaardelijke kans met modelsize = 190 en $\alpha = 0.1$ en $\alpha = 0.2$ (streepjeslijn)

metriek voor $\alpha = 0.1$ beter is dan voor $\alpha = 0.2$, maar de andere metrieken verkiezen $\alpha = 0.2$. Er werd dus als beste individueel algoritme gekozen voor voorwaardelijke kans met $\alpha = 0.2$ met een modelgrootte van 190.

7.3.2 Samengenomen algoritmen

Met het samennemen van algoritmen werd gepoogd om de uitkomst van de evaluatiemetrieken nog te verbeteren voor de Vooruitwebsitedataset. Van elk algoritme werden die met de beste parameters gekozen om vervolgens te combineren met de andere betere. Ook hier werd dezelfde evaluatiestrategie toegepast. Er werden twee manieren getest om algoritmen samen te nemen, zoals beschreven in hoofdstuk 4.5.

Bij beide recommenders blijkt dat de beste algoritmen die zijn die louter bestaan uit voorwaardelijke kansen. Daarom wordt in tabellen 7.4 en 7.6 het algoritme niet expliciet vermeld, wel de specifieke eigenschappen.

Uit de volgende tabellen blijkt dat het eerste algoritme met de gewogen recommender gemiddeld de beste resultaten oplevert. Hetzelfde kan gezegd worden van het laatste algoritme bij de genormaliseerde recommender. Toch blijkt dat de gewogen recommender, voorgesteld door Google, er het best uit komt. Vooral bij rank score is een verschil waarneembaar.

Algoritme	F1	Rank score	AUC
2 × (alfa=0.1 size=190) + 5 × (alfa=0.2 size=190)	**0.0448**	**31.68**	0.84
1 × (alfa=0.1 size=190) + 1 × (alfa=0.2 size=190)	0.0404	27.76	0.84
4 × (alfa=0.1 size=190) + 5 × (alfa=0.2 size=140)	0.0412	27.41	**0.85**
2 × (alfa=0.1 size=70) + 4 × (alfa=0.2 size=190)	0.0411	26.87	**0.85**

Tabel 7.4: Resultaten van samengenomen algoritmen met gewogen recommender

Algoritme	F1	Rank score	AUC
(alfa=0.2 size=190) + (alfa=0.3 size=130)	0.0428	28.21	0.83
(alfa=0.1 size=70) + (alfa=0.2 size=140)	0.0429	28.30	**0.84**
(alfa=0.1 size=190) + (alfa=0.2 size=190)	**0.0433**	**28.45**	**0.84**

Tabel 7.5: Resultaten van samengenomen algoritmen met genormaliseerde recommender

7.3.3 Performantie

De evaluatie toont dat het beste gemengde algoritme beter is dan het beste individuele algoritme. Toch is het belangrijk op te merken dat gemengde algoritmen meer rekenwerk bezorgen. Daarom werd een stresstest losgelaten op deze algoritmen om te zien of het voordeel van het gemengde algoritme dan nog steeds gehandhaafd blijft.

Algoritme	Test	Tijd	Aanbevelingen/seconde
Individueel	1000 aanbevelingen zonder refresh	78.01	12.82
Individueel	1000 gelijkaardige zonder refresh	0.013	76923
Individueel	1000 aanbevelingen met refresh	80.87	12.37
Individueel	1000 gelijkaardige met refresh	0.013	76923
Gemengd	1000 aanbevelingen zonder refresh	150.64	6.638
Gemengd	1000 gelijkaardige zonder refresh	0.027	37037
Gemengd	1000 aanbevelingen met refresh	155.707	6.422
Gemengd	1000 gelijkaardige met refresh	0.031	32258
Last.fm	1000 aanbevelingen zonder refresh	0.557	1795
Last.fm	1000 gelijkaardige zonder refresh	0.024	41666
Last.fm	1000 aanbevelingen met refresh	0.559	1788
Last.fm	1000 gelijkaardige met refresh	0.024	41666

Tabel 7.6: Resultaten stresstest op algoritmen

Het is opvallend dat de gemengde recommender dubbel zoveel tijd nodig heeft als de andere recommender. Een refresh, dit is het herbouwen van het model, heeft echter

geen sterke invloed op het aanbevelen zelf. De Lastfmrecommender werkt steeds snel aangezien het aantal concerten slechts een klein deel is van het aantal producties.

