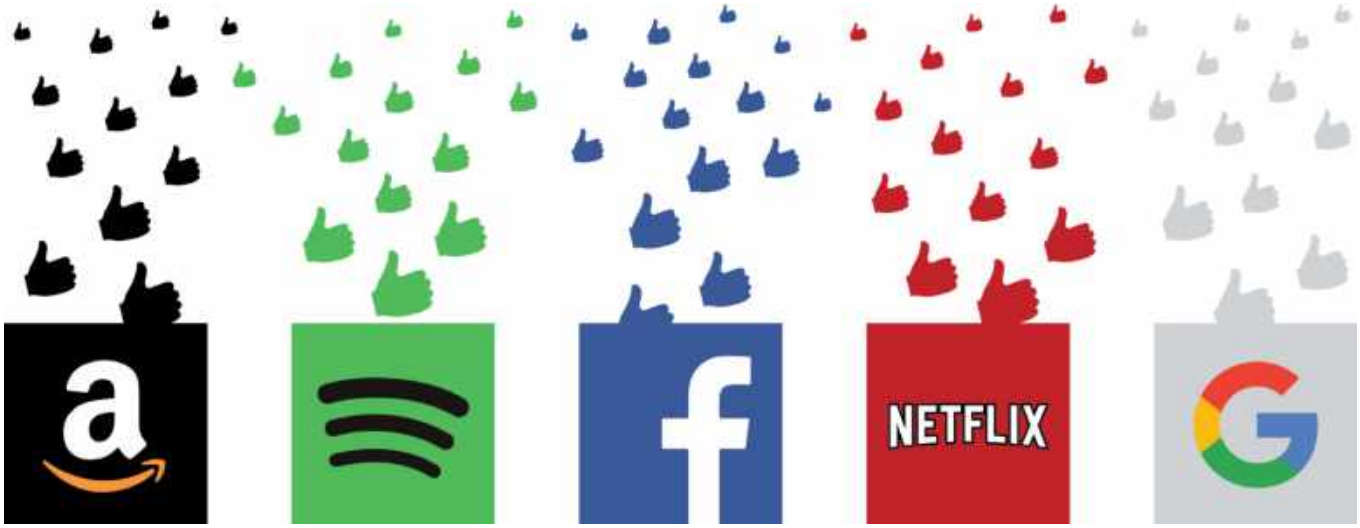
UHASSELT

Recommendation Algoritmes

Bachelorproef voorgedragen tot het behalen van de graad van bachelor in de informatica

Student: **Vervoort Tim**
Promotor: prof. **Van den Bussche Jan**



Academiejaar: **2017-2018**

Universiteit Hasselt

Samenvatting

In deze bachelorproef worden recommendation systemen besproken. Bijna alle online diensten maken gebruik van zulke systemen om items aan de gebruikers aan te bevelen met als uiteindelijke doel de revenue streams te verhogen. De auteur zal in dit document de meest courante methoden om recommender systemen te realiseren oplijsten en verder toelichten. Doorheen deze paper worden algoritmes geïllustreerd aan de hand van een test dataset met evenementen. Uiteindelijk zal een aanbevelingsalgoritme uitgewerkt worden rond deze evenementendatabase en geëvalueerd worden ten opzichte van de besproken methoden. Zo zullen de resultaten van de data extrapolatiemethode vergeleken worden met een test dataset.

Er wordt dieper ingegaan op de werkwijze en correctheid van methodieken, alsook de complexiteit van de operaties. Deze algoritmes worden naast elkaar vergeleken op dezelfde dataset en vergeleken op basis van accuraatheid *(door middel van diverse soorten metingen)*. Daarnaast tracht de auteur van deze bachelorproef een verklaring te geven waarom het ene algoritme beter is of meer juiste resultaten geeft dan het anderen en hoe men tot de notie van de respectievelijke berekening gekomen is.

Het eerste deel van deze bachelorproef bestaat uit een theoretische uiteenzetting van de soorten algoritmes en de achterliggende psychologische factoren die het genereren van aanbevelingen bemoeilijkt. Het tweede deel zal daarentegen de praktische kant van recommendation algoritmes beschrijven aan de hand van een eenvoudige implementatie in Python 3.

De globale structuur van deze paper is grotendeels gebaseerd op het handboek Recommender Systems: The Textbook van Charu C. Aggarwal uitgegeven door uitgeverij Springer.

Tim Vervoort
Studentnummer: 1539818

Foto cover:
medium.com/the-graph/popularity-vs-diversity-c5bc22c253ee

Inhoudsopgave

1. Inleiding

Recommendation algoritmes zijn tegenwoordig niet meer weg te denken uit het dagelijkse bestaan. Heel veel mensen interageren dagelijks met meerdere van deze systemen zonder dat ze dat eigenlijk goed beseffen. Bijna alle - *om niet te zeggen alle* - grote online platformen maken gretig gebruik van aanbevelingssystemen om hun gebruikers relevante inhoud aan te bieden en zo de gebruikerservaring te verbeteren. Uiteraard zijn deze aanbevelingen niet alleen interessant voor de gebruikers. Het primaire doel is natuurlijk om abonnees zoveel mogelijk content aan te bieden die ze gaan consumeren. De doelstelling hiervan is om de revenue streams van het platform te verhogen. Deze aanbevelingssystemen worden gebruikt in heel diverse domeinen:

- **Video platformen**: online streamingdiensten zoals internetgiganten Netflix[1] en Amazon Prime Video[2] bevelen films en series *(ook vaak eigen producties)* aan om de kijkcijfers te verhogen. Het online video sharing platform YouTube[3] gebruikt dan weer recommendation algoritmes om gebruikers meer video's te laten kijken en meer tijd door te laten brengen op de site of app om zo meer advertenties te kunnen tonen, wat resulteert in een verhoging van de advertentie-inkomsten. Deze drie platformen baseren hun suggesties op eerder bekeken video's en/of zoektermen. Hiernaast vindt de lezer een illustratie van YouTube suggesties. In figuur 'Netflix aanbevelingen' in de paragraaf over Soorten recommendation algoritmes worden de aanbevelingen van Netflix geïllustreerd.

- **Muziek streaming**: online audio streaming diensten zoals Spotify[4], Apple Music[5], Jango[6], maar ook de Belgische Klara app[7] genereren afspeellijsten op basis van het luistergedrag van de gebruiker. Via deze suggesties kunnen luisteraars nieuwe artiesten ontdekken binnen het platform. Voor deze bedrijven is het interessant dat gebruikers naar meer nummers luisteren, zodanig dat er meer reclame gehoord kan worden *(via de gratis platformen)* en om de gebruikerservaring te verbeteren.

- **Webshops**: de grootste webshop ter wereld[8], Amazon[9], stelt tijdens het browsen van de website gerelateerde producten van de webwinkel voor die relevant zijn met onlangs bekeken items of uitgevoerde zoektermen. Het aanbevelen van interessante items voor potentiële kopers maakt de kans op aankoop - en daarmee ook de omzet - van webshops hoger.

[1] netflix.com
[2] primevideo.com
[3] youtube.com
[4] spotify.com
[5] apple.com/benl/music
[6] jango.com
[7] klara.be/de-klara-app

[8] Bezos's empire: how Amazon became the world's most valuable retailer
Auteurs: Josh Holder & Alex Hern The Guardian - 24/04/2018
theguardian.com/technology/ng-interactive/2018/apr/24/bezoss-empire-how-amazon-became-the-worlds-biggest-retailer - Geraadpleegd op: 28/05/2018

[9] amazon.com

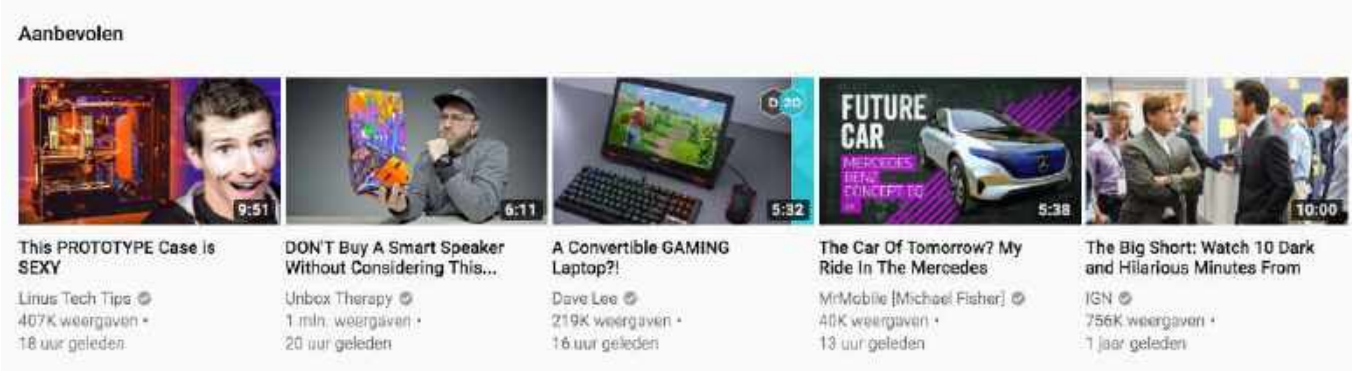
- **Zoekmachines**: de meest populaire zoekmachines op het World Wide Web, waaronder Google[10], Bing[11] en Yahoo[12], personaliseren zoekresultaten voor elke gebruiker op basis van hun gebruikersprofiel. Daarnaast gebruiken deze systemen pageranking en de context van gebruikersactiviteit om de volgorde van de zoekresultaten te optimaliseren en surfers zo de beste zoekresultaten bovenaan te presenteren.

- **Nieuws websites**: online nieuwskanalen zoals Yahoo[13] en Google[14] bevelen krantenartikels aan die overeenstemmen met de gebruiker zijn of haar interesses. Zo kan men op een gebruikersprofiel interesses selecteren op basis van welke men dan nieuws ontvangt.

- **Data filtering**: GMail[15], de maildienst van internetgigant Google, filtert spam uit de mailbox op basis van eerdere gebruikersacties *(zoals het markeren van reclamemails als spam)* en globale gebruikersactiviteit *(meerdere gebruikers markeren een afzender als spam adres)*. De alternatieve mailbox, Inbox van Google[16], filtert belangrijke mails in een priority inbox op basis van eerdere gebruikersacties.

- **Online dating**: online dating bureaus zoals het Amerikaanse OkCupid[17] gebruiken recommendation algoritmes om de 'perfecte' match te zoeken op basis van overeenkomstige interesses en criteria[18].

- **Online advertenties**: Google Adwords[19] *(samen met Google Adsense)* en Yahoo Gemini[20] bieden relevante advertentiemodellen aan door reclame op maat van de surfer *(of de website waarop de reclame zichtbaar is)* te tonen. Via deze weg wordt reclame heel gericht naar de juiste doelgroep geleid en is hun adverteer model interessanter dan andere kanalen.



Figuur: YouTube aanbevelingen

[10] google.be
[11] bing.com
[12] yahoo.com
[13] yahoo.com/news
[14] news.google.com
[15] google.com/intl/nl/gmail/about
[16] google.com/inbox
[17] okcupid.com

[18] Paper: Online dating recommendations: matching markets and learning preferences
K. Tu, B. Ribevio, D. Jensen, D. Towsley, B. Liu, H. Jiang & X. Wang University of Massachutsetss - 11/04/2014
dl.acm.org/citation.cfm?id=2567948.2579240 - Geraadpleegd op: 28/03/2018

[19] adwords.google.com/intl/nl_be/home
[20] gemini.yahoo.com/advertiser/home

Suggesties voor een gebruiker zijn gebaseerd op de **geschatte rating** of associatie die de **gebruiker voor** of met een **item** *(video, nummer, boek …)* zou hebben. Items die de gebruikers als leuk of interessant zou aanduiden *(hoge rating, aankoop van een item of het luisteren van een nummer)* worden weergeven in de suggesties. Vaak wordt een **top K** *(met K als een positief geheel getal)* van items met de hoogst geschatte beoordelingen gesuggereerd aan de gebruikers. De geschatte numerieke ratings worden vrijwel nooit getoond om de aanbevelingen niet te verwarren met reeds beoordeelde items.

Om gebruikersbeoordelingen te kunnen schatten is er data nodig. Deze data omvatten: beoordelingen, item profielen en gebruikersprofielen. Op deze data worden dan berekeningen gedaan die leiden tot een lijst van potentieel interessante items voor een gebruiker. Deze lijst wordt vaak aflopend gesorteerd op waarschijnlijkheid *(hoogste geschatte ratings eerst)* alvorens een top K items aan de gebruiker doorgegeven wordt. Merk op dat deze lijst van aanbevelingen enkel items bevat die de gebruiker nog niet beoordeeld heeft en waarvan de geschatte beoordeling een positieve semantiek heeft.

1.1 Wat is een recommendation algoritme?

Een recommendation algoritme, misschien niet iets waar iedereen zich iets bij kan voorstellen. De naam zegt het eigenlijk zelf, een recommendation algoritme gaat recommendations of **aanbevelingen genereren** voor gebruikers van een platform of dienst.

Op basis van **gebruikersacties,** zoals geschreven recensies, gemaakte beoordelingen, aankopen van items, het bekijken van items, het uitvoeren van zoekfuncties enz. kan een systeem een lijst bijhouden van alle gebruikersrecensies. Deze recensies kunnen gebruikt worden om gebruikers te typeren. Voor elk account kan een gebruikersprofiel opgesteld worden. Wanneer bijvoorbeeld blijkt dat een gebruiker vaak 'Die Hard' films *(actiefilms*) met 5 sterren beoordeeld, kan hieruit afgeleid worden dat de gebruiker mogelijk fan is van actiefilms. Vervolgens kan het attribuut genre met als waarde 'actie' toegevoegd worden aan zijn gebruikersprofiel of de actie quantifier in het gebruikersprofiel kan op 1 gezet worden.

Een analoge redenering bestaat ook voor items. Wanneer de gebruikersprofielen bekend zijn, kan aan de hand van ratings bepaald worden wat voor soort item het is. Meer bepaald kunnen de item attributen van het item profiel ingevuld worden. Wanneer gebruikersprofielen met een voorkeur voor actiefilms een item op regelmatige basis een positieve beoordeling geven, kan er van uitgegaan worden dat dit item waarschijnlijk een actiefilm is.

Gebruikers kunnen met elkaar vergeleken worden, alsook items kunnen onderling vergeleken worden. Door de **nearest neighbours** van een hoog beoordeeld item te zoeken, kunnen suggesties voor de gebruiker gezocht worden. Anderzijds kan er gekeken worden naar hoog beoordeelde items van verwante gebruikers met gebruiker X, welke aanbevelingen kunnen zijn voor gebruiker X.

Grosso modo kan men twee methodes om aanbevelingen te genereren onderscheiden:

- **Content-based**: op basis van de inhoud van item- en gebruikersprofielen wordt gezocht naar aanbevelingen. Item profielen worden gematcht met gebruikersprofielen of andere item profielen. Zo wordt een item aan een gebruiker X aanbevolen indien het item profiel matcht met een eerder reeds door gebruiker X hoog beoordeeld item of indien het item profiel matcht met het gebruikersprofiel van X.

- **Collaborative**: aan de hand van groepen items of groepen gebruikers worden items aanbevolen. Door te kijken naar overeenkomstige beoordelingen kan een item geassocieerd worden met andere items of kan een gebruiker geassocieerd worden met andere gebruikers. Door naar de overeenkomstige groep items of gebruikers te kijken kunnen aanbevelingen geëxtraheerd worden uit de peergroep.

Deze twee groepen recommendation systemen worden in het hoofdstuk 'Soorten recommendation algoritmes' verder toegelicht. Het grote verschil tussen deze twee is dat bij content-based enkel gekeken wordt naar de features in item- en gebruikersprofielen, de 'content' van deze objecten. Bij collaborative methodes wordt er enkel gekeken naar de context van items of profielen. Items worden in verband gebracht met andere items en gebruikers worden in verband gebracht met andere gebruikers. Hierna worden de groepen items of gebruikers met elkaar vergeleken om aanbevelingen te genereren. Vandaar het 'collaborative' of samenwerkend genereren van suggesties.

Formeel is een recommendation systeem een algoritme dat gaat werken op een rating **matrix M** *(een tabel met beoordelingen)* en tracht de onbekende waardes in deze matrix M in te vullen *(schatten)* aan de hand van bestaande ratings in die matrix M. In matrix M vormen de rijen de gebruikers en de kolommen de items. De rating van gebruiker X op item I wordt voorgesteld door M_{ij}. Items met een hoge geschatte rating voor gebruiker X kunnen uiteindelijk aanbevolen worden aan gebruiker X. *(Items met een hoge reeds bestaande rating zijn niet interessant om aan te bevelen, daar de gebruiker reeds geïnterageerd heeft met deze items. Daarnaast zijn items met een lage rating niet interessant om te suggereren, daar de gebruiker waarschijnlijk niet geïnteresseerd is in items die hij of zij niet leuk zal vinden.)* Vaak wordt een top K lijst van items gegenereerd welke aanbevolen kan worden aan de gebruiker. Deze lijst van aanbevelingen kan dan gekoppeld worden aan een gebruikersprofiel om later snel teruggevonden te worden.

1.2 Waarom een recommendation algoritme?

Waarom zou een gebruiker nu artificiële aanbevelingen willen krijgen van een systeem? Reclamevensters zijn ook vormen van aanbevelingen die gegenereerd worden door een categorie van recommendation systemen. Veel mensen zouden deze systemen zelfs als vervelend beschouwen, waarom wordt er dan zoveel geïnvesteerd in én onderzoek gedaan naar recommendation algoritmes?

Recommendation systemen faciliteren een **verhoging** van **inkomsten** voor een platform[21]. De catalogus van items *(video's, muziek, producten ...)* is vaak gigantisch. Een gebruiker zoekt naar een item, maar weet niet precies wat hij of zij zoekt *(zoals een bepaald genre film maar niet perse een titel)* of hoe hij of zij juist zijn of haar zoekterm moet verwoorden. Door relevante items aan de gebruiker te suggereren op basis van wat men weet van de gebruiker, kan de gebruiker uiteindelijk toch vinden waar hij of zij initieel naar zocht. Daarbij kan zijn of haar nieuwsgierigheid geprikkeld worden om een voorgesteld item te gaan ontdekken en ermee te interageren *(item bekijken, aankopen, advertentie impressie enz.)*.

Niet alleen voor de platform beheerders zijn recommendation systemen interessant. Ook gebruikers van platformen kunnen baat hebben bij deze systemen. De gebruiksvriendelijkheid van een systeem wordt vaak verbeterd door een recommendation systeem te gaan gebruiken. Denk maar aan een zoekfunctie die de volgorde van de zoekresultaten wijzigt afhankelijk van de context van individuele gebruikers *(zoals bij Google[22])* of een muziek streaming dienst welke de afspeellijst perfect afstemt op het gevoel van de luisteraar(s) zoals bij Tunify[23].

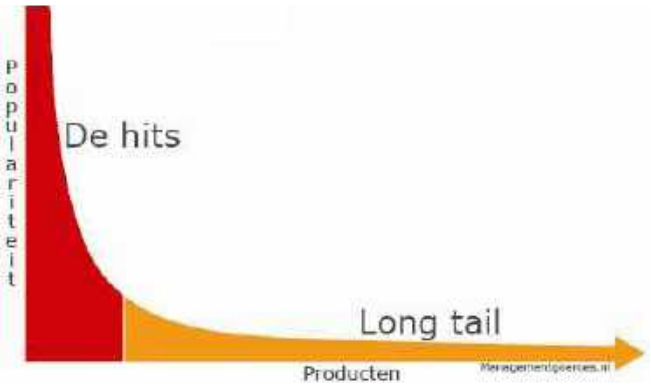
---

[21] Blog post: 4 Ways How Predictive Analysis and Recommendation Systems Can be Used to Increase Sales
Trouvus Recommender Systems company, Canada - 12/01/2016
trouvus.com/4-ways-how-predictive-analysis-and-recommendation-systems-can-be-used-to-increase-sales
Geraadpleegd op: 28/03/2018

[22] Artikel: Google Now Personalizes Everyone's Search Results
Danny Sullivan, Journalist Search Engine Land - 04/12/2009
searchengineland.com/google-now-personalizes-everyones-search-results-31195 - Geraadpleegd op: 29/03/2018

[23] tunify.com/nl_BE/home

1.2.1 Transitie van schaarste naar overvloed

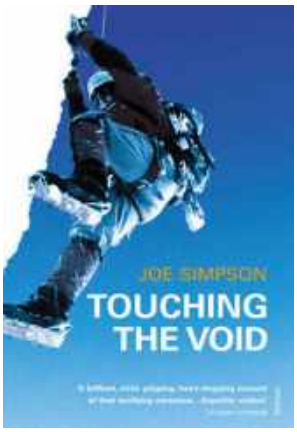


Figuur: long tail

Voor het digitale tijdperk was fysieke winkelruimte **beperkt** en **duur**, uitzendtijd voor televisiezenders een vast tijdslot en het aantal zalen in de cinema en bijgevolg het aantal te projecteren films een vast gegeven. Met de opkomst van het internet is het héél goedkoop geworden om veel informatie en dus veel items digitaal beschikbaar te maken. Het werd mogelijk om enorm veel producten of diensten aan te bieden zonder veel fysieke winkelruimte, zendtijd of cinemazalen ter beschikking te hebben. Een evolutie van schaarste naar overvloed. Een nadeel van zo een grote item catalogus is dat veel items heel moeilijk of zelfs niet vindbaar zijn tussen een overstroming van zoekresultaten. Daarnaast weet de gebruiker vaak niet wat kiezen tussen een overvloed resultaten[24]. Daar deze items niet gevonden worden, daalt hun populariteit nog meer. Dit fenomeen wordt de **long tail**[25] genoemd. Om juist deze items te kunnen suggereren en relevante items te tonen aan de gebruiker zijn recommendation algoritmes ontwikkeld.