Aangezien de performantie van het gemengd algoritme heel wat minder goed is dan dat van het individuele algoritme en de evaluatieresultaten ongeveer even goed zijn, werd geopteerd om het individuele algoritme te gebruiken.

Slotbeschouwing

In ons uitgebreid masterproefvoorstel (zie bijlage B) staat onze oorspronkelijke visie op deze masterproef. Het doel was om een Collaborative Filteringsysteem te realiseren voor Kunstencentrum Vooruit, dat gebruikt kan worden op onder andere de communitywebsite. De volgende vragen werden beantwoord:

1. Hoe verzamel je de nodige gegevens over de voorkeuren van gebruikers?
2. Hoe kan je de kwaliteit van aanbevelingen zo hoog mogelijk maken?
3. Hoe ga je om met *sparsity*, gebrek aan gegevens ?
4. Hoe zorg je ervoor dat het algoritme performant blijft terwijl de dataset groeit?

Het onderzoek werd een uitgebreide literatuurstudie waaruit bleek dat user-based Collaborative Filtering geen goede benadering is voor Kunstencentrum Vooruit omwille van verscheidene redenen. Uit het onderzoeksdeel werd duidelijk wat de achterliggende principes van elk besproken algoritme waren en hoe gestart kon worden met een concrete implementatie.

Het aanvankelijke opzet was om enkel concerten aan te raden, maar dat idee druist in tegen de principes van Vooruit: individuen hebben bijvoorbeeld een bredere interesse dan enkel concerten. De interessegeschiedenis van een gebruiker was niet eenvoudig te achterhalen doordat er met verschillende databases gewerkt wordt en de data niet erg *clean* zijn. Vooral het koppelen van de databases liep mank: de gegevens van een gebruiker in de websitedatabase zijn niet eenduidig te verbinden met de gegevens van dezelfde gebruiker in de ticketingdatabase. Hetzelfde probleem stelde zich voor producties.

Enkele algoritmen werden ingepast in het Taste framework. Daarna werden de algoritmen gecombineerd en geëvalueerd. Doordat het onderzoeksdomein nog niet volwassen is, bestaat er nog geen gestandaardiseerde evaluatiemethode. Er moest dus gezocht worden naar zinvolle metrieken en een goede testprocedure. Dit nam heel wat meer tijd in beslag dan oorspronkelijk voorzien doordat de literatuur soms erg beknopt is. Indien we van in het begin geweten hadden dat het evalueren zelf zoveel tijd vergde, dan waren we daar eerder aan begonnen.

Het is mogelijk om een suggestie te verklaren. Een gebruiker kan aanduiden dat een aanbeveling niet goed is. Bovendien werd er gebruik gemaakt van Last.fm webservices om extra gegevens op te halen en om artiesten aan gebruikers aan te bevelen.

Doordat we er voor gekozen hebben om alle types van producties aan te raden, krijgt een gebruiker ook aanbevelingen die op het eerste zicht niet in het profiel passen: dit is een eigenschap van Collaborative Filtering. In het masterproefvoorstel werd dit als uitbreiding vermeld. Doordat we niet user-based werken is het clusteren van gebruikers en het aanbevelen van voorstellingen aan clusters van gebruikers niet mogelijk. Het clusteren van de voorstellingen zelf is daarentegen wel mogelijk, de vraag *"welke voorstellingen lijken er op voorstelling x?"* kan ook beantwoord worden.

Het verwerken van de wetenschappelijke artikels was ook niet vanzelfsprekend. De informatiedichtheid ervan is erg hoog maar toch ontbreken er belangrijke details om een praktische realisatie mogelijk te maken. Er zijn weinig artikels met duidelijke voorbeelden. In onze thesis hebben we geprobeerd wel hierin te voorzien. Sommige artikels verwijzen naar andere om iets te verklaren, terwijl die verklaring er niet in te vinden is.