1.2.2 De kracht van recommendation algoritmes

Dat recommendation systemen nuttig kunnen zijn voor zowel de gebruiker als de aanbieder lijkt voor de lezer aanneembaar. De kracht van deze systemen blijkt uit een lucratief voorbeeld: het boek 'Touching the Void' op Amazon.



Figuur: boek Touching the Void

Het boek 'Touching the Void' illustreert de kracht van recommender systemen [26]. Toen het boek 'Touching the Void' in 1998 gepubliceerd werd, kreeg het beperkte aandacht van het publiek en belandde het uiteindelijk in de eerder genoemde long tail. Later werd een gelijkaardig boek 'Into Thin Air' gepubliceerd dat echter wel snel een bestseller werd. Het recommendation algoritme van Amazon *(reeds actief als webshop sinds 1995[27])* zag een gelijkenis tussen de producten 'Into Thin Air' en 'Touching the Void' *(de boeken gaan beiden over een klimmers ongeval)* en beveelde aan kopers van het succesvolle boek 'Into Thin Air' het boek 'Touching the Void' aan. Deze aanbeveling op de immens populaire webshop zorgde voor een exponentiële stijging van verkoopcijfers van 'Touching the Void'. Via deze weg werd het achteraf een nog groter succes dan 'Into Thin Air'. Zonder recommendation systemen was het boek 'Touching the Void' waarschijnlijk onopgemerkt gebleven en uiteindelijk verdwenen in de long tail.

---

[24] Blog post: Marketing for Scarcity in a World of Abundance
Clare Tidby, Time Space Media - 28/11/2017
blog.timespacemedia.com/marketing-for-scarcity - Geraadpleegd op: 29/03/2018

[25] Artikel: The long tail Chris Anderson uitleg & voorbeelden
Rick de Vlieger, Management Goeroes - 20/03/2013
managementgoeroes.nl/the-long-tail-chris-anderson-uitleg-voorbeelden - Geraadpleegd op: 26/03/2018

[26] Artikel: The Long Tail
Chris Anderson, WIRED - 01/10/2004
wired.com/wired/archive/12.10/tail.html - Geraadpleegd op: 26/03/2018

[27] Artikel: 15 fascinating facts you probably didn't know about Amazon
Avery Hartmans, Business Insider - 09/04/2017
businessinsider.com/jeff-bezos-amazon-history-facts-2017-4 - Geraadpleegd op: 28/03/2018

De hoofdtaak van een recommendation algoritme is om **relevante objecten aan** te **bevelen** aan de gebruikers van een digitaal platform. Een aanbevelingssysteem schat eigenlijk de score die een gebruiker zou toekennen aan een item.

Bij het geven van aanbevelingen moeten een aantal facetten bekeken en onderzocht worden: het gebruikersprofiel van de gebruiker die de items bekijkt, de items die aanbevolen worden en de context tijdens het raadplegen van de aanbevelingen *(het apparaat, de locatie en de tijd)*. Samen met de context moet ook de interface die de gebruiker heeft in acht genomen worden. De betekenis van de volgorde, grootte en indeling van items varieert naarmate de schermgrootte. Afhankelijk van de indeling kunnen aanbevelingen, meer specifiek de volgorde en indeling van aanbevelingen, een impact hebben op de betekenis die de gebruiker mogelijk hecht aan een item. Ook de toegekende geschatte rating heeft een invloed op de perceptie van de gebruiker[28]. Zo kan een geschatte 4 sterren i.p.v. 5 sterren de indruk geven dat het item toch niet zo heel goed is.

Veel literatuur illustreert recommendation algoritmes aan de hand van de Netflix dataset. De reden hiervoor is dat Netflix een van de weinig publiek toegankelijke datasets was. Echter is deze tabel omwille van privacyredenen niet meer beschikbaar[29]. De voormalige dataset van Netflix bevat records met *userId*, *movieId*, *dateRating* en *score*[30]. De reden dat de dataset offline gehaald werd is dat gebruikers terug gekoppeld konden worden aan de gebruikersprofielen op de populaire film review website IMDb[31]. Dit maakte de Netflix dataset niet langer anoniem.

1.2.3 Simpele aggregaten

In de prille jaren van het internet, toen webshops nog maar pas bestonden, waren recommendation systemen nog niet voorhanden, of toch niet in de vorm hoe dat we die nu kennen. Voor de 'uitvinding' van recommendation systemen werden aanbevelingen met de hand gemaakt *(hand curated)*. Dit is vaak nog steeds op webshops terug te vinden in de vorm van 'staff picks', 'staff favourites' en een lijst van 'essentiële items met de hand gekozen'. Hieronder een screenshot van de website van Coolblue[32] *(maandag 12 maart 2018)* waar de categorie 'Coolblue's Keuze' wordt teruggevonden. Dit is een lijst van producten die een werknemer *(webbeheerder)* handmatig opgesteld heeft *(al dan niet met financiële motivatie van de getoonde merken),* bijvoorbeeld op basis van verkoopcijfers.

Figuur: Coolblue's Keuze

[28] Blog post: Users' Perception of Product Ratings (New Qualitative & Quantitative Findings) Jamie Appleseed, Baymard Institute - 25/03/2015 baymard.com/blog/user-perception-of-product-ratings - Geraadpleegd op: 29/03/2018

[29] archive.ics.uci.edu/ml/noteNetflix.txt
[30] web.archive.org/web/20090925184737/http://archive.ics.uci.edu/ml/datasets/Netflix+Prize
[31] imdb.com
[32] coolblue.be

Later ontstonden simpele aggregaten zoals 'Top 10', 'meest recent' en 'meest populair'. Deze aggregaten kunnen gezien worden als een automatisering van de hand curated lijsten. In de screenshot hierboven is ook te zien dat de webshop van Coolblue gebruikt maakt van 'Top 10' *(links in het menu op de figuur)*. Deze aanbevelingen houden enkel rekening met de globale gebruikers activiteit. Resultaten zijn niet individueel en niet-gepersonaliseerd, enkel afhankelijk van het integrale systeemgebruikers-gedrag *(hoe vaak en hoe lang items gemiddeld worden bekeken of gekocht)*. Pas later zijn recommender systems ontwikkeld die op maat gemaakt zijn voor de individuele gebruikers zoals de algoritmes op webplatformen Netflix en Amazon.

1.3 Hoe werkt een recommendation algoritme?

Een recommendation algoritme probeert de gebruikersbeoordeling van een gebruiker op een item te schatten. Deze schatting maakt het algoritme aan de hand van beschikbare data *(de verzameling ratings, informatie over de items en gebruikers enz).* Deze data kan op diverse manieren verzameld worden.

1.3.1 Dataverzameling

Om goede schattingen te maken van welke items een gebruiker al dan niet hoog zou beoordelen is er veel data nodig. Deze data kunnen zijn: item attributen, gebruikers-voorkeuren, beoordelingen van gebruikers, wanneer welk item bekeken werd enz. De rating matrix M wordt gevuld door data op een platform te verzamelen. Enerzijds is er **implicite dataverzameling** door **gebruikersacties** *(kliks, views, tijd van een view enz.)* te loggen *(automatisch verzamelen)*. Anderzijds kan er **expliciet** data verzameld worden door gebruikers die **recensies achterlaten** over items. Die beoordelingen kunnen een numerieke score zijn of een geschreven beoordeling die omgezet kan worden in een positieve of negatieve score op basis van de geschreven tekst. Het is wel belangrijk om een duidelijk onderscheid te maken tussen gekende en ongekende ratings *(er moet bijvoorbeeld een default waarde zijn die niet kan voorkomen binnen de ratingschaal)* zodat reeds beoordeelde items niet per ongeluk tussen de suggesties terecht komen. De reden hiervoor is dat het zinloos is om reeds beoordeelde items aan de gebruiker aan te bevelen *(tenzij het de bedoeling is om de gebruiker het item opnieuw te laten consumeren).*

Figuur: expliciete feedback op Netflix

Bron figuur: medium.com/@bradknox/4-ways-netflixs-all-thumbs-rating-system-abandons-its-power-users-35405c04cb9

Door enkel expliciete data te verzamelen moet gebruikers van het systeem gevraagd worden om items te beoordelen. Deze methode is niet **schaalbaar,** daar slechts een heel **klein deel gebruikers** een rating of review wil achterlaten[33]. Tenslotte zijn gebruikers sneller geneigd een **negatieve review** achter te laten wanneer hen iets niet beviel, dan aan het item positief te beoordelen wanneer het hen juist wel beviel[34] [35]. Met deze **bias** moet rekening gehouden worden.

Hoewel implicite dataverzameling veel schaalbaarder is dan explicite dataverzameling *(loggen gaat automatisch op elke gebruikersactie)* is de **semantiek** van de data minder duidelijk. Zo kan het kopen van een item gezien worden als een hoge rating geven, maar het niet kopen van een item impliceert niet noodzakelijk een lage rating. *(Een item bekijken maar niet kopen wilt niet automatisch zeggen dat het item voor de gebruiker niet interessant is. Mogelijks heeft de koper het item niet meteen nodig, wacht hij nog even met de aankoop of heeft hij momenteel niet voldoende financiële middelen voor de aankoop.)* Deze ratings zijn moeilijker te gebruiken, tenzij ze een expliciete betekenis hebben.

Matrix M is een **sparse** *(grotendeels lege)* matrix, daar de meeste gebruikers de meeste items niet beoordeeld hebben[36]. Veel ratings zullen niet expliciet gemaakt zijn, waardoor de matrix veel onbekenden bevat.

Impliciete en expliciete dataverzameling kunnen mogelijk **gecombineerd** worden om meer suggesties te genereren. Zo kan het lang bekijken van een eerste zoekresultaat *(impliciete dataverzameling)* impliceren dat het item perfect aansluit bij de zoekterm. Deze observatie kan samen met een goede beoordeling die de gebruiker achterlaat op het item *(expliciete dataverzameling)* gebruikt worden om een gewogen score van dat item te bepalen.

1.3.2 Matrix completion

Het schatten van beoordelingen voor gebruikers kan voor sommige lezers misschien vergelijkbaar lijken met matrix completion technieken *(het invullen van onbekende waardes in een onvolledige matrix)*. Echter is een recommendation algoritme **niet hetzelfde** als een matrix completion algoritme, hoewel de technieken vergelijkbaar zijn. Om aanbevelingen te kunnen genereren zijn enkel items met een hoge verwachtingswaarde interessant. Items die de gebruiker waarschijnlijk als laag zou beoordelen, zullen hopelijk niet getoond worden aan de gebruiker. Het is overbodig om de geschatte beoordelingen voor deze items te berekenen. In essentie moet dus niet de volledige matrix vervoleddigd worden, hetgene wel het doel is bij matrix completion. Om deze reden is matrix completion nog complexer en duurder dan recommendation algoritmes, daar bij recommendation algoritmes niet perse de volledige matrix gevuld moet worden. Natuurlijk moet er dan wel een manier zijn om te voorspellen of een schatting positief of negatief zal uitdraaien. In dit document wordt deze voorspelling overgeslagen om de begrijpbaarheid van de algoritmes te verbeteren.

---

[33] Boek: Recommender Systems Handbook
Francesco Ricci, Lior Rokach, Bracha Shapira & Paul B. Kantor, Springer - 2011
Recommender Systems Handbook paragraaf 4.4

[34] Blogpost: Snapchat's big redesign bashed in 83% of user reviews
Josh Constine - 12/01/2018
techcrunch.com/2018/01/11/snapchat-redesign-uninstall - Geraadpleegd op: 11/04/2018
Bezocht op:

[35] Artikel: Bad Customer Service Interactions More Likely to be Shared Than Good Ones
Marketing Charts - 15/04/2013
marketingcharts.com/digital-28628 - Geraadpleegd op: 09/05/2018

[36] Blog post: The Netflix Prize and Production Machine Learning Systems: An Insider Look
Loren Shure, MATLAB, MathWorks - 22/04/2015
blogs.mathworks.com/loren/2015/04/22/
the-netflix-prize-and-production-machine-learning-systems-an-insider-look - Geraadpleegd op: 09/05/2018

1.3.3 Similarity functie

Beoordelingen kunnen berekend worden door enerzijds feature vectoren met elkaar te vergelijken *(content-based)* of door een gewogen gemiddelde te nemen van neighbours items *(item-item collaborative filtering)* of gebruikers *(user-user collaborative filtering)*. Om twee items of twee gebruikers in verband te kunnen brengen met elkaar moet hun match bepaald worden. De **consensus** tussen twee items of twee gebruikersprofielen wordt bepaald aan de hand van een **score**. Hoe hoger deze score, hoe meer gelijkend de twee profielen zijn en omgekeerd. Indien er tussen twee profielen geen verband bestaat *(score niet gedefinieerd),* kan de neighbour niet gebruikt worden voor de rating bepaling.

Een **similarity functie** kan op diverse manieren gedefinieerd worden:

- Jaccard similarity: aantal gemeenschappelijk beoordeelde items zonder te kijken naar waardes
- Gemiddelde absolute afstand tussen beoordelingen
- Cosinus similarity: cosinus van de hoek tussen vectoren
- Centered cosinus similarity: Pearson correlatie coëfficiënt tussen vectoren
- Nog vele, vele anderen...

Er zijn heel veel soorten similarity functies, elk met hun voor- en nadelen. Hier worden een aantal mogelijke manieren besproken: Jaccard, Cosinus & Pearson similarity.

1.3.3.1 Jaccard similarity

Een eerste poging om gebruikers similarity te berekenen kan aan de hand van de **Jaccard similarity**[37] [38] gebeuren. De Jaccard similarity bekijkt het aantal **gemeenschappelijke ratings** over het aantal ratings dat ze samen maakten. De intuïtieve formule om de Jaccard similarity te berekenen is:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

A en B stellen hier de ratings vectoren voor. $|A \cap B|$ en |A ∪ B| stellen respectievelijk het aantal beoordelingen in de intersectie en unie van de gebruikers voor. Een user vector kan voorgesteld worden als een set van beoordelingen. Door deze intersectie en unie set operaties te gebruiken kan de Jaccard similarity bepaald worden.

Het bereik van deze functie ligt tussen 0 en 1 *(beide inclusief)*. Het resultaat kan nooit negatief worden, daar er altijd minstens 0 gemeenschappelijke beoordelingen zijn *(teller)*. Het resultaat kan ook nooit groter worden dan 1, daar de unie van beoordelingen altijd minstens zo groot is als de intersectie van beoordelingen; de noemer zal altijd groter dan of gelijk aan de teller zijn. Let op dat voor gebruikers die beide geen ratings hebben de Jaccard similarity niet gedefinieerd is, daar de unie *(noemer)* van beoordelingen 0 is. De standaardwaarde kan dan als 0 gedefinieerd worden. Indien beide sets leeg zijn wordt de Jaccard similarity als 1 gedefinieerd.

---

[37] Artikel: Jaccard similarity
Jan Schulz, Code 10 - 15/05/2008
code10.info/index.php?view=article&id=60:article_jaccard-similarity - Geraadpleegd op: 23/05/2018

[38] Artikel: Jaccard Index / Similarity Coefficient
Stephanie, Statistics How To - 02/12/2018
statisticshowto.com/jaccard-index - Geraadpleegd op: 23/05/2018

Om de similarity tussen twee gebruikers A en B te bepalen, wordt het aantal gemeenschappelijke ratings van A en B door het totaal aantal ratings dat beide gebruikers maakten gedeeld. Hiervan ontstaat de verhouding intersectie over unie. De intuïtie achter deze vergelijking is dat gebruikers die veel dezelfde items beoordelen *(waar J(A, B) kort bij 1 ligt)* een sterk overlappende smaak in items hebben. Gebruikers met een beperkte of geen overlappende beoordeling *(waar J(A, B) kort bij 0 ligt)* hebben volgens deze redenering geen gemeenschappelijke smaak.

Om gebruiker A items te suggereren kunnen nu alle items - exclusief de items die gebruiker A al beoordeeld heeft - die gebruiker B goed beoordeeld heeft aan gebruiker A gepresenteerd worden indien J(A, B) kort bij 1 ligt. Formeel wordt de verzameling items (**B \ A**) *(waar de items een positieve beoordeling hebben)* teruggegeven aan de gebruiker.

Dit zeer eenvoudig algoritme blijkt in de praktijk helaas niet zo goed te werken. Gegeven volgende voorbeeld rating matrix M *(let op dat in de praktijk de matrix heel sparse is, maar voor de eenvoud is de voorbeeld matrix hier goed gevuld)* waar gebruikersbeoordelingen binnen de schaal 1 t.e.m. 5 liggen.

	gebruiker A	gebruiker B	gebruiker C	gebruiker D
item I_1			**1**	**4**
item I_2	**5**	**1**	**4**	**2**
item I_3	**4**	**2**	**5**	**2**
item I_4	**1**	**5**		

Tabel: voorbeeld rating matrix

De gebruiker kan het volgende observeren: gebruikers A en C zijn gelijkend en gebruikers B en D zijn gelijkend. Let wel dat gebruikers A en C een tegenovergestelde smaak hebben ten opzichte van gebruikers B en D. Daarnaast zijn items I_2 en I_3 op elkaar gelijkend. Over de relatie tussen items I_1 en item I_4 kan nog momenteel niets gezegd worden, daar ze geen overlapping hebben in ratings.

Om voor gebruiker A de onbekende rating voor item I_1 te schatten, kan het gebruikersprofiel van gebruiker A vergeleken worden met de gebruikersprofielen van de andere gebruikers. In onderstaand voorbeeld worden gebruiker A en gebruiker B met elkaar in verband gebracht:

Gebruikers A en B hebben gemeenschappelijk items I_2, I_3 en I_4 beoordeeld. Samen hebben gebruikers A en B items I_1, I_2, I_3 en I_4 beoordeeld. Deze observaties geven 3 en 4 in de teller en noemer respectievelijk.

$$J(A,B) = \frac{3}{4} = 0.75$$

Intuïtief zou men er al niet van uitgaan dat gebruikers A en B in dit voorbeeld sterk overeenkomen. Ze hebben namelijk een tegenovergestelde smaak van items *(hun ratings zijn sterk uiteenlopend)*. De Jaccard similarity categoriseert deze twee gebruikers echter als 'sterk gelijkend' *(het resultaat van de functie 0.75 ligt dicht bij 1*). In dit voorbeeld zou gebruiker A goed beoordeelde items van B terugkrijgen, namelijk item I_1. De lezer kan zich voorstellen dat gebruiker A item I_1 waarschijnlijk niet interessant zal vinden.

Dit verschil is te verklaren daar de formule de magnitude van rating values volledig negeert, enkel de booleaanse waarheid of dat een item al dan niet beoordeeld is wordt in acht genomen *(of een rating positief of negatief is, heeft geen invloed)*. De lezer kan inzien dat het nodig is om de effectieve waardes van ratings mee op te nemen in de berekening. Uit deze redenering volgt dat de cosinus similarity meer aangewezen is voor deze berekening. Jaccard similarity wordt in de praktijk dan ook bijna niet gebruikt als similarity functie binnen recommendation algoritmes wanneer het rating interval niet binair is.

1.3.3.2 Cosinus similarity

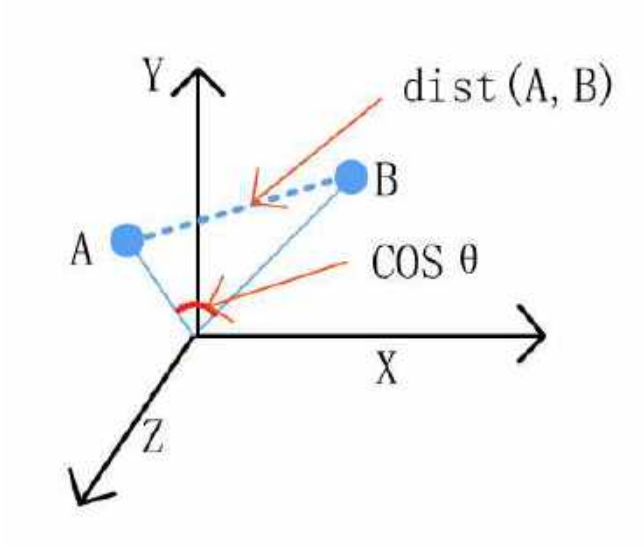
Stelt u zich een user vector *(de vector met alle beoordelingen van een gebruiker)* voor. Deze vector in n-dimensionale ruimte *(met n het maximale aantal beoordelingen en tevens het aantal items in de catalogus)* vertrekt vanuit de oorsprong. De user vector van een gebruiker A kan nu vergeleken worden met een user vector van een gebruiker B. Hoe sterk deze vectoren op elkaar gelijken kan bepaald worden aan de hand van de afstand tussen beiden. De afstandsbepaling kan gebeuren aan de hand van de cosinus similarity.