Enkele onderwerpen worden best verder onderzocht. Momenteel wordt er bijvoorbeeld geen rekening gehouden met de tabulijsten bij het berekenen van similariteiten en aanbevelingen. Door de beperkte tijd werd dit niet meer onderzocht. Dit zou zeker een nuttige toevoeging zijn aan het systeem. Ook het samennemen van algoritmen kan verbeterd worden.

Aangezien de codebase van Taste sinds kort overgenomen is door Apache Mahout zou het binnenkort eenvoudig moeten zijn om een schaalbaar aanbevelingssysteem te bouwen dat uitgevoerd kan worden op een cluster. Hoe de belasting verdeeld kan worden over een groep servers is een interessant onderzoeksdomein.

Een content-based aanbevelingsysteem kan geschreven worden om aanbevelingen te genereren gebaseerd op inhoud. Er kan daarmee een hybride systeem gebouwd worden dat de resultaten van content-based en Collaborative Filtering combineert.

Al bij al vonden we het een boeiende thesis. Het zelfstandig onderzoeken van wetenschappelijke artikels en daarmee iets praktisch realiseren was interessant. Enkele maanden meewerken in een bedrijf met een heel beperkte informatica-afdeling was leerrijk.

Literatuurlijst

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, 2005.

[2] Inc. Amazon.com. Amazon.com. `http://www.amazon.com`.

[3] Michelle Anderson, Marcel Ball, Harold Boley, Stephen Greene, Nancy Howse, Daniel Lemire, and Sean McGrath. RACOFI: A Rule-Applying Collaborative Filtering System. In *Proceedings of COLA'03*. IEEE/WIC, October 2003.

[4] Christopher Avery and Richard Zeckhauser. Recommender Systems for Evaluating Computer Messages. *Commun. ACM*, 40(3):88–89, 1997.

[5] Daniel Billsus and Michael J. Pazzani. Learning Collaborative Information Filters. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 46–54, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[6] J. Breese, D. Heckerman, and C. Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52, San Francisco, CA, 1998. Morgan Kaufmann.

[7] Lukas Brozovsky. Recommender System for a Dating Service. Master's thesis, Charles University in Prague, 2006.

[8] Mark Claypool, Phong Le, Makoto Wased, and David Brown. Implicit Interest Indicators. In *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*, pages 33–40, New York, NY, USA, 2001. ACM.

[9] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google News Personalization: Scalable Online Collaborative Filtering. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 271–280, New York, NY, USA, 2007. ACM.

[10] M. Deshpande and G. Karypis. Item-based Top-N Recommendation Algorithms, 2004.

[11] Jeffrey D'heer. Prefuse. `http://www.prefuse.org`.

[12] T. Fawcett. ROC Graphs: Notes and Practical Considerations for Data Mining Researchers, 2003.

[13] Malcolm Gladwell. The Science of the Sleeper: how the Information Age Could Blow Away the Blockbuster, 1999.

[14] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using Collaborative Filtering to Weave an Information Tapestry. *Commun. ACM*, 35(12):61–70, December 1992.

[15] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[16] University of Minnesota GroupLens Research. GroupLens. `http://www.grouplens.org`.

[17] University of Minnesota GroupLens Research. MovieLens. `http://www.movielens.org`.

[18] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An Algorithmic Framework for Performing Collaborative Filtering. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, New York, NY, USA, 1999. ACM.

[19] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining Collaborative Filtering Recommendations. In *CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250, New York, NY, USA, 2000. ACM.

[20] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[21] H. Hotelling. Analysis of a Complex of Statistical Variables into Principal Components. *Journal of Educational Psychology*, (24):417–441, 1933.

[22] Z. Huang, D. Zeng, and H. Chen. A Link Analysis Approach to Recommendation under Sparse Data. In *Proc. 2004 Americas Conf. Information Systems*, 2004.

[23] Zan Huang, Hsinchun Chen, and Daniel Zeng. Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering. *ACM Trans. Inf. Syst.*, 22(1):116–142, 2004.

[24] Zan Huang, Daniel Zeng, and Hsinchun Chen. A Comparison of Collaborative Filtering Recommendation Algorithms for E-commerce. *IEEE Intelligent Systems*, 22(5):68–78, 2007.

[25] George Karypis. Evaluation of Item-Based Top-N Recommendation Algorithms. In *CIKM*, pages 247–254, 2001.