Een mogelijk beter alternatief voor de Jaccard similarity is de **cosinus similarity**. De cosinus similarity wordt afgeleid van het Euclidisch dotproduct *(n is de dimensie van vectoren A en B)*:

$$A.B = |A| \,.\, |B| \,.cos\theta \qquad \textit{(Euclidisch dotproduct)}$$

$$C(A,B) = cos(\theta) = \frac{A \cdot B}{|A| \,.\, |B|} = \frac{\sum_{i=1}^{n} A_i.B_i}{\sqrt{(\sum_{i=1}^{n} A_i^2)}.\sqrt{(\sum_{i=1}^{n} B_i^2)}}$$

De teller in de formule staat voor het product van vectoren A en B. De noemer staat voor de grootte van beide vectoren. De uitkomst van deze breuk is de cosinus van de hoek θ tussen deze twee vectoren. De cosinus geeft hier een indicatie van de onderlinge afstand tussen beide vectoren.



Figuur: afstand tussen twee vectoren

Bron figuur:
researchgate.net/publication/320914786_An_Opportunistic_Routing_for_Data_Forwarding_Based_on_Vehicle_Mobility_Association_in_Vehicular_Ad_Hoc_Networks

Om te zorgen dat de cosinus similarity overal gedefinieerd is, moeten onbekende waarden ingevuld worden met een standaard waarde. Kies hiervoor 0. Het idee achter de cosinus similarity methode komt van het idee dat vectoren in n-dimensionale ruimte dicht bij elkaar liggen indien de **hoek tussen** deze **vectoren** klein is. De cosinus van de hoek weerspiegelt de hoekgrootte. Een cosinus waarde van 1 geeft aan dat de vectoren parallel met elkaar lopen en de afstand dus eigenlijk 0 is. Een cosinus waarde van 0 geeft aan dat de vectoren loodrecht op elkaar staan en de afstand eigenlijk oneindig is. Een kleine afstand impliceert een goede match tussen vectoren, een grote afstand niet.

Het bereik van de cosinus similarity zal dus tussen -1 en 1 liggen waar een waarde die dicht bij 1 ligt aangeeft dat beide vectoren een kleine onderlinge afstand hebben in dezelfde richting. Deze vectoren matchen in de positieve zin. Vectoren die in tegenovergestelde zin wijzen zullen een cosinus similarity hebben die dicht bij -1 ligt. Wanneer vectoren loodrecht op elkaar staan hebben ze vrijwel een oneindige onderlinge afstand, zij genereren een cosinus similarity van 0. Een uitkomst van 0 duidt op geen correlatie tussen beide vectoren.

Om de cosinus similarity te illustreren maken we opnieuw gebruik van voorbeeld rating matrix M in paragraaf Jaccard similarity. Cosinus similarity toegepast op gebruikers A en B geeft:

$$C(A,B) = \frac{0 \times 0 + 5 \times 1 + 4 \times 2 + 1 \times 5}{\sqrt{(0^2 + 5^2 + 4^2 + 1^2)}.\sqrt{(0^2 + 1^2 + 2^2 + 5^2)}} = 0.52$$

Het resultaat van deze berekening benadert al beter onze intuïtie van de geschatte beoordeling, maar is nog steeds suboptimaal. 0.5 ligt centraal en duidt dus noch op similarity, noch op dissimilarity *(negatieve similarity)*. Toch als de lezer de beoordelingen van gebruikers A en B vergelijkt, kan men stellen dat deze twee gebruikers helemaal niet op elkaar gelijken. *(Hun beoordelingen zijn als het ware omgekeerden.)*

Dit verschil kan verklaard worden daar missende ratings als negatief beschouwd worden. Hoewel 0 buiten de ratingschaal ligt, heeft het in de schaal van [1, 5] toch een uiterst negatieve semantiek. Een onbekende beoordeling impliceert niet noodzakelijk een negatieve beoordeling. Het is niet omdat een gebruiker een item niet gekocht heeft dat de gebruiker het item niet leuk vindt. *(Een mogelijkheid is dat de gebruiker het product niet kent, nooit gevonden heeft of het momenteel niet nodig heeft. De gebruiker heeft daarom niet onmiddellijk een negatieve associatie met het product.)* Een verbetering van cosinus similarity moet rekening houden met ontbrekende beoordelingen.

Een verbetering van de cosinus similarity zou het invullen van een neutrale waarde voor onbekende ratings. In de rating schaal van [1, 5] heeft 3 een neutrale waarde. 0 vervangen door 3 zou een mogelijk beter resultaat kunnen geven. Echter houdt cosinus similarity geen rekening met het rating gedrag van gebruikers. Zo heeft een positieve rating van een easy rater *(een rater die meestal positief beoordeeld)* nu evenveel waarde als een positieve rating van een 'tough rater' *(een rater die meestal negatief beoordeeld)*. De lezer kan zich voorstellen dat een positieve rating voor de 'tough rater' meer waarde heeft dan voor de easy rater. Een betere similarity functie moet dus ook rekening houden met de user bias.

1.3.3.3 Centered cosinus similarity

De **centered cosinus similarity** methode **normaliseert** de ratings van een gebruiker. Door het gemiddelde van een gebruikers rating te gebruiken mag een 0 wel ingevuld worden voor onbekende waardes zonder daarmee een negatieve rating uit te drukken. De onbekende ratings van gebruiker X worden nu als gemiddelde ratings uitgedrukt. Reeds bestaande ratings worden gecentreerd rond 0. Eerst berekent men het rij-gemiddelde van alle bekende waardes *(sommeer de bekende ratings en deze som delen door het aantal bekende ratings)*.

Hierna wordt van elke bekende rating het rij-gemiddelde afgetrokken om een gecentreerde rating te verkrijgen. De magnitude *(absolute waarde)* van een beoordeling laat nu ook zien hoe sterk een item geliket wordt *(positieve genormaliseerde rating)* of hoe sterk een item niet geliket wordt *(negatieve genormaliseerde rating)*. Centered cosinus similarity houdt om deze reden ook rekening met **tough** en **easy raters** *(personen die alles heel negatief of alles heel positief beoordelen),* daar hun rating negatief of positief is ten opzichte van de gemiddelde van ratings van de gebruiker en niet ten opzichte van het middelpunt van de ratingschaal. Centered cosinus similarity wordt in de literatuur ook wel de **Pearson correlatie**[39] genoemd.

Wel moet men een kleine zijnoot opmerken; waar ratings voormalig misschien integer waardes hebben *(indien het rating interval dit impliceert)* is het nu mogelijk dat genormaliseerde ratings reële waardes zullen aannemen *(dit zou mogelijks een negatieve impact kunnen hebben op opslag van data en de snelheid van berekeningen hierop)*.

Het normaliseren van de vector elementen geeft niet altijd een goed resultaat. Wanneer een vector bestaat uit identieke elementen, dan zal de genormaliseerde vector bestaan uit enkel nul elementen en getransformeerd worden naar een punt, de oorsprong. Bijgevolg zal de similarity tussen twee gebruikers met identieke beoordelingen telkens nul zijn wat niet altijd de waarheid benaderd. Stelt u zich twee gebruikers voor die beiden altijd 3 sterren geven. De centered cosine similarity tussen deze twee gebruiker zal 0 bedragen. Intuïtief is echter duidelijk dat zij een perfecte match vormen. In dit randgeval kan best geopteerd worden om de vectoren niet te normaliseren en terug te vallen op reguliere cosine similarity.

Merk op dat cosinus similarity of Pearson correlatie zinloos is voor binaire ratings *(views, kliks, aankoop)*. Bij een binaire ratingschaal wordt beter geopteerd voor de Jaccard gelijkheid *(bij de Jaccard gelijkheid wordt enkel rekening gehouden met de booleaanse waarheid of een item al dan niet beoordeeld is)*. Daarnaast zal de geobserveerde of verwachte ratio die berekend wordt verbeteren met het aantal gebruikers. Tenslotte is de Jaccard similarity berekening ook veel goedkoper dan de *(centered)* cosinus similarity.

De centered cosinus similarity toegepast op de voorbeeld rating matrix uit de paragraaf over Jaccard similarity geeft:

Stap 1: Ratings normaliseren
De rating vector van gebruiker A is nu (_, 5, 4, 1) en de rating vector van gebruiker B is (_, 1, 2, 5). Het gemiddelde van vector A is $\frac{5+4+1}{3} = 3.33$ en van vector B is $\frac{1+2+5}{3} = 2.67$. Deze gemiddelden worden dan van de waardes afgetrokken zodat ze gecentreerd worden rond 0. *(Merk op dat de ratingschaal nu van -2.5 tot 2.5 loopt en dat hier het teken van de genormaliseerde rating bepaalt of een rating positief of negatief is. Rekening houdend met de overige ratings van een gebruiker en de magnitude van de beoordeling, bepaalt de waarde van de beoordeling nu hoe sterk positief of negatief de beoordeling van de gebruiker is ten opzichte van al zijn of haar ratings.)*

Voor onbekende waardes kan wel zomaar 0 ingevuld worden daar het binnen een genormaliseerde schaal een neutrale betekenis heeft voor de gebruiker. Vector A wordt dan: A' = (0, 5 - 3.33, 4 - 3.33, 1 - 3.33) of (0, 1.67, 0.67, -2.33). Vector B wordt getransformeerd tot: B' = (0, 1 - 2.67, 2 - 2.67, 5 - 2.67) of (0, -1.67, -0.67, 2.33).

[39] Artikel: Pearson Product-Moment Correlation
Lund Research Ltd - 2018
statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide.php
Geraadpleegd op: 05/05/2018

Stap 2: Cosinus similarity invullen

$$C(A,B) = \frac{0 \times 0 + 1.67 \times (-1.67) + 0.67 \times (-0.67) + (-2.33) \times 2.33}{\sqrt{(0^2 + 1.67^2 + 0.67^2 + (-2.33)^2)}.\sqrt{(0^2 + (-1.67)^2 + (-0.67)^2 + 2.33^2)}}$$

$$C(A,B) = -1.0$$

Het resultaat -1.0 is de ondergrens van het bereik van de centered cosinus similarity. Deze meest negatieve waarde geeft de indicatie dat de vectoren ver uit elkaar liggen en een tegenovergestelde zin hebben. Deze vector zijn zijn dus helemaal niet gelijkend. Deze uitkomst komt ook overeen met de intuïtie van de lezer daar de beoordelingen van deze gebruikers als het ware omgekeerden zijn binnen de ratingschaal *(hun onderlinge afstand is groot)*.

Het gevolg van deze uitkomst is dat de beoordeling van gebruiker B voor item I_1 beter niet opgenomen wordt in de berekening van de schatting van de beoordeling van gebruiker A voor item I_1 daar ze geen matchend profiel hebben.

Merk op dat er nog vele, vele andere similarity functies mogelijk zijn. Voor elk type dataset zal een categorie van similarity functies beter werken dan een ander. Elk hebben ze een eigen karakteristiek en zijn ze al dan niet geschikt voor een specifieke dataset. Similarity measures kunnen ook gecombineerd worden met elkaar en kunnen daarbij extra rekening houden met de context van aanbevelingen enz.

2. Soorten recommendation algoritmes

Veel recommender systemen zijn gestart als **nearest neighbor** recommendations. Tot op vandaag is dit de meest gebruikte methode. Deze industriestandaard zal dan ook verder gebruikt worden om de algemene werking van recommendation algoritmes te verduidelijken. Er zijn diverse groepen recommendation algoritmen om gebruikers suggesties te bieden: content-based systemen, collaborative systemen en knowledge-based systemen. Technisch gezien zijn handpicked items en simpele aggregaten ook recommendation systemen, maar hierop wordt in dit document niet verder ingegaan, daar deze triviaal in werking zijn.

Content-based systemen werken op informatie over de **inhoud** *(attributen)* van items gebaseerd op item profielen, de metadata *(eigenschappen)* van deze items. **Collaborative** filtering systemen werken op sociale informatie; de **interacties** *(ratings)* tussen gebruikers en items. In de praktijk zijn de resultaten van collaborative filtering systemen meestal beter. Een mogelijke uitleg hiervoor is dat attributen niet allesbepalend zijn voor items en dat gebruikers niet altijd items met bepaalde attributen appreciëren.

Formeel gaat een recommendation systeem een matrix M met ratings *(waar rijen de gebruikers zijn en kolommen de items zijn)* waar vaak meer dan 99% van de ratings onbekend zijn, onbekende ratings proberen te schatten. *(Liefst ook enkel de hoge geschatte ratings.)*

2.1 Content-based systemen

Een content-based recommendation algoritme gaat **items** aan gebruiker X aanbevelen die **gelijkend** zijn op items die gebruiker X voorheen een **hoge rating** gegeven heeft en die matchen met de gebruiker zijn of haar gebruikersvoorkeuren. Op basis van gebruiker X zijn of haar ratings en de attributen van zijn of haar beoordeelde items kan een gebruikersprofiel gegenereerd worden. Ook elk item in de catalogus heeft een item profiel met een verzameling van features. Voor elke feature bestaat een bijhorende vector van booleans of reële numerieke waardes. Natuurlijk moet er voor elke persoon en voor elk item wel een profiel gegenereerd worden. Dit kan op twee manieren: impliciet of expliciet.

2.1.1 Item profiel opstellen

Om een item profiel op te stellen moeten **attributen** voor het item **gevonden** worden. Voor films kunnen dit de regisseur, het genre, de gesproken taal, jaar van publicatie, de acteurs enz. zijn. Evenementen kunnen gecategoriseerd worden op locatie, organisator, type *(fuif, festival, concert, studentenfuif, chrysostomos enz.*), datum, muziekstijl, DJ's enz. Niet voor alle soorten objecten is het even triviaal om attributen te definiëren. Een voorbeeld van niet-triviale objecten zijn teksten. Een mogelijke manier om teksten te categoriseren is door belangrijke woorden op te nemen in het item profiel. Belangrijke woorden kunnen in een tekst gedefinieerd worden aan de hand van **TF-IDF**[40] *(term frequency x inverse doc frequency)*. Deze statistische berekening bepaalt hoe belangrijk een woord is door middel van het aantal voorkomens van dat woord in een tekst tegenover het voorkomen van dat woord in alle andere teksten af te wegen. Het categoriseren van afbeeldingen, muziek en video's gebeurt tevens op speciale manieren. Features toekennen aan media objecten wordt in deze cursus niet besproken.

Features toekennen aan een item kan impliciet of expliciet gebeuren. Expliciet kunnen features opgegeven of geëxtraheerd worden uit het item. Impliciet kunnen item attributen afgeleid worden van gebruikersbeoordelingen en gebruikersprofielen. Hierover meer in Gebruikersprofiel opstellen.

[40] tfidf.com

2.1.1.1 TF-IDF

TF-IDF bestaat uit twee termen: **Term Frequency** (*TF*) en **Inverse Document Frequency** (*IDF*) *(vandaar de naam)*. TF meet hoe frequent een woord voorkomt in een document rekening houdend met de lengte van of het aantal woorden in dat document *(daar het mogelijk is dat een woord vaker zou voorkomen in een langer document)*. IDF meet dan weer hoe belangrijk de term is ten opzichte van de andere documenten in de catalogus. Lidwoorden *(de, het, een)* komen vaak voor in elk document en zouden zonder de IDF factor heel belangrijk zijn in elk document. *(Veel voorkomens van een lidwoord groeit lineair mee met de lengte van de tekst en zorgt zo voor de hoge TF waarde.)* Door het aantal verschijningen van een woord af te wegen tegen het aantal voorkomens van dat woord in alle andere documenten, wordt de belangrijkheid van dat woord bepaald. Door deze tweede term *(IDF)* te gebruiken worden woorden die weinig voorkomen belangrijker dan woorden die veel voorkomen in alle teksten. *(Lidwoorden komen veel voor in alle teksten.)* Het product van deze twee termen geeft TF-IDF. De formule moet telken beschouwd worden in de context van alle andere teksten in de catalogus.

$$TF\text{-}IDF(W, D) = \left(\frac{aantal\ voorkomens\ W\ in\ D}{totaal\ aantal\ woorden\ in\ D} \times\right) \log\left(\frac{totaal\ aantal\ documenten}{aantal\ documenten\ met\ W}\right)$$

In de tweede term is het opvallend dat de log functie gebruikt worden. Het resultaat is is dat de relevantie van een term in een tekst niet lineair meegroeit met het aantal voorkomens van dat woord in de tekst[41]. Dit komt doordat de logfunctie een sub-lineaire functie is *(een functie die trager groeit dan de lineaire functie)*. De intuïtieve reden is dat grote uitkomsten gedempt worden en om relatief grote verschillend tussen TF-IDF waardes te verkrijgen. Vaak wordt ook de log functie opgeteld bij 1 om zo een onderscheid te kunnen maken tussen wanneer TF 0 is en wanneer TF niet 0 is *(log van 1 is 0)*.

In de context van een evenementen aanbevelings engine zouden woorden de beschrijving van een evenement gecategoriseerd kunnen worden aan de hand van TF-IDF zoals geïllustreerd in volgend voorbeeld:

Veronderstel twee items: item A heeft een beschrijving van 200 woorden en bevat vier maal de term 'TD' *(wat de andere woorden in de beschrijving zijn, doet er niet toe)*. Item B heeft een langere beschrijving van 500 woorden en bevat vijf maal de term 'TD' in de beschrijving. De catalogus van evenementen bevat 2.000.000 elementen waarvan 1% *(20.000 items)* de term 'TD' bevat. IDF is voor beide items gelijk, namelijk: $log(2.000.000 / 20.000) = 2$ *(logaritme met basis 10)*. De IDF waarde zou hergebruikt kunnen worden en moet niet voor elk item afzonderlijk berekend worden. TF moet wel voor elk woord apart berekend worden. TF is voor beide evenementen verschillend:

$$TF('TD', A) = \frac{4}{200} = 0.02 \Rightarrow TF - IDF('TD', A) = 0.02 \times 2 = 0.04$$

$$TF('TD', A) = \frac{5}{500} = 0.01 \Rightarrow TF - IDF('TD', A) = 0.01 \times 2 = 0.02$$

Item A heeft in dit voorbeeld de hoogste TF-IDF score en is daarmee een relevanter item voor zoekterm 'TD' en zal daarmee eerder weergeven in de zoekresultaten of suggesties. *(De zoekresultaten kunnen bijvoorbeeld aflopend gesorteerd worden op TF-IDF waarde.)* Indien er meerdere zoektermen zijn kan een gewogen gemiddelde genomen worden van de respectievelijke TF-IDF waardes voor elke zoekterm.

[41] Vraag: Understanding the use of logarithms in the TF-IDF logarithm
stats.stackexchange.com/questions/161640/understanding-the-use-of-logarithms-in-the-tf-idf-logarithm
Geraadpleegd op: 30/05/2018

2.1.2 Gebruikersprofiel opstellen

Een gebruikersprofiel kan tevens impliciet of expliciet bepaald worden, afhankelijk van hoe dat de feature vectoren in het gebruikersprofiel berekend werden.

Een gebruikersprofiel wordt **impliciet** opgesteld op basis van **item profielen** van **beoordeelde items** uit de catalogus. Stel de beoordeelde items voor als I_1, I_2, ..., I_n. Deze methode is heel eenvoudig; door het gewogen gemiddelde van elk beoordeelde item *(magnitude van ratings maal item vector)* te gebruiken wordt een realistisch resultaat bereikt. Door deze gewogen gemiddelden te normaliseren rond 0 kunnen ook 'negatieve ratings' *(ratings onder het gemiddelde)* gebruikt worden. Deze baseline van een rating r van gebruiker X ten opzichte van het gemiddelde van ratings van gebruiker X geeft een indicatie of rating r hoger of lager dan de overige ratings van gebruiker X.

Laten we de voorbeeld dataset van paragraaf Jaccard similarity als fictief voorbeeld gebruiken. Voor dit voorbeeld gaan we item attributen toekennen aan items I_1, I_2, I_3 en I_4. Een attribuut kan hier een reële waarde tussen 0 en 1 aannemen. Hieronder het overzicht van de attributen voor deze items:

Item	Attribuut Y	Attribuut Z
item I_1	**0.15**	**0.95**
item I_2	**0.95**	**0.05**
item I_3	**1.00**	**0.10**
item I_4	**0.10**	**0.90**

Tabel: voorbeeld verzameling item attributen

Vanuit deze item features *(Y en Z)* kunnen profiel vectoren voor de gebruikers *(impliciet)* gegenereerd worden. Concreet kunnen aan gebruikers A, B, C en D ook de attributen Y en Z toegekend worden. Profiel vectoren bevatten *(een deel van)* de item attributen. Deze profiel features kunnen bijvoorbeeld met volgende formule berekend worden waar n het aantal bekende beoordelingen van een gebruiker X is. I is de waarde van het item attribuut, r_{XI} is de rating van gebruiker X voor item I. De 5 in de formule staat voor de maximale rating waarde in het rating interval. Het rating interval is [1, 5]. Hieronder de formule om feature F voor gebruiker X te bepalen:

$$U_{XF} = \frac{\sum_{i=1}^{n} r_{Xi} \times I_i}{n \times 5}$$

Het gebruikersprofiel van gebruiker A kan als volgt berekend worden:

$$U_{AY} = \frac{5 \times 0.95 + 4 \times 1.00 + 1 \times 0.10}{3 \times 5} = 0.61$$

$$U_{AZ} = \frac{5 \times 0.05 + 4 \times 0.10 + 1 \times 0.90}{3 \times 5} = 0.10$$

Het gebruikersprofiel voor gebruiker A in dit voorbeeld is (0.61, 0.10).

Een gebruikersprofiel kan ook **expliciet** bepaald worden. Hier kan de gebruiker voor zijn profiel **selecteren** welke **attributen** hij of zij interessant vindt. Deze features zullen dan gebruikt worden om te matchen met de items. Het is mogelijk dat deze expliciet bepaalde features niet overeenstemmen met de reeds gemaakte beoordelingen van de gebruiker. Dit probleem illustreert dat het heel belangrijk is om goede features te selecteren voor items en dat deze niet te ruim gekozen mogen worden. Het attribuut horror kan bijvoorbeeld nog eens onderverdeeld worden in thriller, fictie, detective enz. Het attribuut studentenfuif kan dan weer verder opgedeeld worden in TD, chrysostomos, fakbar, lustrum enz. Ook zijn niet alle features even relevant voor de perceptie van de gebruiker voor een item. Zo is het aantal beats per minuut niet noodzakelijk relevant voor de categoriebepaling van een nummer *(tenzij dan voor een DJ)*. Daarnaast zal de afspeelduur van een speelfilm weinig invloed hebben op de voorkeur van de kijker.

Dat de match tussen de features van een beoordeeld item en het gebruikersprofiel kan afwijken van de gemaakte beoordelingen, geeft een indicatie waarom content-based recommendation algoritmes in de praktijk minder accuraat zijn dan bijvoorbeeld collaborative methodes.

2.1.3 Items aanbevelen aan gebruikers

Om nu een schatting te maken van een rating worden paren genomen van een gebruikersprofiel en een **item profiel**. Hoe beter de **match** tussen het **gebruikersprofiel** en het item profiel, des te hoger de **geschatte rating** zal zijn. Beide vectoren *(profielen van attributen)* bevinden zich in een hoog dimensionale ruimte *(meerdere attributen om een profiel te modelleren)*. Hoe sterk beide vectoren matchen kan men bepalen aan de hand van een afstandsbepaling tussen de twee vectoren. In het hoofdstuk Similarity functie zagen we al eerder diverse similarity measures die hiervoor gebruikt kunnen worden.

Om de methode om items aan te bevelen te illustreren gebruiken we in het volgende voorbeeld de cosinus similarity[42]. Door de hoek tussen twee vectoren te berekenen krijgt men een inzicht in de hun onderlinge afstand[43].

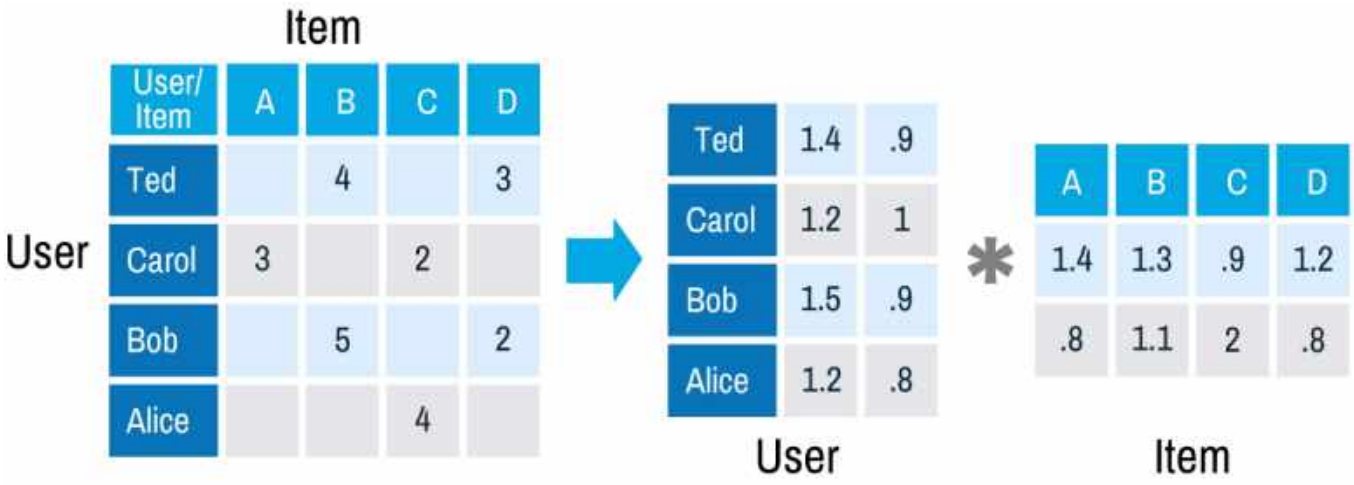
$$C(A,B) = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{(\sum_{i=1}^{n} A_i^2)} \times \sqrt{(\sum_{i=1}^{n} B_i^2)}}$$

Wel moet men opletten dat deze berekening niet gedefinieerd is bij undefined of unobserved waarden in de vectoren. Deze waarden zullen met een standaardwaarde *(rating)* ingevuld moeten worden *(bv. 0)* of genegeerd moeten worden. Om concreet de beste aanbevelingen voor een gebruiker X te bepalen wordt de cosinus similarity tussen de feature vector van gebruikers profiel X met de feature vector van elk item I_1 berekend *(van de intersectie van gemeenschappelijke ratings)*. De K producten met de hoogste cosinus waarde *(items en gebruikersprofielen met de kleinste hoek)* zijn de items die aanbevolen kunnen worden. Merk op dat het bereik van de cosinus similarity functie binnen het interval [0, 1] inclusief ligt.

---

[42] Blog post: Euclidean vs. Cosine Distance
Chris Emmery Tilburg University - 25/03/2017
cmry.github.io/notes/euclidean-v-cosine - Geraadpleegd op: 05/04/2018

[43] Blog post: Machine Learning :: Cosine Similarity for Vector Space Models (Part III)
Christian S. Perone Machine Learning Engineer Montreal - 12/09/2013
blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii
Geraadpleegd op: 05/04/2018



Figuur: werkwijze items aanbevelen

Bron figuur: mapr.com/ebooks/spark/08-recommendation-engine-spark.html

Hieronder wordt de rating berekend die gebruiker A zou toekennen aan item I_1 op basis van het item profiel en het gebruikersprofiel, gebruikmakende van de voorbeeld dataset in Jaccard similarity. Gebruikmakende van de cosinus similarity *(een andere similarity functie gebruiken zou uiteraard ook mogelijk zijn)* krijgt men volgende similarity score:

$$C(A, I_1) = \frac{0.15 \times 0.61 + 0.95 \times 0.10}{\sqrt{(0.15^2 + 0.95^2)} \times \sqrt{(0.61^2 + 0.10^2)}} = 0.12$$

Om tot de effectieve waardering voor item I_1 te komen kan bijvoorbeeld de similarity score vermenigvuldigd worden met de gemiddelde rating die gebruiker A geeft, waarvan samen met de gemiddelde rating van item I_1 een gemiddelde genomen kan worden:

$$r_{XI} = \frac{meanRating(X) \times similarity(X, I) + meanRating(I) \times (1 - similarity(X, I))}{2}$$

Een variant hiervan is het gewogen gemiddelde nemen van de match maal de gemiddelde gebruikersbeoordeling en de gemiddelde item beoordeling:

$$r_{XI} = \frac{similarity(X, I) \times meanRating(I) + meanRating(X)}{similarity(X, I) + 1}$$

Een meer eenvoudige methode is om de similarity score tussen het item de de gebruiker te schalen naar het interval [0-1] en om deze factor te vermenigvuldigen met de maximale score. In de voorbeeld dataset die we hier gebruiken is het rating interval [1, 5]. De minimale en maximale score zijn respectievelijk 1 en 5. De Jaccard similarity drukt de similarity uit in het interval [0, 1]. Om dit te schalen van 1 tot 5 sterren kan volgende formule gebruikt worden:

$$r_{XI} = J(X, I) \times 4 + 1$$

De cosinus of centered cosinus similarity drukt de similarity score uit in het interval [-1, 1] en moet voor deze berekening herschaald worden naar [1, 5]. Via volgende formule kan de *(centered)* cosinus similarity gebruikt worden.

$$r_{XI} = C(X, I) \times 2 + 1$$

Bij deze laatste gaat men ervan uit dat een item dat 100% overeenkomt met het gebruikersprofiel de maximale score zou krijgen van de gebruiker en een item dat helemaal niet overeenkomt met het gebruikersprofiel *(0% match)* de minimale score zou ontvangen.

Deze laatste formule wordt geïllustreerd door de beoordeling van gebruiker A voor item I_1 te bepalen.

$$r_{XI} = 0.12 \times 2 + 1 = 1.24$$

Volgens deze content-based methode zal gebruiker A aan item I1 een score van 1.24 toekennen. Dit is een lage score en het item zal dus niet aan gebruiker A aanbevolen worden. De lezer kan zich intuïtief voorstellen dat item 1 mogelijk niet interessant is voor gebruiker A. (Met de cosinus similarity methode bij collaborative modellen is de geschatte rating 1.00, wat nogmaals illustreert dat content-based en collaborative systemen een heel verschillende werking en accuraatheid hebben.)

Uiteraard zijn er nog vele andere methoden mogelijk. Alles hangt af van de creativiteit van de ontwikkelaar en het domein van de items.

2.1.4 Complexiteit content-based filtering

Bij **content-based** methoden moeten er geen neighbours gezocht worden. Er moet enkel een gebruikersprofiel gegenereerd worden waarna de gebruikersprofielen met item profielen vergeleken kunnen worden. Wanneer zowel de gebruikersprofielen als item profielen gekend zijn, is de complexiteit van een content-based methode **O(**$U \times I$**)**. De complexiteit van een recommendation algoritme wordt grafisch geïllustreerd in het hoofdstuk Implementatie.

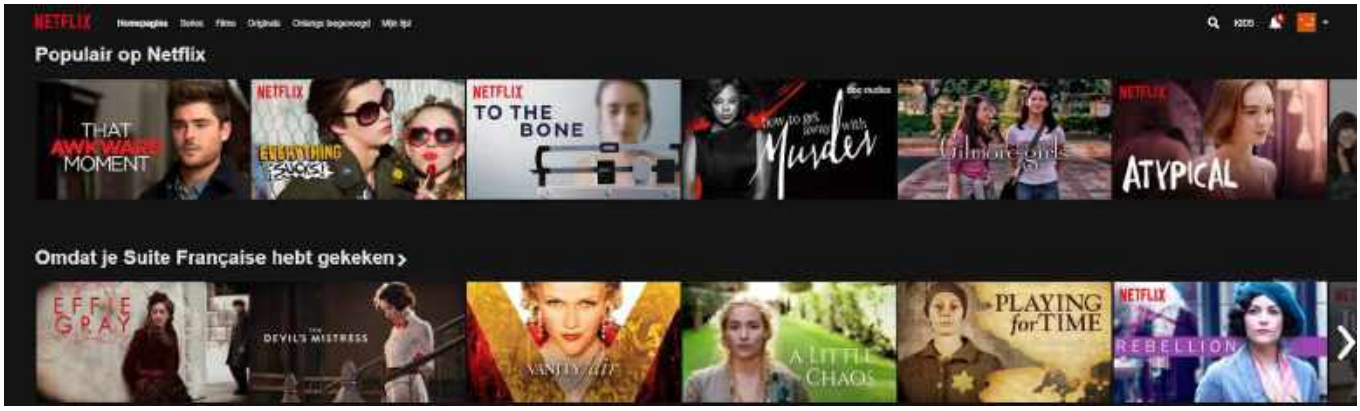
2.1.5 Voor- en nadelen content-based filtering

De content-based methode heeft een aantal voordelen. Ten eerste is er **geen data** nodig van **andere gebruikers,** daar gebruikers aanbevelingen individueel berekend kunnen worden zonder interpolatie met andere gebruikers. Om deze reden is dergelijk algoritme interessant voor platformen waar het **aantal gebruikers klein** is *(in een kleine verzameling gebruikers is het moeilijk om een ideale match te vinden tussen gebruikersprofielen)*.

Een bijkomend voordeel is dat dit systeem ook goed werkt indien gebruikers een heel **unieke smaak** hebben. *(Wanneer gebruikers een heel unieke smaak hebben en dus zeer uiteenlopende gebruikersprofielen hebben werkt een collaboratieve aanpak minder goed.)*

Ten derde kunnen **nieuwe items** en **niet-populaire items** *(in de long tail)* ook aanbevolen worden aan gebruikers, daar deze items niet in verband gebracht moeten worden met andere *(wel populaire)* items, het zogenaamde **first-rater probleem**.

Tenslotte is het met de cosinus similarity methode relatief eenvoudig om voor elke suggestie te informeren **waarom** dat item juist **aanbevolen** wordt. Netflix geeft weer waarom items aanbevolen worden. In het voorbeeld hieronder worden series en films aanbevolen die overeenstemmen met Suite Française. *(Overeenstemmen kan hier betekenen: films met dezelfde eigenschappen als Suite Française of gebruikers die Suite Française ook gekeken en goed beoordeeld hebben.)*



Figuur: Netflix aanbevelingen

Content-based methoden hebben helaas ook enkele nadelen. Een eerste probleem stelt zich bij het **definiëren van features**. Niet voor elk type items is het even triviaal om features te identificeren. Voor een film kunnen bijvoorbeeld de regisseur, de acteurs en het genre als attributen gebruikt worden. De inhoud van de verhaallijn is moeilijker te modelleren. Hetzelfde probleem geldt voor afbeeldingen en muziek. Daarnaast is het ook mogelijk dat een gebruiker niet van een bepaalde categorie items *(bv. een genre)* houdt maar toevallig enkel items uit die categorie beoordeeld heeft *(te **ruime features**)*. Het is niet omdat een film een actiefilm is dat een gebruiker met een voorkeur voor actiefilm dit item automatisch een goede film zal vinden. De lezer kan hier opmerken dat meerdere attributen noodzakelijk zijn om een goede voorspelling te maken.

Dit probleem kan nog verder doorgetrokken worden naar een probleem van **overspecialisatie**. De mogelijkheid bestaat dat gebruikers geen aanbevelingen krijgen die buiten hun gebruikersprofiel vallen. Personen hebben mogelijk meerdere interesses maar hebben die nog niet duidelijk gemaakt door items in deze interessevelden te raten. Deze beperking zorgt voor een gebrek aan **diversiteit** tussen de aanbevelingen en zorgen dat de gebruiker geen nieuwe categorieën kan ontdekken *(het probleem voedt zichzelf als het ware)*.

Tenslotte ontstaat een **'cold start'** problematiek voor nieuwe gebruikers. Zij hebben nog geen gebruikersprofiel *(daar ze nog geen items beoordeeld hebben in het verleden)* en kunnen daardoor niet met items gematcht worden. Wel kan een initieel gebruikersprofiel opgesteld worden door de gebruiker expliciet enkele interessevelden te specificeren, globale *(generieke)* populaire items aan te bevelen *(een gemiddeld profiel construeren op basis van het systeem gemiddelde profiel)* of door rekening te houden met de context van de gebruiker *(het gemiddelde bekijken van gebruiker uit dezelfde demografische groep enz.)*.

2.2 Collaborative systemen

Collaborative filtering is de oudste en meest gebruikte methode om aanbevelingen te genereren. Er zijn twee soorten groepen collaborative filtering systemen; **user-user** relaties en **item-item** relaties. Collaborative filtering is een relatief eenvoudige manier om recommandations te bepalen. Het algoritme is wel vrij krachtig *(vele aanbevelingen kunnen gevonden worden)*, kan rekening houden met de baseline van gebruikers en items en heeft een hele grote userbase, daar dit vrijwel de industriestandaard is.

Collaborative algoritmes kunnen via twee wegen tot aanbevelingen komen. Enerzijds via een **user-user** aanpak en anderzijds via een **item-item** methode. Via een user-user methodiek wordt een gebruiker in verband gebracht met zijn peer gebruikers. Analoog wordt in een item-item werkwijze een item gekoppeld aan een groep vergelijkbare items. Beide manieren zijn een nearest neighbourhood methode om suggesties te genereren.

Merk op dat collaborative systemen helemaal niet kijken naar de features van item- en gebruikersprofielen maar enkel rekening houden met **peer groepen** van items en gebruikers, de neighbours.

2.2.1 User-user relatie

Een user-user aanpak werkt in grote lijnen als volgt: om gebruiker X items aan te bevelen wordt een **groep gebruikers** exclusief gebruiker X gezocht die dezelfde items liken en disliken als *(of gebruikers in de buurt van)* de gegeven gebruiker X. Om deze groep gebruikers te vinden wordt een similarity functie gedefinieerd op de user-user relatie, welke in essentie de **K-nearest neighbours** zoekt van gebruiker X op basis van gelijkaardige **gebruikersprofielen**.

In essentie wordt het gebruikersprofiel van gebruiker X met de gebruikersprofielen van alle andere gebruikers vergeleken. Gebruikersprofielen met de beste match met gebruiker X vormen de top K-nearest neighbours.

2.2.2 Item-item relatie

Een item-item aanpak werkt in grote lijnen als volgt: voor een gebruiker X worden zijn beoordeelde items overlopen en voor elk item worden vergelijkbare items gezocht *(exclusief de reeds beoordeelde items)* welke aanbevolen kunnen worden aan gebruiker X. Om deze gerelateerde items te vinden wordt een similarity functie gedefinieerd op de item-item relatie welke in essentie de **K-nearest neighbours** zoekt van een item I op basis van gelijkaardige **item profielen**.

De item-item methode werkt in feite analoog met de user-user manier. In de praktijk werkt de item-item werkwijze minder goed dan de user-user methode. Dit wordt geïllustreerd op het einde van dit document. Merk op dat wanneer het aantal gebruikers groter is dan het aantal items in de rating matrix de item-item methode over het algemeen een minder duur algoritme is. Deze conclusie volgt uit onderstaande redenering:

Stel het aantal items voor als I en het totaal aantal gebruikers als U. Bij de user-user methode zal de complexiteit U x U x I zijn en bij de item-item methode is dat I x I x U.

2.2.3 Items aanbevelen

Nu dat het voor de lezer bekend is hoe neighbours voor zowel items als gebruikers gevonden kunnen worden, rest nog de methodiek om items aan te bevelen voor gebruiker. Een geschatte rating wordt berekend aan de hand van de similarity met de peer groep *(zijnde andere items of gebruikers)* en aan de hand van de ratings op die items of de ratings van die gebruikers kan de uiteindelijke beoordeling geschat worden. Er zijn twee manieren om neighbours te zoeken: user-user filtering en item-item filtering.

2.2.3.1 User-user filtering

De user-user filtering methode wordt vaak gebruikt wanneer items vaak wijzigen of wanneer er minder items zijn dan gebruikers (*daar in een grote verzameling gebruikers het eenvoudiger is om gebruikers te matchen dan wanneer er weinig items zijn en er items gematched moeten worden*). Voor **alle gebruikers** worden **alle items** in de catalogus afgegaan. Om het item **I** aan te bevelen aan gebruiker X worden volgende stappen ondernomen:

- De ratings van gebruiker X worden voorgesteld in een rating vector r_X.
- Laat N de verzameling zijn van de gebruikers *(exclusief gebruiker X)* die het meest lijken op gebruiker X *(neighbours)* en ook het item I een beoordeling gegeven hebben. *(Zij moeten ook het item I beoordeeld hebben daar we hun beoordelingen en similarity met gebruiker X zullen gebruiken om een rating voor gebruiker X te schatten voor item I.)* De verzameling N wordt gevonden door de rating vector r_X te vergelijken met de rating vectors van alle andere gebruikers. Meestal worden enkel neighbours die positief matchen met r_X gebruikt om de score te bepalen.
- Combineer de similarity tussen gebruiker X en de peers in N om de score voor item I te bepalen.

Nu zijn er twee mogelijke pistes. Enerzijds kan het gemiddelde van beoordelingen van gebruiker X zijn buren eenvoudig berekend worden door het gemiddelde te nemen van de ratings van de neighbours. De variabele n geeft hier het aantal neighbours weer. Een beoordeling van gebruiker X voor item I kan met volgend formule geschat worden:

$$r_{XI} = \frac{\sum_{y=1}^{n} r_{yI}}{n}$$

N vormt hier het aantal gebruikers die een beoordeling voor item I achtergelaten hebben. Hier zijn de ratings van gebruikers met een identieke smaak als gebruiker X even belangrijk als gebruikers met een matige similarity met gebruiker X. *(De lezer kan zich voorstellen dat gebruikers met een sterk overeenstemmende smaak met gebruiker X relevanter zijn voor de berekening van de geschatte beoordeling dan gebruikers met een minder of niet overeenstemmende smaak.)* Een andere piste - tevens een verbetering van de eerste methode - die gevolgd kan worden is rekening houden met hoe sterk neighbours overeenkomen met gebruiker X *(door gebruik te maken van een similarity functie zoals eerder besproken in Similarity functie)*. Deze formule zorgt ervoor dat gebruikers die meer gemeenschappelijk hebben met gebruiker X, meer doorslaggevend zijn in de berekening van de geschatte rating voor gebruiker X dan gebruikers die minder gemeenschappelijk hebben met gebruiker X. Een beoordeling van gebruiker X voor item I kan als volgt berekend worden:

$$r_{XI} = \frac{\sum_{y=1}^{n} r_{yI} \times similarity(X,y)}{\sum_{y=1}^{n} similarity(X,y)}$$

Deze tweede optie leunt dichter aan bij onze intuïtie, daar gebruikers met een sterk overeenkomende smaak meer gemeenschappelijk zullen hebben dan gebruikers met maar een beperkte overlap. De teller berekent een som van gewogen ratings over alle gebruikers die item I beoordeeld hebben, rekening houdend met de gegeven score. De noemer herleidt de bovenstaande som naar een gemiddelde voor alle gebruikers.