[26] Byeong Man Kim, Qing Li, Chang Seok Park, Si Gwan Kim, and Ju Yeon Kim. A new Approach for Combining Content-Based and Collaborative Filters. *J. Intell. Inf. Syst.*, 27(1):79–91, 2006.

[27] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Reidl. Grouplens: Applying Collaborative Filtering to Usenet News. *Communications of ACM*, 40(3):77–87, March 1997.

[28] Daniel Lemire and Anna Maclachlan. Slope One Predictors for Online Rating-Based Collaborative Filtering. In *Proceedings of SIAM Data Mining (SDM'05)*, 2005.

[29] Daniel Lemire and Sean McGrath. Implementing a Rating-Based Item-to-Item Recommender System in PHP/SQL. Technical Report D-01, Ondelette.com, January 2005.

[30] Greg Linden, J.A. Jacobi, and E.A. Benson. US Patent 6,266,649 (to Amazon.com): Collaborative Recommendations Using Item-to-Item Similarity Mappings, 2001.

[31] Greg Linden, Brent Smith, and Jeremy York. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[32] Last.fm Ltd. Last.fm. `http://www.last.fm`.

[33] David Maltz and Kate Ehrlich. Pointing the Way: Active Collaborative Filtering. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 202–209, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[34] P. Massa and P. Avesani. Trust-Aware Collaborative Filtering for Recommender Systems, 2004.

[35] P. Massa and B. Bhattacharjee. Using Trust in Recommender Systems: an Experimental Analysis, 2004.

[36] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 17–24, New York, NY, USA, 2007. ACM.

[37] Paul McJones. Eachmovie Collaborative Filtering Dataset, DEC Systems Research Center, 1997.

[38] Bhaskar Mehta, Thomas Hofmann, and Wolfgang Nejdl. Robust Collaborative Filtering. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 49–56, New York, NY, USA, 2007. ACM.

[39] P. Melville, R. Mooney, and R. Nagarajan. Content-boosted Collaborative Filtering for Improved Recommendations, 2002.

[40] D. Nichols. Implicit Rating and Filtering. In *Proceedings of 5th DELOS Workshop on Filtering and Collaborative Filtering*, pages 31–36. ERCIM, 1997.

[41] Michael O'Mahony, Neil Hurley, Nicholas Kushmerick, and Guénolé Silvestre. Collaborative Recommendation: a Robustness Analysis. *ACM Trans. Inter. Tech.*, 4(4):344–377, 2004.

[42] K. Pearson. On Lines and Planes of Closest Fit to Systems of Points in Space. *Philosophical Magazine*, 2(6):559–572, 1901.

[43] David M. Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles. Collaborative Filtering by Personality Diagnosis: A Hybrid Memory and Model-Based Approach. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 473–480, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[44] Rob Procter and Andy McKinlay. Social Affordances and Implicit Ratings for Social Filtering on the Web, 1997.

[45] R Project. R Project for Statistical Computing. `http://www.www.r-project.org`.

[46] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, and John Riedl. Getting to Know You: Learning new User Preferences in Recommender Systems. In *IUI '02: Proceedings of the 7th international conference on Intelligent user interfaces*, pages 127–134, New York, NY, USA, 2002. ACM.

[47] E. Rich. User Modeling via Stereotypes. *Cognitive Science*, pages 335–366, 1979.

[48] John Riedl, Joseph Konstan, and Eric Vrooman. Word of Mouse: The Marketing Power of Collaborative Filtering, 2002.

[49] Badrul M. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of Dimensionality Reduction in Recommender Systems – a Case Study, 2000.

[50] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Reidl. Item-based Collaborative Filtering Recommendation Algorithms. In *World Wide Web*, pages 285–295, 2001.

[51] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Analysis of Recommendation Algorithms for E-commerce. In *ACM Conference on Electronic Commerce*, pages 158–167, 2000.

[52] Badrul M. Sarwar, Joseph A. Konstan, Al Borchers, Jon Herlocker, Brad Miller, and John Riedl. Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System. In *CSCW '98: Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 345–354, New York, NY, USA, 1998. ACM.

[53] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and Metrics for Cold-Start Recommendations. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, New York, NY, USA, 2002. ACM.

[54] Upendra Shardanand and Patti Maes. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.

[55] Tobias Sing, Oliver Sander, Niko Beerenwinkel, and Thomas Lengauer. ROCR: Visualizing Classifier Performance in R. *Bioinformatics*, 21(20):3940–3941, October 2005.

[56] L. Terveen and W. Hill. Beyond Recommender Systems: Helping People Help Each Other, 2001.

[57] S. Upendra. Social Information Filtering for Music Recommendation, 1994.

[58] C.J. Van Rijsbergen. *Information Retrieval*. Butterworth, second edition, 1979.

[59] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. Unifying User-Based and Item-Based Collaborative Filtering approaches by Similarity Fusion. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508, New York, NY, USA, 2006. ACM.

[60] Jun Wang, Arjen P. de Vries, and Marcel J.T. Reinders. A User-Item Relevance Model for Log-based Collaborative Filtering. In *Proc. of European Conference on Information Retrieval (ECIR 2006), London, UK*, 2006.

[61] Kai Yu, Anton Schwaighofer, Volker Tresp, Xiaowei Xu, and Hans-Peter Kriegel. Probabilistic Memory-Based Collaborative Filtering. *IEEE Trans. on Knowl. and Data Eng.*, 16(1):56–69, 2004.

[62] Kai Yu, X. Xu, J. Tao, M. Ester, and H. Kriegel. Instance Selection Techniques for Memory-Based Collaborative Filtering, 2002.

Deel III

Bijlagen

Bijlage A

Documentatie

Deze bijlage bevat confidentiële informatie. Daarom worden hier enkel de titels gepubliceerd.

A.1 Databases

A.1.1 Algemene configuratie

A.1.2 Websitedatabase

Gebruikers ophalen

Producties ophalen

Ratings ophalen

Artiestinformatie ophalen bij producties

Productie of artiest toevoegen aan tabulijst

A.1.3 Ticketingdatabase

Gebruikers ophalen

Producties ophalen

Ratings ophalen

A.2 WebsiteRecommenderService

A.2.1 Aanraden per gebruiker

A.2.2 Aanraden per productie

A.2.3 Tabulijsten aanpassen

A.2.4 Offline vernieuwing van de algoritmen

A.3 Serverconfiguratie

A.3.1 Tomcatconfiguratie

Logbestanden

Vooruitwebserviceconfiguratiebestand

Vernieuwen gegevens

Vooruitwebservice ontplooien

A.3.2 Rubyconfiguratie

A.3.3 Website

A.4 Toevoegingen aan de website

Bijlage B

Uitgebreid masterproefvoorstel

B.1 Inleiding

In dit document geven wij onze visie op onze masterproef. Eerst beschrijven we het probleem; daarna geven we een overzicht van wat ons te doen staat en mogelijke extra's. We vermelden ook de technologieën die we gaan gebruiken om onze masterproef tot een goed einde te brengen. Vervolgens behandelen we de vernieuwende aspecten van onze thesis die niet aan bod komen in onze opleiding. Tenslotte stellen we een structuur voor van de scriptie. Indien jullie nog meer informatie willen, dan verwijzen wij jullie naar onze thesiswebsite, nl. `http://vooruit.0110.be`.

B.2 Probleemstelling & doelstelling

De hoeveelheid informatie in de wereld neemt veel sneller toe dan we ze kunnen verwerken. Iedereen kent het gevoel verloren te zwemmen in de oceaan van informatie bij het zoeken naar een album, boek, film, concert, ... die bij je persoonlijke smaak past. Tegenwoordig bestaan er technologieën die dit probleem trachten op te lossen; de meest belovende is *Collaborative Filtering*.

Collaborative Filtering (CF) gebruikt een databank van voorkeuren van gebruikers voor items. Een nieuwe gebruiker wordt vergeleken met de gegevens uit de databank om zijn *nearest neighbours* te ontdekken die een gelijkaardige smaak hebben. De items die deze buren verkiezen, worden aanbevolen aan de nieuwe gebruiker, want de kans is groot dat deze ze zal appreciëren.