Laten we centered cosinus similarity toepassen op de voorbeeld dataset in het hoofdstuk over Jaccard similarity. De bedoeling is om de rating die gebruiker A zou geven aan item I_1 te schatten. Analoog met de centered cosinus similarity berekening uit paragraaf Centered cosinus similarity wordt het gewicht van alle buren gebruikt. Dit gewicht is nodig om het gewogen gemiddelde te kunnen nemen van de beoordelingen van gebruiker X zijn peers. In deze formule kan men ervoor kiezen om als beoordelingen ofwel de effectieve beoordeling of de genormaliseerde beoordeling te gebruiken. Bij deze laatste moet men er dan wel van uitgaan dat het resultaat tevens een genormaliseerde rating zal zijn voor gebruikers en terug herleid moet worden.

$$C(A, B) = -1.0$$
$$C(A, C) = 0.26$$
$$C(A, D) = -0.32$$

Gebruikmakende van de tweede formule eerder in deze paragraaf krijgt men:

$$r_{AI1} = \frac{0.26 \times 1}{0.26} = 1$$

De geschatte rating voor item I_1 van gebruiker A wordt in dit voorbeeld enkel bepaald door buur gebruiker C. Met andere woorden zal gebruiker A het item I_1 negatief beoordelen, bijgevolg is het niet interessant om item I_1 aan te bevelen aan gebruiker A. Men zou dit resultaat verwachten.

Merk op dat gebruiker B niet gebruikt wordt in deze formule, daar gebruiker B item I_1 geen beoordeling gegeven heeft. Daarnaast wordt ook gebruiker D niet gebruikt, daar gebruiker D een negatieve samenhang heeft met gebruiker A en niet gebruikt kan worden omdat ze een verschillende smaak hebben.

Natuurlijk bestaan er ook varianten waar de maker er wel voor geopteerd heeft om negatieve similarity mee te nemen in de berekening. Op het einde van dit document wordt deze parameter gebruikt om de verschillen aan te tonen in de implementatie.

2.2.3.2 Item-item filtering

Item-item collaborative filtering is de oudste collaborative filtering methode. In tegenstelling tot user-user collaborative filtering waar voor alle gebruikers gelijkaardige gebruikers gezocht worden, zal item-item collaborative filtering over alle items gaan om gelijke items te zoeken. Deze methode wordt gebruikt wanneer er weinig gebruikers zijn *(gebruikersprofielen matchen is moeilijk in een beperkte verzameling gebruikers)*. Dezelfde similarity metrics en prediction functie als in user-user collaborative filtering worden ook hier gebruikt. De geschatte rating van item I wordt nu gebaseerd op de ratings van gelijkaardige items. Het zal de lezer opvallen dat item-item filtering vrij analoog werkt met user-user filtering.

Om een item I aan te bevelen aan gebruiker X wordt de neighbourhood N berekend; de verzameling van items beoordeeld door gebruiker X die gelijkaardig zijn aan *(maar exclusief)* item I.

Een formule om de rating door gebruiker X op item I te schatten kan een gewogen gemiddelde van alle gebruiker ratings voor item I zijn, rekening houdend met de similarity tussen items I en zijn neighbours.

$$r_{XI} = \frac{\sum_{j=1}^{n} r_{Ij} \times similarity(I, j)}{\sum_{j=1}^{n} similarity(j, I)}$$

De geschatte rating van gebruiker X van item I wordt hier geschat door te sommeren over alle gewogen ratings op items rekening houdend met de similarity van deze items waarna het gemiddelde genomen wordt over alle ratings heen.

Volledig analoog met het voorbeeld eerder in **user-user filtering** wordt aan de hand van hetzelfde voorbeeld de geschatte beoordelingen voor item I_1 bepaald.

In theorie zijn item-item filtering en user-user filtering duale aanpakken en zouden ze eenzelfde theoretische performance moeten hebben. In de praktijk blijkt echter dat de user-user collaborative filtering methode de item-item collaborative filtering aanpak in de meeste gevallen sterk outperformed. Dit kan verklaard worden door het feit dat items eenvoudigere modellen zijn dan gebruikers. Gebruikers kunnen op meerdere manieren met elkaar in verband gebracht worden.

2.2.4 Complexiteit content-based filtering

Stel het aantal items in de catalogus voor als I en het aantal gebruikers in de rating matrix als U. Het is duidelijk dat de dimensie van rating matrix M $U \times I$ is.

In een **collaboratieve** aanpak is de meest complexe computationele stap het zoeken van de K meest gelijkende gebruikers of items. Deze stap heeft complexiteit O(U) of O(I) afhankelijk of dat de user-user filtering of de item-item filetering gebruikt wordt. Het zoeken van neighbours moet voor elk item of voor elke gebruiker gebeuren wat een kwadratisch resultaat geeft, **O(** U^2I **)** of **O(** UI^2 **)**. Omwille van deze reden is een collaborative aanpak duurder dan een content-based werkwijze. De complexiteit van een recommendation algoritme wordt grafisch geïllustreerd in het hoofdstuk Implementatie.

Meer optimale methoden zijn: near-neighbour search in high dimensions LSH waar hashing wordt toegepast, clustering en dimensionality reduction *(matrix factorization)*. De werking van deze methodieken wordt in deze paper niet besproken.

2.2.5 Voor- en nadelen content-based filtering

Het belangrijkste voordeel om collaborative filtering te gebruiken is dat het voor **elk type item** in de catalogus zou werken. Het is niet nodig om voor alle items features te bepalen ***(feature selection fase)***. Dit is goed voor items waar features of attributen moeilijk te bepalen zijn zoals video's, foto's en muziek. Daarnaast zijn de resultaten van collaborative filtering in de praktijk vaak beter dan die van content-based methoden.

Helaas introduceert collaborative filtering ook enkele nadelen. Een eerste probleem is '**cold start**'. Wanneer er niet voldoende gebruikers zijn die een item I beoordeeld hebben in een item-item based methode, is het moeilijk om dat item te matchen met andere reeds beoordeelde items en zal het minder snel aanbevolen worden door het recommendation algoritme. Hetzelfde geldt ook voor de user-user methode waar nieuwe gebruikers die nog geen beoordelingen gegeven hebben en gebruikers met een heel uniek profiel moeilijk of niet gematcht kunnen worden met andere gebruikersprofielen. Voor zulke gebruikers is het heel moeilijk om relevante items aan te bevelen. Hier zou men bijvoorbeeld kunnen terugvallen op de context van de gebruiker *(locatie, datum, tijd, besturingssysteem)* of het systeem gemiddelde. Dit probleem is vooral van toepassing op nieuwe gebruikers en nieuwe items ***(first rater probleem)***. Een zekere voorkennis van gebruikers en items is noodzakelijk. Hier zou mijn bijvoorbeeld kunnen terugvallen op een global baseline methode zoals besproken in Global baseline.

Daarnaast bemoeilijkt de **sparsity** in de utility matrix het om aanbevelingen te genereren. Zelfs wanneer er voldoende gebruikers zijn in het systeem, hebben de meeste gebruikers niet de meeste items beoordeeld. Het matchen van items en gebruikers blijft daardoor complex.

Tenslotte treedt een **popularity bias** op, het fenomeen dat vaak populaire items voorgesteld zullen worden. Dit vloeit voort uit het feit dat populaire items *(reclame uit de media, prijzen in de wacht gesleept)* vaak hogere ratings genieten *(het Harry Potter probleem bij Amazon[44])* en om deze reden sneller aanbevolen worden of een hogere geschatte rating genieten. Daar de meeste gebruikers een populair item beoordeeld hebben, zal dit item nog sneller aanbevolen worden. De reden hiervoor is dat deze extra ratings gebruikt kunnen worden om meer neighbours te kunnen vinden. Dit probleem vergroot zichzelf door enkel populaire items aan te bevelen en geen minder of niet-populaire items aan te bevelen. Bijgevolg verdwijnen niet-populaire items nog verder in de long tail en zullen ze steeds minder en minder vaak als suggestie naar boven borrelen. Hierover meer in de paragraaf over Problemen.

2.3 Knowledge-based systemen

Knowledge-based recommendation systemen zijn zeer specifieke aanbevelingssystemen[45]. aan de hand van een **knowledge base** en kennis over bepaalde items kan een recommendation systeem aanbevelingen geven die aansluiten op de noden van de gebruiker.

Een voordeel van deze methodiek is dat bepaalde problemen van traditionele recommendation systemen niet bestaan binnen knowledge-based manieren. Bijvoorbeeld de cold start en first rater problemen zijn hier niet van toepassing.

Knowledge-based recommender systemen worden vaak geïmplementeerd aan de hand van **AI** technieken zoals neurale netwerken, machine learning en anderen. De theoretische uiteenzetting over deze systemen wordt hier niet verder besproken.

2.4 Hybride systemen

Zoals blijkt uit vorige hoofdstukken hebben de diverse recommender methoden voor- en nadelen die vaak complementerend zijn. Een natuurlijk gevolg is om methodes te gaan combineren tot hybride methoden. Enkele voorbeelden van hybride recommender systemen zijn:

- Een content-based methode samen met een collaborative filtering. Zo kunnen item profielen gebruikt worden voor nieuwe items die nog geen ratings hebben *(oplossing first rater probleem)* en demografische gegevens *(gebruikers gemiddelde)* kunnen gebruikt worden voor nieuwe gebruikers. Overige beoordelingen gebeuren door middel van collaborative filtering.

- Recommenders en predictions kunnen gecombineerd worden in een lineair model. Zo kan een global baseline *(meest populaire items over het systeem)* gebruikt worden in samenwerking met een collaborative filtering algoritme.

---

[44] Vraag: Recommendation Systems: What exactly is "Harry Potter Problem"?
Amit Sharma, Postdoctoral researcher Microsof - 13/05/2013
quora.com/Recommendation-Systems-What-exactly-is-Harry-Potter-Problem - Geraadpleegd op: 11/04/2018

[45] Hoorcollege: Knowledge-Based Recommender Systems
Francesco Ricci - 07/12/2015
pdfs.semanticscholar.org/presentation/f0ca/45f0d1e7495af86fb9cfd6b6fc5693f35509.pdf
Geraadpleegd op: 11/04/2018

2.4.1 Global baseline

Wanneer een item I mogelijk aanbevolen kan worden aan gebruiker X, maar gebruiker X geen items beoordeeld heeft die verwant zijn met item I of er geen peer gebruikers zijn voor gebruiker X die item I beoordeeld hebben *(sparsity probleem),* kan er geen rating geschat worden voor item I van gebruiker X. Hier zou het algoritme kunnen terugvallen op een **global baseline** methode. De **gemiddelde score** van het item I in het hele systeem *(het gemiddelde van alle ratings voor item I, eventueel rekening houdend met de similarity van de gebruikers wie item I beoordeeld hebben)* samen met de gebruiker baseline en item baseline gebruikt worden om een beoordeling te genereren.

De gemiddelde ratings van gebruiker X kunnen vergeleken worden met de gemiddelde ratings van alle andere gebruikers om zo een **baseline rating** voor **gebruiker** X te bepalen. Een global baseline quantifier kan op diverse manieren berekend worden. Een voorbeeld werkwijze wordt hier gegeven:

- r_{XI} is de geschatte score die gebruiker X zou geven aan item I
- g is de gemiddelde score over heel de catalogus *(gemiddelde van alle ratings in matrix M)*
- d_I is het verschil tussen g en de gemiddelde score van item I
- d_X is het verschil tussen g en de gemiddelde rating van gebruiker X

$$r_{xi} = g + d_i + d_x$$

Intuïtief kan aan de hand van een **user baseline** rating bepaald worden wat voor een rater gebruiker X is. Indien gebruiker X gemiddeld gezien een lagere score geeft dan het systeem gemiddelde, is het duidelijk dat gebruiker X over het algemeen een lagere score zal toekennen aan de meeste items en omgekeerd. Daarnaast is het vanzelfsprekend dat wanneer een item I een hogere score heeft dan de gemiddelde score van andere items over heel de catalogus, item I door de meeste gebruikers hoger beoordeeld zal worden dan de andere items. Dit is de **item baseline**.

Wanneer men een global baseline estimate combineert met collaborative filtering kan het algoritme kijken naar de lokale neighborhood van items of gebruikers en de globale systeem informatie. Hier wordt een lineaire combinatie gemaakt van de global baseline classifier en de collaborative filtering classifier om zo tot een uiteindelijke schatting te komen.

Vaak wordt een collaborative filtering methode zo gedefinieerd voor item-item similarity:

- μ is de gemiddelde itemscore over heel de catalogus
- b_X is de rating afwijking van gebruiker X, de gemiddelde ratings van gebruiker X - μ
- b_I is de rating afwijking van item I, de gemiddelde ratings van item I - μ
- b_{XI} is de baseline estimate voor r_{xi} met volgende formule:

$$b_{xi} = \mu + b_i + b_x$$

Uitgewerkt op de voorbeeld dataset in Jaccard similarity geeft dat voor de aanbeveling van item I_1 voor gebruiker A:

$$\mu = \frac{1+4+5+1+4+2+42+5+2+1+5}{12} = 3$$

$$b_A = \frac{5+4+1}{3} - 3 = 0.33$$

$$b_{I1} = \frac{1+4}{2} - 3 = -0.5$$

$$b_{AI1} = 3 + 0.33 - 0.55 = 2.83$$

Het resultaat van de global baseline zou gecombineerd kunnen worden met een user-user of item-item methode. Stel dat een rating r_{xi} gegenereerd werd door een neighbourhood methode, dan kan deze gecombineerd worden met de global baseline b_{xi} in een lineaire combinatie. Een voorbeeld hiervan kan zijn:

$$M_{xi} = \frac{1}{2} \times (b_{xi} + r_{xi})$$

De lezer kan opmerken dat de beoordeling van gebruiker A voor item I_1 volgens de global baseline methode hoger ligt dan bij bijvoorbeeld de centered cosine methode zoals geïllustreerd in User-user filtering.

Global baseline kan ook op diverse manieren gecombineerd worden met een nearest neighbour methode zoals die in User-user filtering. Enkele use-cases zijn:

- Terugvallen op global baseline indien geen of weinig *(threshold)* peers te vinden zijn
- Gemiddelde van global baseline met cosinus similarity gebruiken

Merk op dat dit slechts één manier is om een global baseline te gebruiken. Uiteraard zijn er vele andere mogelijkheden. Een ander voorbeeld kan zijn om de global baseline procentueel te laten doorwegen afhankelijk van het aantal peers dat gebruikt wordt in de schatting.

3. Evaluatie van recommendation algoritmes

Wanneer een recommendation algoritme gecodeerd wordt, is het belangrijk om te weten of het systeem wel degelijk de juiste aanbevelingen aan de gebruikers geeft. Het is mogelijk om expliciet de gebruikers te vragen om hun suggesties te beoordelen. Met **expliciete feedback** kan het systeem rekening houden voor toekomstige suggesties *(machine learning)*. Zowel Facebook als YouTube gebruiken deze manier zoals te zien in de illustraties hieronder.

Figuur: YouTube expliciete feedback

Figuur: Facebook expliciete feedback

YouTube laat toe om aanbevolen items te verbergen zodat deze in te toekomst niet meer worden weergeven. Op basis hiervan verbetert YouTube haar recommendation algoritme in de 'black box'.[46]

Facebook laat toe om items te verbergen uit de nieuwsfeed zodat deze *(en vergelijkbare items)* in de toekomst minder of niet meer worden weergeven.[47] Dezelfde mogelijkheid bestaat ook om advertenties te verbergen.

Hoe sterk beide platformen rekening houden met deze expliciete feedback is voor discussie vatbaar.

In de praktijk gaat men echter vaak opteren om een **test dataset** te gebruiken waar de ratings al bekend zijn. Van deze dataset wordt een deel van vergeleken met de uitkomst van de berekende dataset. Er wordt gekeken of de ratings die het algoritme genereert overeenkomen met de effectieve ratings en de test verzameling.

3.1 RMSE

Om een recommender systeem te evalueren gebruikt men vaak een test dataset. Een test dataset is een *(subset van de effectieve)* rating matrix dat integraal beschouwd wordt als onbekende beoordelingen. Als tester is men op de hoogte van de effectieve ratings in de test dataset, het algoritme is dat niet. Om het algoritme te evalueren zal dezelfde subset van berekende ratings dat het algoritme output, vergeleken worden met de effectieve subset, de test dataset. Het doel is om het **verschil** tussen de **effectieve ratings** *(de gekende maar weerhouden ratings in de rating matrix)* en de **berekende ratings** te **minimaliseren**. Formeel wil men de standaardafwijking zo dicht mogelijk 0 laten benaderen.

[46] Paper: The YouTube Video Recommendation System
J. Davidson, B. Liebald, J. Liu, P. Nandy & T. Van Vleet ontwikkelaars Google - 11/08/2014
researchgate.net/publication/221140967_The_YouTube_video_recommendation_system

[47] Artikel: Recommending items to more than a billion people
Maja Kabiljo & Aleksandar Ilic ontwikkelaars Facebook - 02/05/2015
code.facebook.com/posts/861999383875667/recommending-items-to-more-than-a-billion-people
Geraadpleegd op: 09/05/2018

Stel de verzameling ratings voor als T met |T| het aantal beoordelingen *(aantal gebruikers maal aantal items)*. Vaak wordt de root-mean-square error ***(RMSE)***[48] berekend. RMSE wordt berekend aan de hand van r_{xi}, de berekende rating en r'_{xi}, de effectieve rating. De formule voor RMSE luidt als volgt:

$$RMSE(T) = \sqrt{(\frac{\sum_i^{|T|}(r'_{xi} - r_{xi})^2}{|T|})}$$

De bedoeling is nu om magnitude van de RMSE waarde te minimaliseren, wat men 'Least mean square prediction error' noemt. M.a.w. moet de limiet van de RMSE waarde naar 0 toegaan. Deze methode is heel eenvoudig, maar zal niet altijd een correct resultaat opleveren. Dit komt doordat alle waarden in de testset vergeleken worden en niet enkel de aanbevolen items. In paragraaf Problemen meer informatie over de moeilijkheden met het werken met een RMSE qualifier.

Merk op dat RMSE gebruik maakt van een kwadratisch verschil. De reden hiervoor is zodat negatieve fouten en positieve fouten elkaar niet zouden opheffen. *(Stel dat ergens in de dataset er een fout is van -1 en voor een andere waarder er een afwijking is van +1, door deze twee op te tellen krijgt men 0 wat impliceert dat er geen fout is.)* Door positieve kwadraten op te tellen kunnen ze elkaar nooit opheffen. De vierkantswortel maakt deze kwadratering na de optelling en deling weer ongedaan.

De lezer kan observeren dat het bereik van de RMSE functie strikt tussen de randwaarden van het rating interval ligt. In het voorbeeld dat doorheen dit document gebruikt wordt is de rating schaal 1-5 sterren of punten. Bijgevolg is de maximale RMSE waarde 5 - 1 = 4. De lezer kan zich voorstellen dat de RMSE waarde niet buiten het interval [0, 4] kan liggen en dat een optimale RMSE waarde zich dicht bij 0 situeert.

Een andere manier om een recommendation algoritme te evalueren is door enkel te kijken naar het feit of een aanbeveling al dan niet gemaakt werd (bv. dat een rating hoger is dan 2.5). Via deze weg wordt de effectieve waarde van de rating genegeerd. Logischerwijs is deze meting vergeeflijker dan RMSE. Het percentage concrete aanbevelingen geeft in deze werkwijze de indicatie van de accuraatheid van het recommendation algoritme.

Om goede aanbevelingen te kunnen genereren moeten een aantal obstakels overwonnen worden. Enerzijds de diversiteit en anderzijds een aantal problemen met de verifiëring van de aanbevelingen.

3.2 Leave one out

Een andere evaluatie metriek is **leave one out**. Het idee is dat er voor gebruikers telkens **één** reeds **gekende beoordeling** gekozen wordt en als **withheld** wordt gemarkeerd. Deze waarde is dan niet langer zichtbaar voor het recommendation algoritme en het recommendation algoritme zal deze beoordeling gaan schatten. Deze gegenereerde beoordeling kan dan vergeleken worden met de effectieve beoordeling. Deze afwijking kan bijvoorbeeld gebruik maken van RMSE. De test kan herhaald worden om telkens andere targets te kiezen die left out zijn om zo toch een gemiddelde op te kunnen stellen.

Via deze methode kan enkel de accuraatheid van deze **left outs** bepaald worden. Men gaat er van uit dat deze representatief zijn voor de gehele dataset. Deze methode wordt in de statistiek ook wel **cross-validation** genoemd.

[48] Artikel: RMS Error
Susan Holmes Stanford University - 28/11/2000
statweb.stanford.edu/~susan/courses/s60/split/node60.html - Geraadpleegd op: 09/05/2018

3.3 Problemen

3.3.1 Diversiteit aan aanbevelingen

RMSE kijkt alleen naar de numerieke correctheid van beoordelingen maar niet naar de variatie van suggesties. Een gebruiker die de film Die Hard heeft gezien zou best niet enkel andere Die Hard films gesuggereerd krijgen. De kans is namelijk groot dat deze kijker de anderen ook al gezien heeft *(maar nog geen beoordeling gegeven heeft)*. Door telkens dezelfde aanbevelingen te laten zien, zullen gebruikers het mogelijk beu raken of denken dat het recommendation algoritme niet goed werkt. Deze negatieve gebruikers perceptie wil men uiteraard vermijden. Diversiteit maakt het voor de gebruiker interessanter. Naast Die Hard films kunnen de aanbevelingen gemixed worden met Rambo en Terminator films. Dit kan echter wel niet gerealiseerd worden puur op basis van gebruikers-beoordelingen, meer data is hiervoor noodzakelijk. Data zoals hoe sterk films overeenkomen met andere films *(item-item similarity of content-based similarity)* die niet op basis van sociale interactie van gebruikers berekend werd, is hier nodig.