Collaborative Filtering is bijzonder succesvol, zowel als onderzoeksdomein als in de praktijk. Het wordt zowel toegepast in e-commerce (bv. `www.amazon.com`) als in in-

formatiefilterende applicaties (bv. `www.movielens.com`). De verschillende uitdagingen zijn:

1. Hoe verzamel je de nodige gegevens over de voorkeuren van gebruikers?
2. Hoe kan je de kwaliteit van aanbevelingen zo hoog mogelijk maken?
3. Hoe ga je om met *sparsity*, gebrek aan gegevens ?
4. Hoe zorg je ervoor dat het algoritme performant blijft terwijl de dataset groeit?

In onze thesis zullen we de bovenstaande uitdagingen aangaan rekening houdend met de beschikbare data en de wensen van de marketingafdeling van Kunstencentrum Vooruit. De thesis gaat uit twee delen bestaan:

1. Eerst een onderzoeksdeel waarin we een overzicht geven van de verschillende methoden om de voornoemde problemen aan te pakken.
2. Het tweede deel bestaat uit het implementeren en evalueren van verschillende Collaborative Filteringalgoritmen. Het algoritme of de algoritmen die het beste resultaat voor Vooruit levert of leveren worden daarna verpakt in een bruikbare webservice.

B.2.1 Onderzoek

In het onderzoeksdeel van de thesis gaan we op zoek naar gekende Collaborative Filteringalgoritmen. We brengen ze daana onder in verschillende categorieën en bespreken de voor- en nadelen van elke methode.

- Memory based $\Leftrightarrow$ model based,
- expliciet $\Leftrightarrow$ impliciet filteren,
- user-based $\Leftrightarrow$ item-based filteren.

Aan het einde van het onderzoeksdeel zou duidelijk moeten zijn wat de achterliggende principes zijn van elk besproken algoritme en hoe gestart kan worden met een concrete implementatie, met de specifieke eisen.

B.2.2 Implementatie

Tijdens de vergadering met de marketingafdeling en de externe promotor werden er enkele vereisten vastgelegd. Zo is het niet de bedoeling enkel items aan te bevelen binnen Vooruit zelf, maar ook externe items dienen aan bod te komen. Zo kunnen er bij concerten aanbevelingen gedaan worden voor albums van groepen die geen concert geven in Vooruit.

Ook werd vastgelegd dat het programma in de eerste plaats ingezet wordt voor muziek, maar het programma moet uitbreidbaar zijn voor andere items zoals theater en dansvoorstellingen.

De data moeten genormaliseerd worden voordat we deze kunnen gebruiken. De data wordt uit verschillende databases gehaald, zoals de website- en de ticketingsysteemdatabase. Daarnaast gebruiken we ook externe bronnen zoals de last.fm[1] webservices. Het datamodel moet ook uitbreidbaar zijn, andere databronnen moeten toegevoegd kunnen worden. Aan de hand van die data bepalen we welke algoritmen we kunnen gebruiken. Van die algoritmen wordt ofwel een implementatie gemaakt en ingepast in het open source framework Taste[2], ofwel wordt een beschikbare implementatie gebruikt. Daarna testen we de algoritmen op kwaliteit en performantie en kiezen we de beste. Indien dit er meerdere zijn, combineren we deze.

Elke aanbeveling moet een oorsprong aanduiden, bijvoorbeeld a.d.h.v. externe bronnen en vrienden raden we dit concert aan. Daarnaast is om twee redenen een feedbacksysteem gewenst:

1. De gebruikers kunnen aanduiden welke aanbevelingen niet kloppen met hun persoonlijke smaak. Ze hebben de aanbevelingen zo beter in de hand.
2. De data die door het feedbackmechanisme verzameld worden, is een goede indicator van smaak en kan dus gebruikt worden als invoer voor het systeem.

Daarna wordt er voor de interface naar de buitenwereld gezorgd via een webservice. Die webservice zal voornamelijk geconsumeerd worden door de website, maar eventueel ook door andere applicaties (zie uitbreidingen).

B.3 Uitbreidingen

Er kunnen bijzonder veel uitbreidingen gekoppeld worden aan deze thesis, hieronder sommen we de meest voor de hand liggende op.

[1] Om groepen te linken aan andere groepen zie `www.audioscrobller.net`

[2] Zie "Taste, collaborative filtering for java": `http://taste.sourceforge.net/`

De bezoekers van Vooruit zijn vragende partij om aanbevelingen te krijgen die *niet* in het profiel passen van een bepaalde gebruiker. Het publiek van Vooruit is namelijk nieuwsgierig en wil niet in een vakje ingedeeld worden. Zo kunnen ze nieuwe zaken ontdekken.

Het is de bedoeling dat we Collaborative Filtering in de eerste fase enkel maar implementeren voor concerten, omdat we daarvoor over de meeste gegevens beschikken. Toch is het een mooie uitbreiding om het ook toe te passen voor theater, dans, ... Alles wat we maken moet dus perfect uitbreidbaar zijn en uiteraard ook goed gedocumenteerd.

In plaats van aanbevelingen te maken voor individuele gebruikers, kunnen we dit doen voor een groep van gebruikers. Dit wil wel zeggen dat ze zich zelf lid maken van een bepaalde de groep (bv. studenten). Die groep krijgt een bepaald profiel en aan de hand hiervan kunnen aanbevelingen gemaakt worden voor de volledige groep i.p.v. individuele gebruikers.

Eventueel kunnen de gebruikers ook zelf kiezen aan welke parameters ze het meeste belang hechten. Aan die parameters geven ze een bepaald gewicht en zo krijgen ze nog specifiekere voorstellen. Bijvoorbeeld kunnen ze opteren voor alternatieve concerten i.p.v. populaire.

Een mogelijke uitbreiding, die eigenlijk weinig te maken heeft met Collaborative Filtering is er een voor de marketing. Door het clusteren van gebruikers a.d.h.v. het nearest neighbouralgoritme kunnen demografische gegevens achterhaald worden. Met deze gegevens kan bv. de vraag: 'Wie gaat naar rockconcerten?' opgelost worden.

B.4 Technologieën

Er zijn een aantal technologieën die gebruikt worden in de huidige infrastructuur, daar kunnen we dus niet van afwijken. De website van Vooruit is geschreven in Ruby on Rails, een webapplicationframework in de scripttaal Ruby. De website gebruikt het MySQL DBMS en is volledig onafhankelijk van het back-office ticketingsysteem, dat een Oracle database gebruikt.

Wij zullen hoogstwaarschijnlijk gebruik maken van JAVA om de algoritmen en de webservice te implementeren en wel om de volgende redenen:

- Er zijn reeds enkele Open Source frameworks, geschreven in JAVA, beschikbaar die CF algoritmen implementeren.
- We hebben allebei veel ervaring met die programmeertaal.
- Er zijn voorzieningen die het creëren van webservices eenvoudiger maken.

- Het is platformonafhankelijk en kan dus op om het even welke server gebruikt worden.

B.5 Vernieuwende aspecten

Hoewel we nu al weten hoe om te gaan met eenvoudige algoritmen is een CF-algoritme toch andere koek. Het verwerken en implementeren ervan is in tegenstelling tot de geziene algoritmen niet in een namiddag te klaren. Wetenschappelijke artikels zijn minder toegankelijk dan de geziene cursussen. Ook het zoeken van relevante artikels loopt niet altijd van een leien dakje. Het systeem zal effectief operationeel worden. Er komt veel meer finetuning en optimalisatie aan te pas dan bij een algoritme oefening tijdens de labo's.

Collaborative filtering werd, voor zover wij weten, nog nooit gebruikt in een kunstencentrum en nog nooit om voorstellingen aan te bevelen. CF gebruiken is ook vernieuwend om het aanraden van items die zich niet direct in de database bevinden (artiesten die geen concert geven in Vooruit).

Het gebruik maken van Ruby on Rails is ook nieuw. Tijdens onze opleiding hebben wij wel verschillende MVC-webframeworks gezien, maar toch is deze anders. Ook de scripttaal Ruby hebben we nog niet gezien.

B.6 Overzicht scriptie

I Onderzoek

(a) Probleemschets

(b) Mogelijke methoden

(c) Collaborative filtering

i. Geschiedenis

ii. User based - item based

iii. Impliciet filteren - expliciet filteren

iv. Categorieën algoritmen

II Implementatie

(a) Beschikbare data

(b) Gekozen algoritmen

(c) Taste

(d) Evaluatie algoritmen

   i. Hoe?