Niet alleen is het voor de gebruiker interessanter om nieuwe aanbevelingen te krijgen, het zorgt er ook voor dat de gebruiker mogelijks geïnteresseerd raakt in nieuwe categorieën om zo een heel ander deel van de catalogus te gaan exploreren. Door de gebruiker meer te laten interageren verbetert mogelijk de gebruikerservaring en kunnen de inkomsten van het platform mogelijk stijgen.

Prediction diversity is een mogelijk gevolg wanneer alle voorgestelde items te gelijkaardig zijn. Zo zou een gebruiker die de eerste film van Harry Potter een hoge rating gaf, allemaal Harry Potter films als aanbevelingen ontvangen *(daar Harry Potter films een hoge onderlinge similarity hebben)*. Zo zou de gebruiker enkel films van eenzelfde genre gaan beoordelen en nog meer items van hetzelfde genre aanbevolen krijgen. Het is niet omdat en feestganger een TD als leuk markeert dat hij interesse heeft in alle TD's in het land.

3.3.2 Context van aanbevelingen

Prediction context is een probleem waar items wel interessant zijn voor de gebruiker, maar slechts in een bepaalde context. Het is niet interessant om een kerstfilm aan te bevelen in de zomer of een actiefilm in de ochtend. In een online boekenwinkel zou het niet interessant zijn reisgidsen aan te bevelen wanneer de gebruiker reeds terug is van vakantie. Voor evenementen is het niet interessant om een klein feest aan de andere kant van het land te organiseren. Het zou voor de gebruiker immers niet interessant zijn om helemaal naar daar te gaan. Een ander voorbeeld is om items te blijven aan-bevelen na aankoop, een fenomeen dat vaak voorkomt bij webshops zoals in de illustratie hier-langs.

Figuur: foute context van aanbevelingen

3.3.3 Biased rating gedrag

Biased rating gedrag ontstaat wanneer gebruikers een verschillende perceptie van de ratingschaal hebben. Films die een vervolg zijn van een eerder succes worden vaak minder goed gewaardeerd; is dat omdat de vervolgen slechter zijn? Dat is voor discussie vatbaar. De lezer kan zich voorstellen dat dit niet altijd het geval is. Een verklaring hiervoor is dat wanneer een film goed is, de kijker er van uit gaat dat het vervolg ervan ook goed moet zijn. Zijn verwachtingen liggen dus hoger en zijn beoordeling zal dan ook strenger zijn voor het vervolg *(wat kan resulteren in een lagere beoordeling*).

Wanneer de publieke opinie een film als fantastisch omschreven heeft, kan ook dit het rating gedrag van kijkers gaan beïnvloeden. Zij kunnen de publieke trend volgen en hoger gaan beoordelen omdat iedereen de film goed vindt. Anderzijds kan men juist kritischer gaan kijken en lager gaan beoordelen net omdat de verwachtingen zo hoog liggen door al de heisa.

Daarnaast kan een film die een kijker al eerder gezien heeft *(bijvoorbeeld in zijn jeugdjaren)* beter gaan beoordelen dan de film eigenlijk is. Dit fenomeen vloeit voort uit het nostalgisch gevoel of omdat de verhaallijn van de film nog in het geheugen van de kijker zit[49]. Zo kan hij of zij al dan niet bekend zijn met de verhaallijn en kan het slot van de film een invloed hebben op de kijkervaring.

Stelt u zich voor dat u als kind een horror film heeft gezien en u zich de film bent blijven herinneren, omdat u de film in uw jeugdjaren als eng ervaren heeft. De kans is groot dat wanneer u de film nu, jaren later, opnieuw zou bekijken, u de film als minder eng zal ervaren. Redenen hiervoor kunnen zijn: eigen evolutie, verouderde special effects of omdat u al weet hoe het verhaal zal aflopen. Is de film daarom minder goed dan toen?

Beoordelingen vormen een moeilijke data om mee te werken. Zelfs wanneer men rekening kan houden met alle factoren kan het nog zo zijn dat de gebruiker **niet consistent** is in zijn of haar beoordelingen. Vaak zal de beoordeling van de gebruiker afhankelijk zijn van zijn huidige gemoedstoestand, wat ervoor zorgt dat een beoordeling die de gebruiker nu maakt wellicht niet identiek zal zijn aan een beoordeling die hij volgende week maakt.

3.3.4 Niet-gebalanceerde dataset

Een **niet-gebalanceerde dataset** bemoeilijkt het schatten van ratings. Niet alle gebruikers geven evenveel ratings en andere gebruikers geven enkel een rating indien ze ontevreden zijn en niet wanneer ze wel tevreden zijn. Voor zulke gebruikers is het complexer om een correct model te genereren. Ook de verdeling van de dataset is niet-uniform. Dit ontstaat door een klein aantal gebruikers en items waar veel mee geïnterageerd wordt, die heel populair zijn. Een groot stuk van de catalogus heeft weinig tot geen interactie *(long tail)*. Een concreet voorbeeld hiervan zijn internationale sterren op social media.

[49] Persartikel: Spoilers actually enhance your enjoyment
Alison Flood journalist The Guardian - 17/08/2011
theguardian.com/books/booksblog/2011/aug/17/spoilers-enhance-enjoyment-psychologists
Geraadpleegd op: 28/03/2018

Katy Perry heeft op Twitter de meestal volgers[50] en haar Twitterprofiel is daarmee een heel belangrijk item binnen het volgers netwerk op Twitter. Zij bereikt met haar account het grootste deel van de Twitter gebruikers. Dit fenomeen wordt nog meer in de hand gewerkt door optimalisaties. Deze meest populaire items zijn dan ook vaak de items die gecached worden *(films op Netflix)* of items die in een fysiek filiaal beschikbaar zijn *(producten bij Amazon)* en sneller aan de mensen getoond kunnen worden. Hieruit ontstaat een nieuw probleem: hoe worden niet-populaire items wel populair? Om op deze laatste vraag een oplossing te bieden zou men kunnen terugvallen op simpele aggregaten. Zo zouden hand picked items, trending items *(globale interactie)* of recent toegevoegde items aanbevolen kunnen worden. Dit sluit tevens aan met het probleem van diversiteit in aanbevelingen.

3.3.5 User-interface

De **volgorde** van suggesties is ook van belang. Dit hangt sterk samen met de indeling van de gebruikersinterface. De gebruiker kan suggesties die naast elkaar staan opgelijst *(evenwaardig)* als een andere metafoor beschouwen dan items die onder elkaar staan weergeven *(hiërarchie)*. Daarnaast kan ook een grootteverschil tussen aanbevelingen een andere waarde impliceren. Zoals eerder vermeld kan het voor de gebruiker ook een verschil zijn indien een item een geschatte beoordeling van 4 of 5 sterren geeft. Hij of zij gaat er mogelijk van uit dat een item met slechts 4 van de 5 sterren helemaal niet zo goed is. Om deze problemen te vermijden wordt vaak de berekende score van een suggestie niet weergeven.

3.3.6 Popularity bias

Daarbij treedt een **popularity bias** op, het fenomeen dat vaak populaire items voorgesteld worden. Dit vloeit voort uit het feit dat populaire items *(reclame uit de media, prijzen in de wacht gesleept)* vaak hogere ratings genieten *(het Harry Potter probleem bij Amazon)* en sneller door een recommendation algoritme aanbevolen zullen worden.

3.3.7 Accuraatheid vs. doel

Door de sterke focus op **accuraatheid** *(minimaal verschil tussen berekende ratings en effectieve ratings)* wordt het punt van aanbevelingen eigenlijk gemist. Het doel van een recommendation algoritme is niet om exact de scores van items te schatten die gebruikers aan items zouden toekennen *(dit is bekend als matrix completion)*[51], maar om items aan te bevelen die de gebruiker interessant zal vinden *(dit zijn niet perse de items met de hoogste scores)*.

RMSE kan bepaalde recommender systemen afstraffen die eigenlijk goede resultaten boeken. Wanneer een algoritme goed scoort *(ratings genereert kort bij de effectieve ratings)* voor items met een hoge score maar slecht scoort voor items met een lage score zal RMSE het algoritme als middelmatig tot slecht beschouwen. Echter zijn enkel items met een hoge score interessant om aan te bevelen aan de gebruiker. In praktijk heeft het algoritme wel goed gewerkt, maar enkel voor de **hoge scores**, de items die ook uiteindelijk aan de gebruiker getoond zullen worden. De effectieve ratings voor items die niet aanbevolen worden zijn niet interessant voor de gebruiker. *(Het doet er dan ook niet toe dat de exacte score voor deze items fout is.)* Een alternatief is noodzakelijk.

---

[50] Twitter: Most followers
friendorfollow.com/twitter/most-followers - Geraadpleegd op: 15/03/2018

[51] Paper: Top-N Recommender System via Matrix Completion
Zhao Kang, Chong Peng & Qiang Cheng Southern Illinois University - 19/01/2016
aaai.org/ocs/index.php/AAAI/AAAI16/paper/download/11824/11581

Mogelijk kan enkel de precisie van het algoritme getest worden in de top K aanbevolen items, de items met een hoge score die effectief aan de gebruiker getoond worden. Dit probleem wordt geïllustreerd aan de hand van volgend voorbeeld, dezelfde dataset als in paragraaf Jaccard similarity, maar dan volledig ingevuld en met enkele weerhouden velden.

	gebruiker A	gebruiker B	gebruiker C	gebruiker D
item I_1	**2**	**4**	**1**	**4**
item I_2	**5**	**1**	**4**	**2**
item I_3	**4**	**2**	**5**	**2**
item I_4	**1**	**5**	**1**	**4**

Tabel: voorbeeld volledig ingevulde rating matrix

De gele cellen duiden de beoordelingen aan die voor dit voorbeeld werden toegevoegd zodat de rating matrix volledig is. Stel dat een recommendation algoritme de gele cellen met volgende waardes benaderd heeft:

	gebruiker A	gebruiker B	gebruiker C	gebruiker D
item I_1	**1**	**4**	**1**	**4**
item I_2	**5**	**1**	**4**	**2**
item I_3	**4**	**2**	**5**	**2**
item I_4	**1**	**5**	**3**	**5**

Tabel: voorbeeld volledig ingevulde rating matrix geschat door recommendation algoritme

Laten we eerst kijken naar een waarde van een hoge rating welke het systeem goed geschat heeft. In dit voorbeeld zullen we kijken naar de beoordeling die het systeem schat dat gebruiker B aan item I_1 zou toekennen. Uit de eerste matrix kan men afleiden dat de effectieve waarde 4 is en het aanbeveling algoritme eveneens een 4 gokt als mogelijke beoordeling. Ingevuld in de RMSE formule geeft dat:

$$RMSE = \sqrt{(\frac{(4-4)^2}{1})} = 0$$

Het resultaat is hier 0 wat wil zeggen dat de fout 0 is. Een perfecte score dus *(een goede score is wanneer de RMSE waarde dicht bij 0 ligt)*. Het systeem blijkt voor alle hoge ratings goed te werken. Wanneer RMSE test op de ratings die het systeem laag geschat heeft, namelijk de items die de gebruiker volgens het systeem niet zal leuk vinden, wordt het resultaat afgestraft. Om dit te illustreren kunnen we kijken naar de beoordeling van gebruiker C voor item I_4. Uit de eerste matrix kan men afleiden dat de effectieve waarde 1 is en dat het aanbeveling algoritme 3 aangeeft als geschatte beoordeling. Deze beoordeling is veel hoger dan de effectieve beoordeling. De RMSE waarde zal dan ook ver van 0 liggen.

$$RMSE = \sqrt{(\frac{(3-1)^2}{1})} = 2$$

Het resultaat is hier 2, wat ver van 0 ligt *(relatief ten opzichte van de ratingschaal [1, 5] en de maximale fout die tussen 0 en 4 ligt)*. Het aanbevelingssysteem scoort voor deze beoordeling slecht volgens de RMSE formule. Men kan echter beargumenteren dat een item dat 3 punten krijgt in een schaal van [1, 5] niet zo fantastisch hoog gescoord heeft en daarom beter niet aanbevolen zou moeten worden. 3 is dan ook slechts nipt hoger dan een neutrale 2.5 waarde.

Als het systeem een threshold zou hebben van minstens 3.5 of 4 punten zou het item I_4 zelfs niet aanbevolen worden aan gebruiker C en is het irrelevant wat de beoordeling voor item I_4 is, zolang het maar niet wordt voorgeschoteld aan gebruiker C.

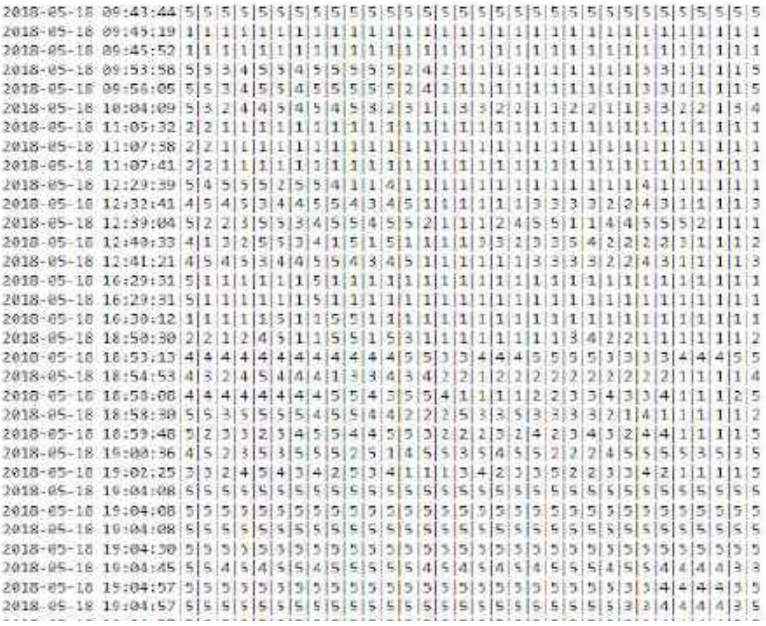
In dit voorbeeld zou men dus beter testen op de top K aanbevolen items en niet op de gehele test dataset. Merk op dat het testen van reeds gekende beoordelingen zinloos is doordat de RMSE waarde altijd 0 zal bedragen. Reeds bekende waardes meenemen in het resultaat zou de accuraatheid van het recommendation algoritme op papier ook alleen maar verbeteren.

4. Implementatie

Initieel wilde de schrijver van deze bachelorproef gebruik maken van een **evenementen dataset** afkomstig van zijn eigen privé project 'Party Locator'. Echter na het privacy incident met Facebook in April 2018 werden enkele Facebook API endpoints depecrated met het gevolg dat Party Locator niet langer toegang heeft tot evenement data. Er is uiteindelijk geopteerd om dan een andere dataset te gaan gebruiken. Wel is er nog steeds gekozen voor een evenementen dataset maar in de vorm van een enquête. Deelnemers werd gevraagd om in een lijst van 33 evenementen telkens hun beoordeling te geven op de evenementen zoals te zien op de illustratie hieronder.

Figuur: online enquête om impliciet data te verzamelen

Deze beoordelingsmatrix, bestaande uit al de reacties op deze vragenlijst, vormt de test dataset voor de implementatie.Een snapshot van deze beoordelingen kan hieronder bekeken worden:



Figuur: rating matrix gegenereerd uit expliciete dataverzameling

De implementatie is gebeurd in **Python 3** zonder gebruik te maken van libraries. Immers draait deze bachelorproef rond de **fundamentele werking** van deze algoritmes en niet de fancy optimalisatie-technieken die bekende bibliotheken gebruiken. De algoritmes zijn dan ook exacte uitwerkingen van de formules in dit document.

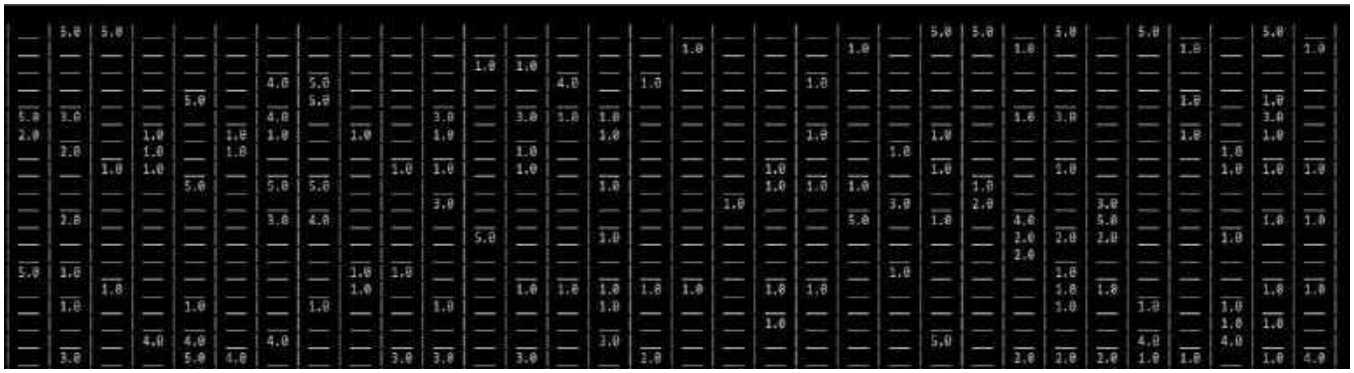
Opmerkelijk tijdens het schrijven van de code is dat veel **keuzes** binnen het algoritme de resultaten van de **uitvoer** sterk kunnen **veranderen**. Belangrijke factoren zijn: het al dan niet gebruiken van peer gebruikers met een negatieve similarity en het al dan niet gebruiken van een top K peers om het resultaat te bepalen. In zeer kleine datasets ontstond ook het probleem dat door enkel aanbevelingen te genereren aan de hand van positieve similarity gebruikers, er soms te weinig of zelfs geen matches waren om een schatting te maken. *(Dit probleem vergroot zichzelf wanneer de sparsity van de matrix toeneemt.)* Daarnaast was het resultaat van de berekende rating ook afhankelijk van de gebruikte ratings; de initiële rating matrix of de genormaliseerde rating matrix.

In onderstaande grafieken geeft de blauwe lijn de RMSE waarde aan voor de gehele test dataset. De rode lijn weergeeft een andere formule, namelijk het percentage juiste aanbevelingen *(enkel beoordelingen met een waarde hoger dan 3 min het aantal foute aanbevelingen)*. De oranje en groene lijn weergeven respectievelijk de RMSE formule enkel toegepast op de berekende waarden en de top K berekende waardes. De threshold om als top gecategoriseerd te worden is in dit voorbeeld 3.

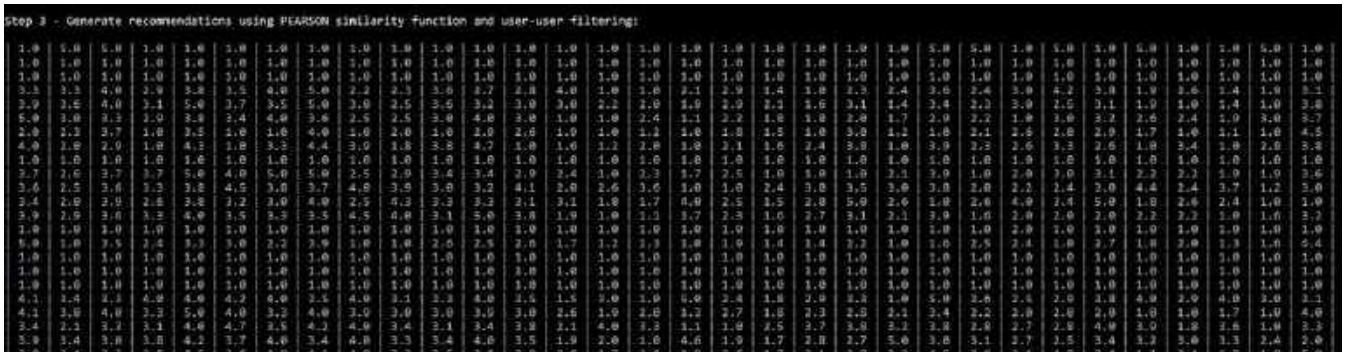
De horizontale X-as stelt de sparsity van de matrix voor waar 0% een volledig lege matrix is en 100% een volledige gekende rating matrix. De verticale Y-as stelt enerzijds de absolute foutmarge voor bij de RMSE metrieken. Anderzijds stelt het interval [0, 1] op de Y-as het percentage correcte aanbevelingen voor. Het rating interval is hier [1, 5] *(inclusief)*.