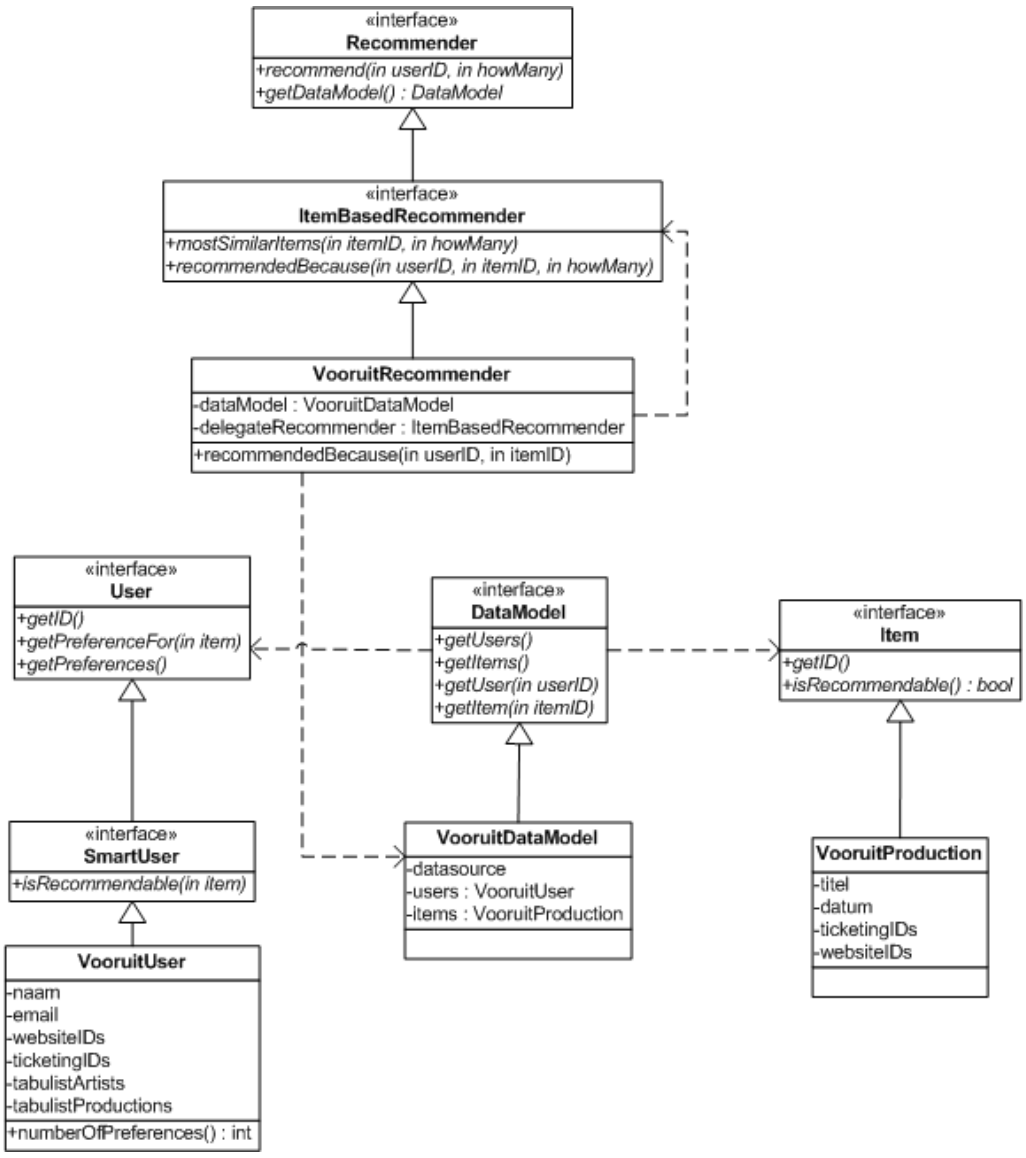
   ii. Resultaten

   iii. Keuze - besluit

(e) Optimalisatie

Bijlage C

UML-diagram



Figuur C.1: Klassendiagram van belangrijkste componenten van het aanbevelingssysteem