Een enkele test bestaat uit een loop die van 1% sparsity tot 100%, in stappen van 1%, de accuraatheid van het recommendation algoritme evalueert. De test werd uitgevoerd voor de drie geziene methoden in deze paper: content-based, collaborative user-user en collaborative item-item methode. Voor elke test werd de invloed van het al dan niet gebruiken van de global baseline getoetst evenals de impact van het incorporeren van alle peer gebruikers/items of enkel de top K *(positieve)* matches. Deze methodes werden voor elke beschreven similarity functie gedaan: Jaccard, cosine en Pearson. Om een test te doen werd een deel van de volledige rating matrix *(at random)* onbekend gemaakt. Deze waardes worden vervolgens geschat door het recommendation algoritme waarna ze vergeleken worden met de effectieve waardes. Om fluctuaties tussen verschillende testen te minimaliseren werd de test meerdere keren uitgevoerd door telkens andere data uit de matrix te halen en vormen de grafieken een illustratie van het gemiddelde van al deze testen.

In de figuur hieronder wordt een deel van de rating matrix getoond waar 20% van de ratings behouden werd *(de matrix is 80% sparse)*:



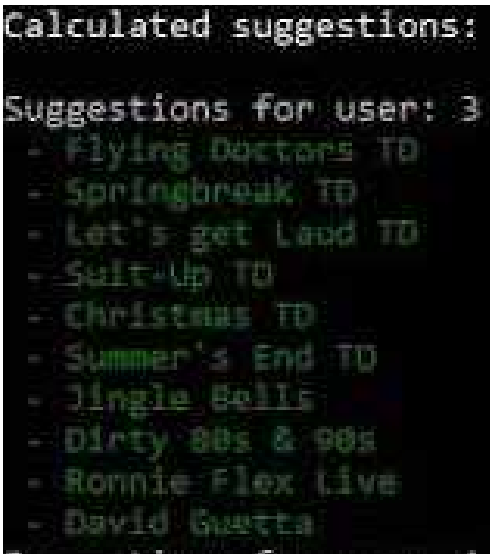
Onderstaande figuur illustreert een deel van de rating matrix waarin onbekende ratings ingevuld werden door een recommendation algoritme. Dit voorbeeld maakt gebruik van user-user filtering en de centered cosine *(Pearson)* similarity functie.



Een overzicht van de effectieve beoordelingen en de gegenereerde beoordelingen voor gebruiker 3 *(4de record in deze dataset)*. Rij 1 is de effectieve data, rij 2 de weerhouden data en rij 3 is tenslotte de berekende data.

5	5	3	4	5	5	4	5	5	5	5	5	2	4	2	1	1	1	1	1	1	1	1	1
						4	5						4		1				1				
3.3	3.3	4.0	2.9	3.8	3.5	4	5	2.2	2.3	3.6	2.7	2.8	4	1.0	1	2.1	2.9	1.4	1	2.3	2.4	3.6	2.4	3.0

De afbeelding hieronder illustreert welke aanbevelingen naar gebruiker 3 gestuurd zouden worden na afloop van het recommendation algoritme:

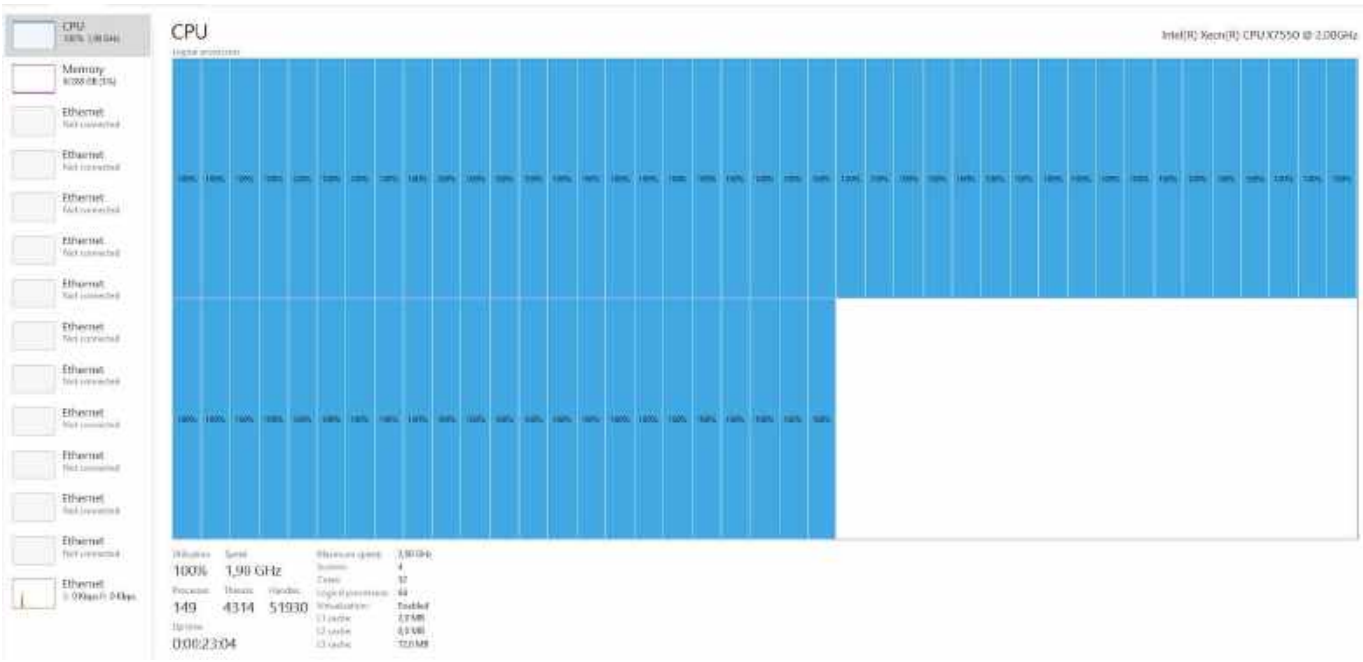


De laatste illustratie toont het gegenereerde gebruikersprofiel voor diezelfde gebruiker 3 *(4 de rij in de dataset)*:

De lezer kan zich inbeelden dat dit een lange berekening is. Voor alle gebruikers werden alle items overlopen en de volledige rating matrix werd zo ingevuld. Deze test werd voor elk percentage sparsity *(100 testen)* uitgevoerd. Ook moest de matrix volledig overlopen worden om de accuraatheid te kunnen toetsen.

Een opmerking die de aandachtige lezer kan maken is dat een volledig **lege matrix** *(sparsity 100%)* niet mee opgenomen is in het resultaat. De reden hiervoor is dat in een volledig lege matrix een recommendation algoritme geen data heeft om berekeningen te kunnen doen. Het genereren van beoordelingen is dan puur at **random** *(of een constante).* Het is vrijwel onmogelijk om de waardes van een matrix op magische wijze te schatten.

Om het evalueren te **versnellen**, wordt de applicatie **multithreaded** geschreven. Elke thread voert één test uit *(elk sparsity niveau evalueren voor een gegeven methodiek).* De applicatie moet wel voor elke combinatie van recommendation algoritme met global baseline en positieve matching een nieuwe test uitvoeren. Het gebruik van een server *(HP ProLiant DL580 G7)* met 64 threads versnelt de evaluatie In de figuur hieronder wordt het venster taakbeheer getoond tijdens het genereren van één grafiek. Zo duurde het maken van één grafiek ongeveer een kwartier. Een verdere optimalisatie is om de rating matrix te kopiëren naar elke thread. Via deze weg is er minder gedeeld geheugen en kan elk proces sneller de ratings raadplegen zonder op anderen te moeten wachten. Alleen een dictionary wordt gedeeld tussen de processen waarin de tussenresultaten opgeslagen worden. Van al deze tussenresultaten wordt uiteindelijk de grafiek geplot op basis van deze gemiddelden.



Figuur: 64 threads om recommendation algoritmes te evalueren

4.1 Algemene observaties

Als men naar de grafieken kijkt in paragraaf Accuraatheid recommendation algoritme, kan geobserveerd worden dat de correctheid van een aanbevelingsalgoritme globaal gezien toeneemt *(de standaardfout neemt af)* naarmate er meer waardes bekend zijn in de rating matrix. Dit is een logisch gevolg van het feit dat er meer data is om neighbours te vinden wanneer de rating matrix meer gevuld *(minder sparse)* is.

Daarnaast neemt het aantal correcte suggesties *(de geschatte waarde van de beoordeling heeft geen belang, enkel het al dan niet gesuggereerd worden van een item)* toe naarmate de rating matrix minder en minder sparse is.

Een algemene trend is dat de blauwe lijn die de RMSE waarde illustreert, sneller daalt en dus een betere score impliceert dan de oranje en groene lijnen. De reden hiervoor is dat de blauwe lijn de gemiddelde afwijking over alle waardes in de rating matrix meerekent, ook de reeds bekende waardes die niet berekend werden. De afwijking van deze waardes met de effectieve waardes is uiteraard 0. Deze waardes incorporeren heeft dan ook een positieve invloed op de globale afwijking, vandaar de snellere daling dan de andere metrieken.

Een andere observatie is dat alle RMSE waardes op 100% *(daar waar de sparsity 0% is en alle waardes reeds gekend zijn)* naar 0 vallen en dat de rode lijn die het percentage correcte aanbevelingen modelleert naar 100% stijgt. De reden hiervoor is triviaal. Wanneer alle waardes gekend zijn moeten er geen aanbevelingen gegenereerd worden. Elke metriek gaat dan over een matrix waar alle waardes identiek zijn aan de gegeven rating matrix. Bijgevolg is de afwijking 0 en het percentage correcte aanbevelingen 100% of 1.

Wat ook opmerkelijk is, is dat de rode lijn vrijwel niet onder 50% valt. Een mogelijke verklaring hiervoor is dat het al dan niet aanbevolen worden in deze test vrijwel binair is. De kans op een binaire rating is dan ook 50%.

4.2 Methodes met elkaar vergeleken

In de collaboratieve aanpak die gebruik maakt van user-user relaties valt het op dat alle drie de RMSE lijnen sneller dalen in het interval 1-15%, hierna is de daling zachter. Een mogelijke verklaring is dat er **weinig correlatie** is in de erg kleine dataset. Het percentage correcte aanbevelingen neemt in dit interval *(tussen de 1% en 15%)* ook het sterkste toe.

Opmerkelijk is dat bij de content-based methode de cosinus similarity als enige relatief goed scoort. Jaccard similarity en centered-cosine similarity blijven constant slecht en geven zelfs minder goede resultaten met meer data.

In de user-user methode scoort Jaccard similarity beduidend slechter dan cosine of centered cosine. Vanaf er meer dan 60% data beschikbaar is in de rating matrix geeft de Jaccard similarity wel een beter resultaat, zelfs even goed als de cosine en centered cosine similarities.

Daarnaast scoren zowel **content-based** als collaboratieve met de **item-item** relaties beduidend **minder goed** dan eender welke collaboratieve aanpak met **user-user** filtering. Dit is zoals verwacht zoals aangegeven in Soorten recommendation algoritmes. Bij deze methodieken komt het ook enkel voor dat de oranje lijn beter scoort dan de groene lijn. Dit wil zeggen dat bij de user-user aanpak het algoritme beter scoort voor positieve beoordelingen dan voor negatieve beoordelingen. Bij de overige methoden is de accuraatheid van positieve aanbevelingen beduidend slechter dan de accuraatheid van alle aanbevelingen.

Verder is het opmerkelijk dat bij de collaboratieve item-item filtering er vrijwel **geen verschil** is tussen de **similarity functies**. De reden hiervoor is mogelijk gebruikers niet perse van een bepaald type evenement houden maar eerder kijken naar de naam, line-up en andere factoren.

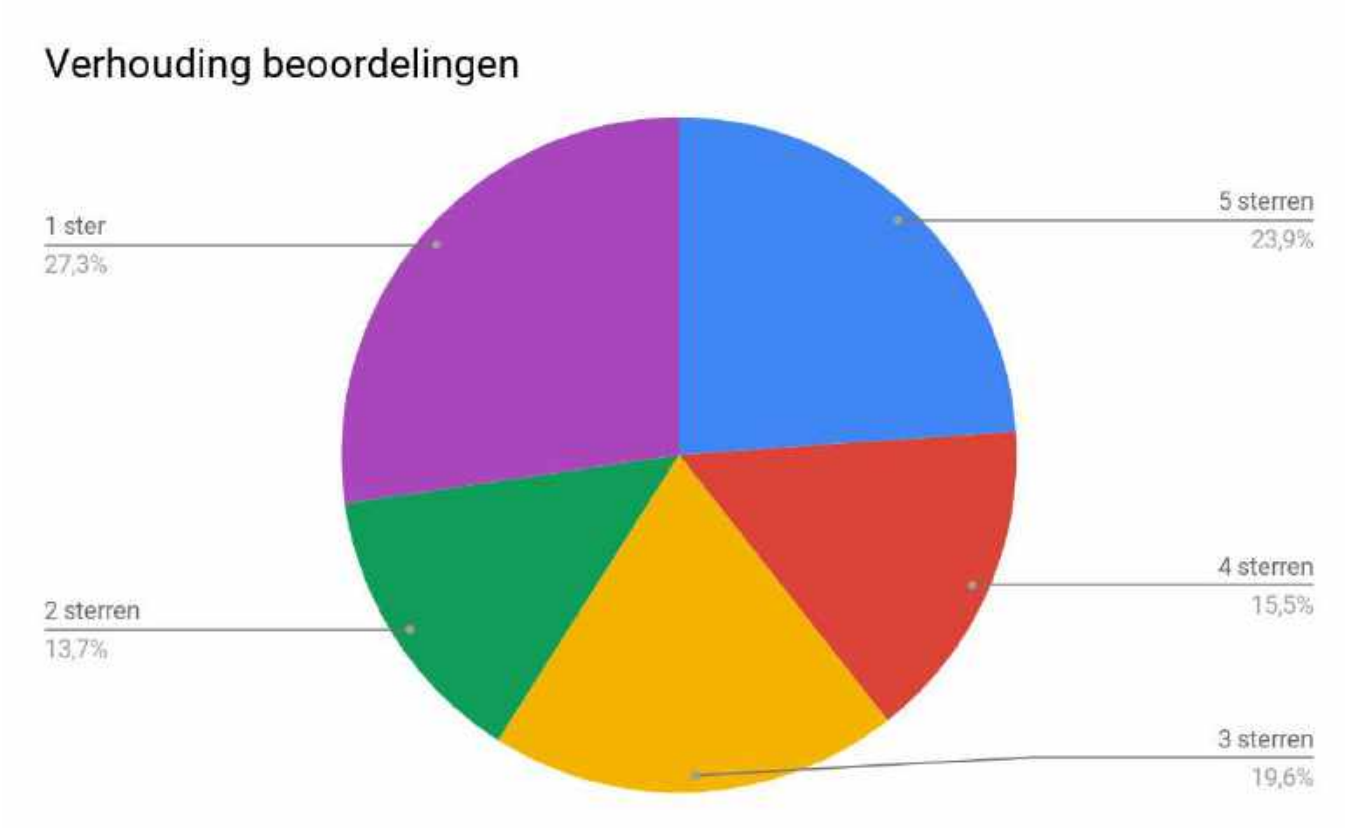
Het gebruik van de **global baseline** heeft een negatieve invloed op de accuraatheid van de recommendation algoritmes in dit voorbeeld. De reden hiervoor is mogelijk te wijten aan het feit dat het globale gemiddelde te sterk doorweegt op gewogen gemiddelde van de nearest neighbours. Daar deze dataset sterk uiteenlopende gebruikers omvat is het mogelijk dat global baseline beter niet gebruikt wordt in dit voorbeeld.

Bij evenementen is het vaak zo dat bepaalde vriendengroepen naar eenzelfde soort evenementen gaan en voor dit evenement het gemiddelde omhoog kunnen trekken. Daarnaast speelt de invloed van een studentenvereniging ook een rol. Zo zullen mogelijks studenten van de ene vereniging de evenementen van een andere vereniging mindere scores gegeven hebben. Ook dit kan een invloed gehad hebben op de item-item methode.

Voor dit voorbeeld kan geconcludeerd worden dat **user-user** similarity zonder gebruik te maken van een global baseline de beste resultaten geeft. Het slecht scoren van de content based en methode kan te wijten zijn aan te weinig features of juist de verkeerde attributen die werden toegekend aan items. Dit resultaat wijst nogmaals op het feit dat het moeilijk is om correcte features toe te kennen en dat features mogelijk niet sluitend zijn voor een bepaald type item.

Zoals verwacht geeft de user-user methode samen met de cosinus similarity of centered cosinus similarity de beste resultaten. Via deze werkwijze kan al vanaf 20% gekende data een standaardafwijking van 1.5 ster bereikt worden.

De verdeling van de ratings is:



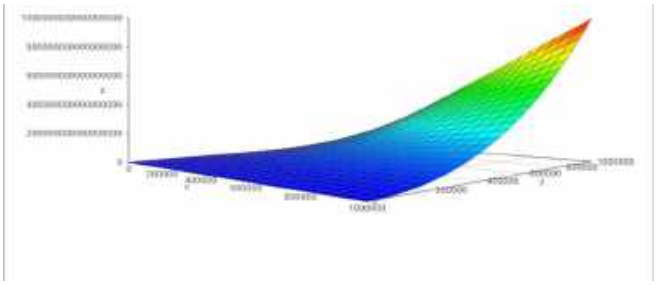
Dit impliceert dat de ratings goed verdeeld zijn en dat de deelnemers van de enquête niet binair antwoorden gegeven hebben. Ook zijn er een aantal **outliners** die enkel 1 ster of enkel 5 sterren gegeven hebben. Drie personen gaven op elk evenement 1 ster en zes personen gaven op elk evenement 5 sterren. Er was ook één persoon die enkel 3 sterren gaf, een heel neutrale mening. Voor de user-user methode met positieve matching en zonder gebruik te maken van een global baseline quantifier werd ook een test gedaan zonder deze outliners. Het resultaat hiervan is terug te vinden onder Accuraatheid recommendation algoritme. Het resultaat zonder incorporatie van outliners maakt het algemene resultaat een beetje beter maar niet opvallend.

4.3 Complexiteit implementatie

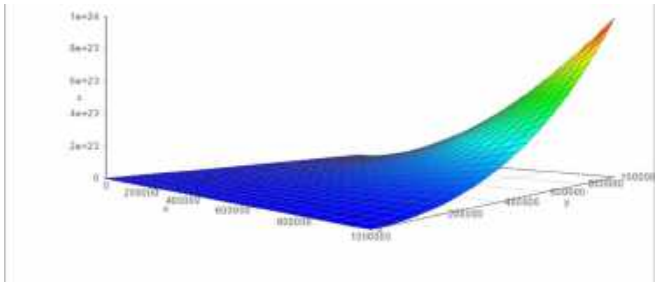
De **complexiteit** van de naïeve implementatie kan als volgt berekend worden. Stel dat U het aantal gebruikers is, I het aantal items en A het aantal features in een item profiel:

- Content based
  - Jaccard/cosinus similarity: (U x I x A) x I x A = $O(UI^2A^2)$
  - Centered cosinus similarity: (U x I x A) x I x A x A = $O(UI^2A^3)$
- Item-item
  - Jaccard/cosinus similarity: I x (I - 1) x U x 2U = $O(I^2U^2)$
  - Centered cosinus similarity: I x (I - 1) x U x U x 2U = $O(I^2U^3)$
- User-user
  - Jaccard/cosinus similarity: U x (U - 1) x I x 2I = $O(U^2I^2)$
  - Centered cosinus similarity: U x (U - 1) x 1 x 1 x 2I = $O(U^2I^3)$

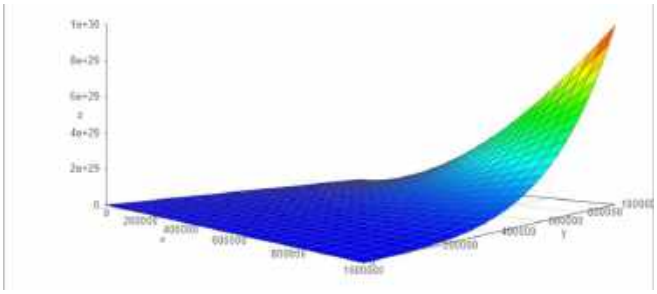
Het aantal item features is meestal een constante. Hieronder wordt de complexiteit van het aantal items ten opzichte van het aantal gebruikers afgewogen. Lagere complexiteit is natuurlijk beter.



Figuur: complexiteit gebruikers vs. items content based
academo.org/demos/3d-surface-plotter/?expression=x*y&xRange=0,1000000&yRange=0,1000000&resolution=25



Figuur: complexiteit gebruikers vs. collaborative Jaccard/cosinus similarity
academo.org/demos/3d-surface-plotter/?expression=x*x*y*y&xRange=0,1000000&yRange=0,1000000&resolution=25



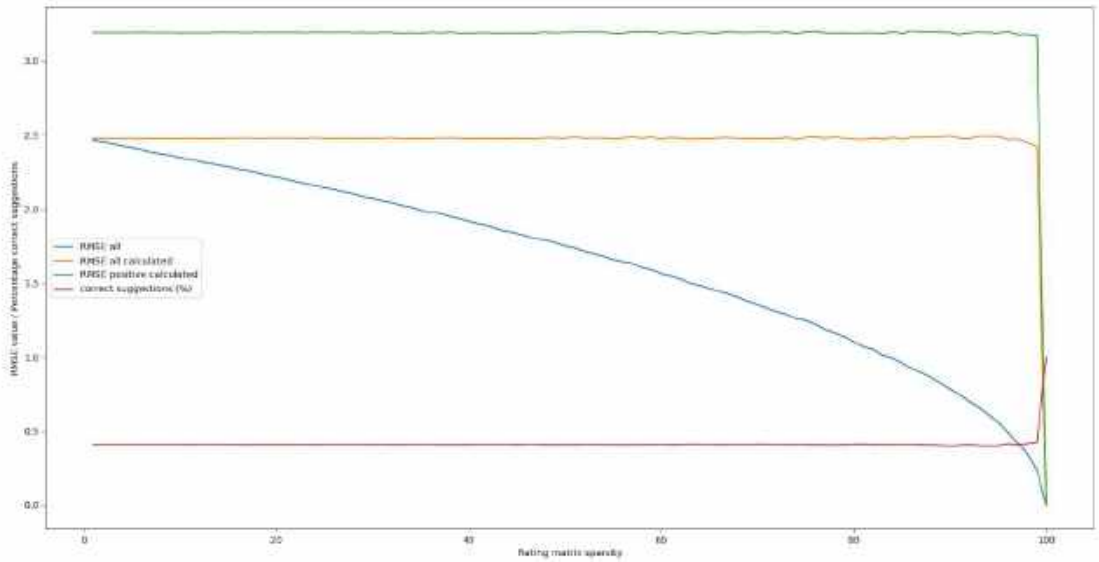
Figuur: complexiteit gebruikers vs. collaborative centered cosinus similarity
academo.org/demos/3d-surface-plotter/?expression=x*x*y*y*y&xRange=0,1000000&yRange=0,1000000&resolution=25

Zoals te verwachten stijgen alle grafieken kwadratisch op basis van de invoer grootte *(het aantal gebruikers en het aantal items)*. Opvallend is dat collaborative filtering duidelijk duurder is dan de andere methoden. Met name de centered cosine similarity in een collaborative methode is de meest dure implementatie van de hier opgelijste alternatieven. Niet toevallig geeft deze werkwijze de beste resultaten voor de test dataset die hier gebruikt werd.

4.4 Accuraatheid recommendation algoritme

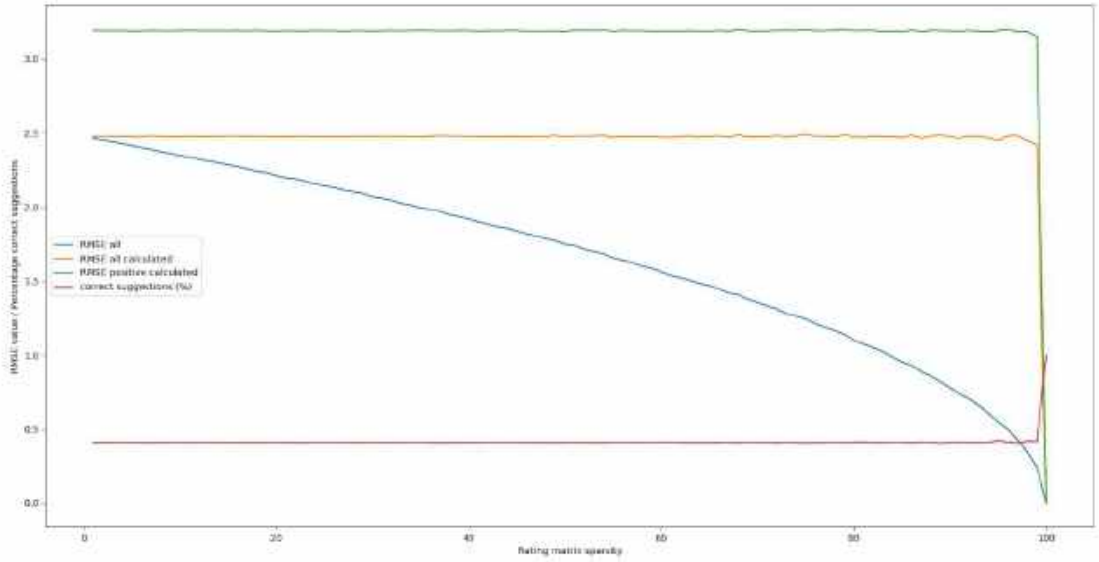
4.4.1 Collaborative filtering: item-item relaties

Grafiek 1: - Collaborative Item-item filtering | Jaccard similarity



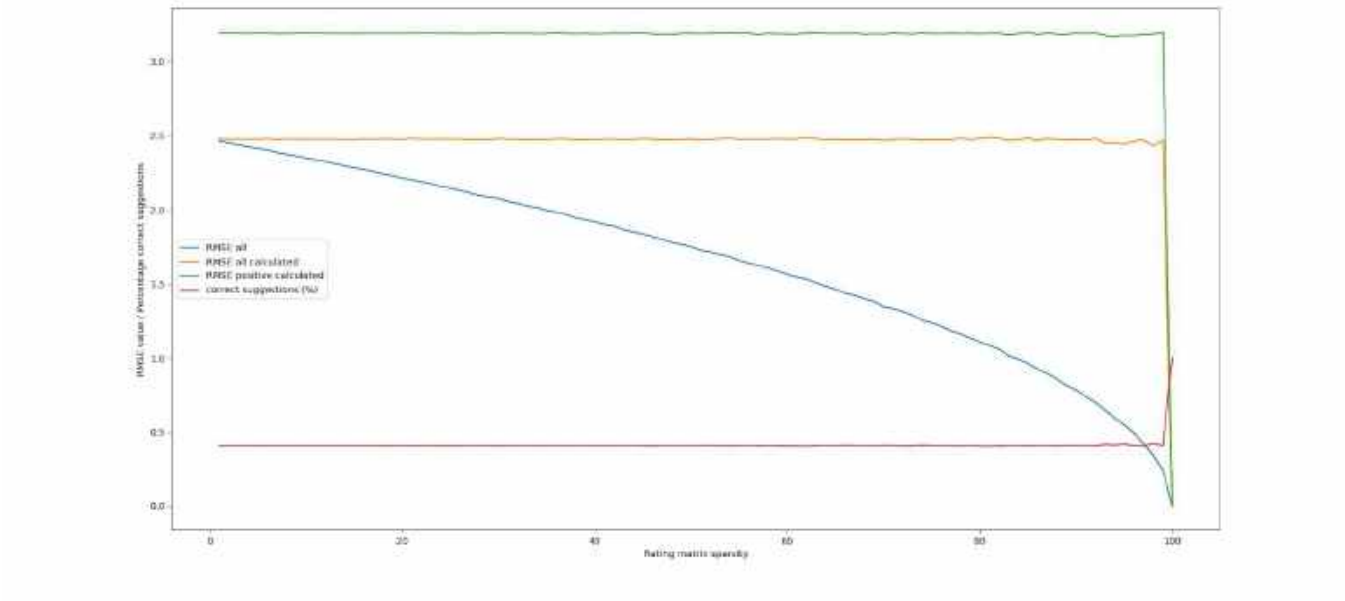
Outliners mee opgenomen in het resultaat
Geen global baseline gebruikt
Alle gebruikers met positieve similarity in de catalogus als neighbours beschouwd

Grafiek 2: - Collaborative Item-item filtering | Cosine similarity



Outliners mee opgenomen in het resultaat
Geen global baseline gebruikt
Alle gebruikers met positieve similarity in de catalogus als neighbours beschouwd

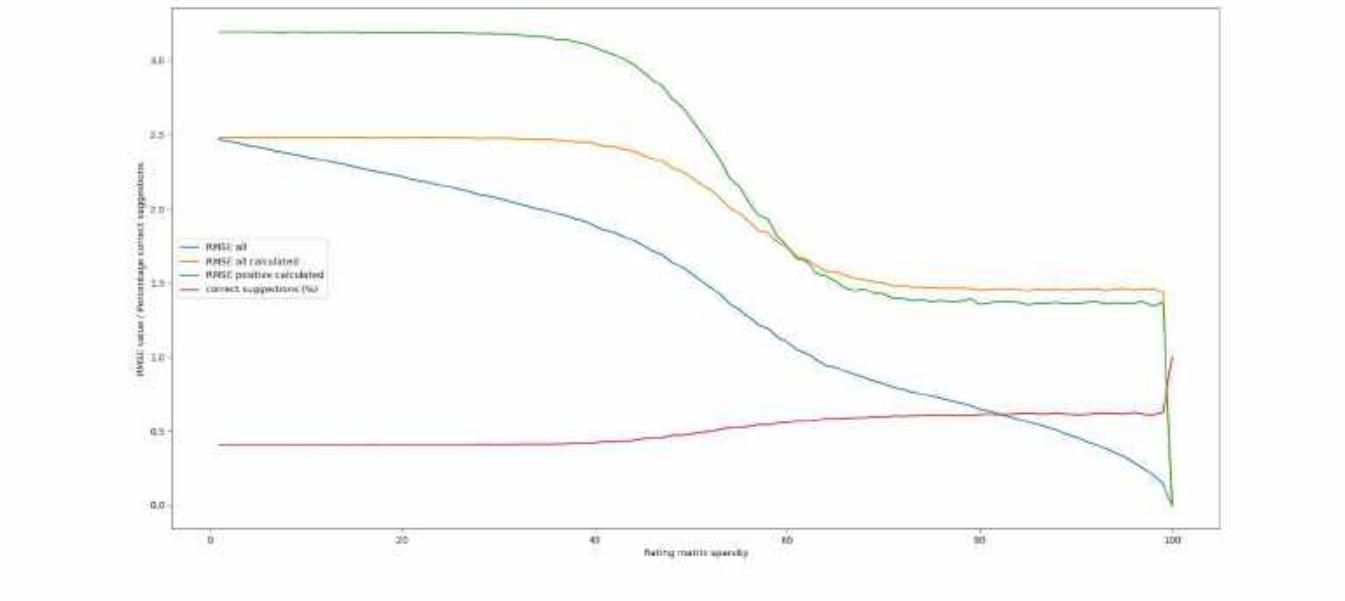
Grafiek 3: - Collaborative Item-item filtering | Centered cosine similarity



Outliners mee opgenomen in het resultaat
Geen global baseline gebruikt
Alle gebruikers met positieve similarity in de catalogus als neighbours beschouwd

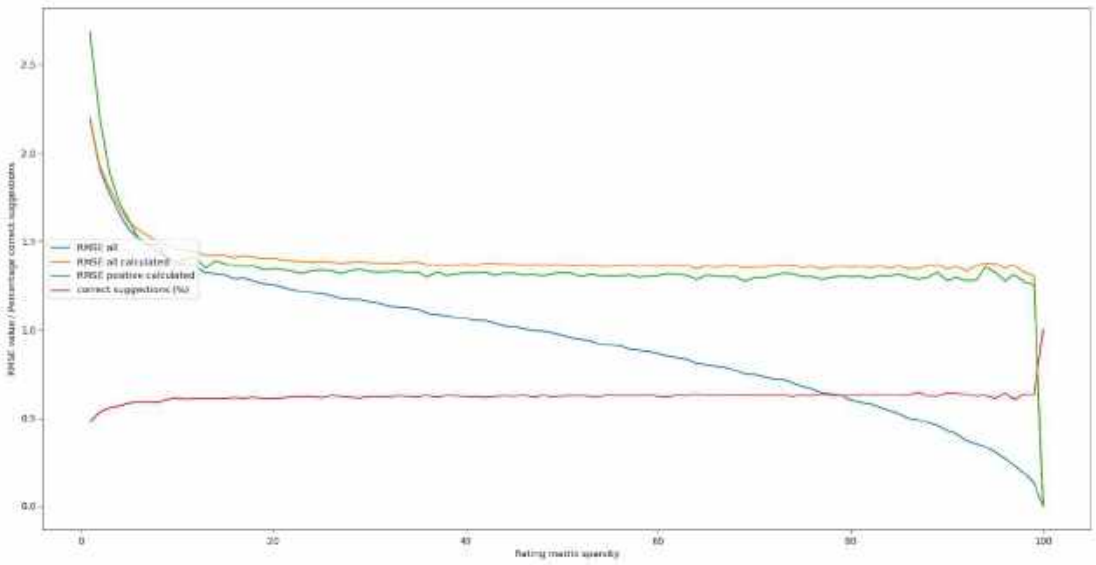
4.4.2 Collaborative filtering: user-user relaties

Grafiek 4: - Collaborative user-user filtering | Jaccard similarity



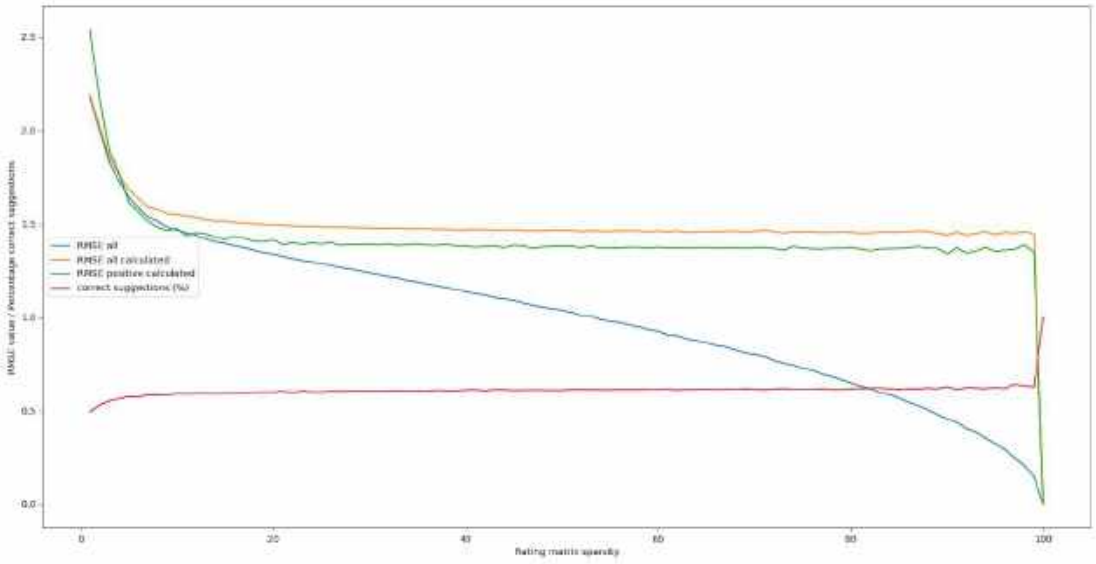
Outliners mee opgenomen in het resultaat
Geen global baseline gebruikt
Alle gebruikers met positieve similarity in de catalogus als neighbours beschouwd

Grafiek 5: - Collaborative user-user filtering | Cosine similarity



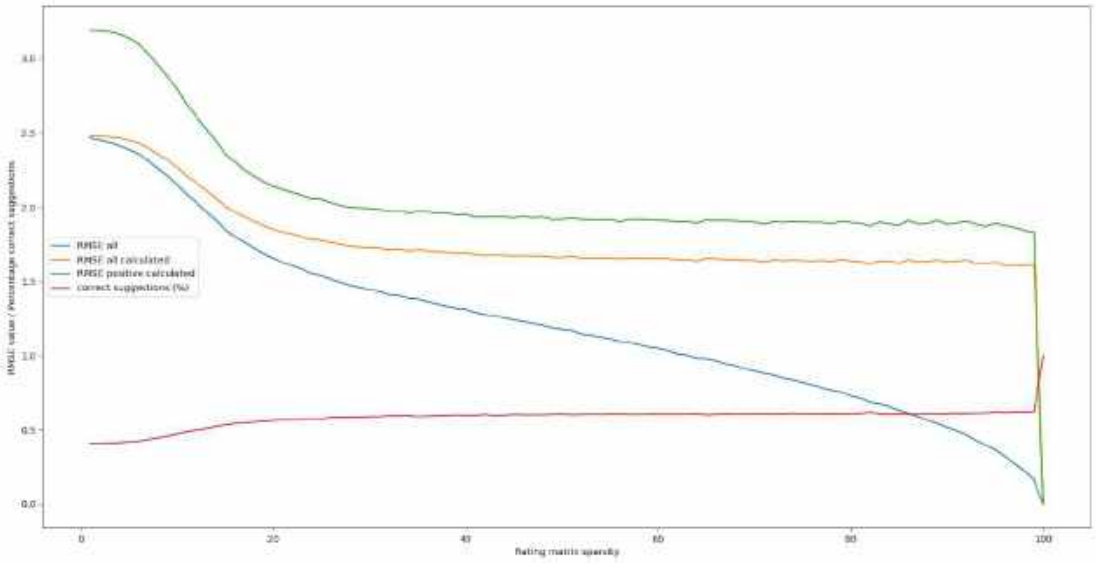
Outliners mee opgenomen in het resultaat
Geen global baseline gebruikt
Alle gebruikers met positieve similarity in de catalogus als neighbours beschouwd

Grafiek 6: - Collaborative user-user filtering | Centered cosine similarity



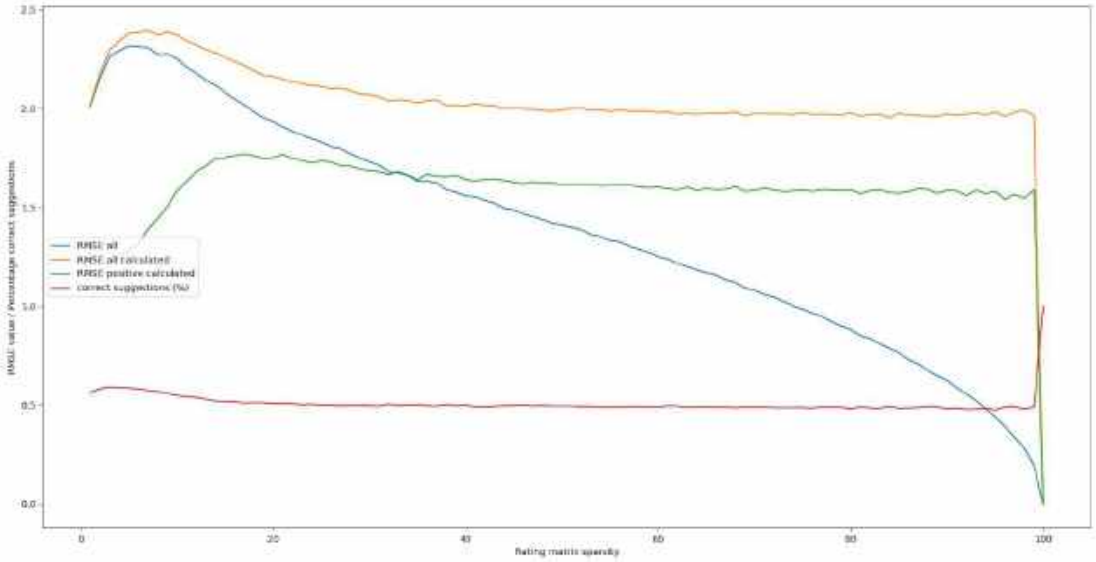
Outliners mee opgenomen in het resultaat
Geen global baseline gebruikt
Alle gebruikers met positieve similarity in de catalogus als neighbours beschouwd

Grafiek 7: - Collaborative user-user filtering | Centered cosine similarity



Outliners mee opgenomen in het resultaat
Geen global baseline gebruikt
Enkel top 5 meest matchende neighbours beschouwd

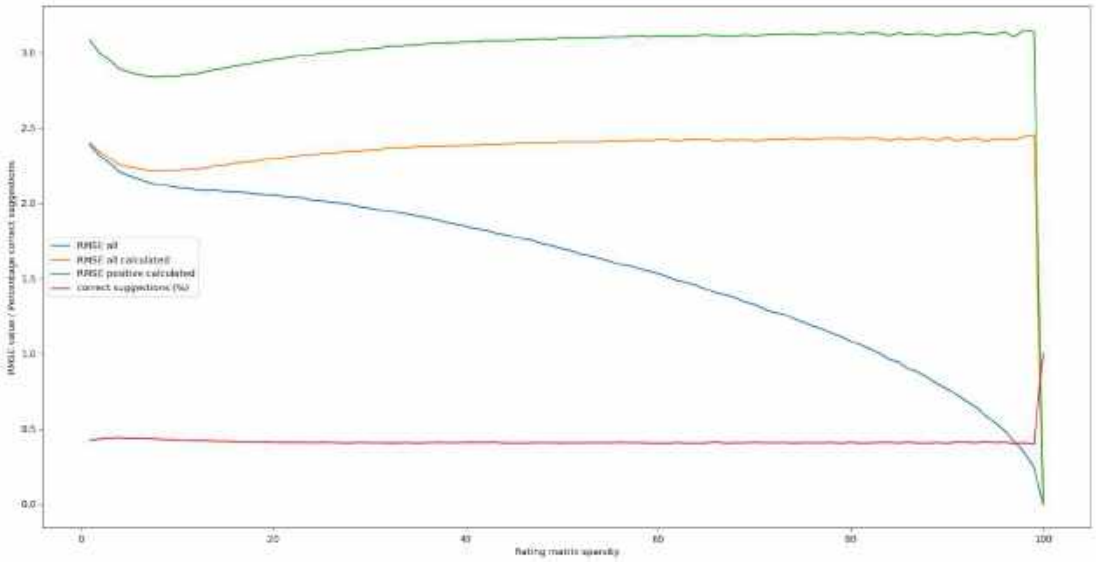
Grafiek 8: - Collaborative user-user filtering | Centered cosine similarity



Outliners mee opgenomen in het resultaat
Global baseline gebruikt in combinatie met gewogen gemiddelde van de neighbours
Alle gebruikers met positieve similarity in de catalogus als neighbours beschouwd

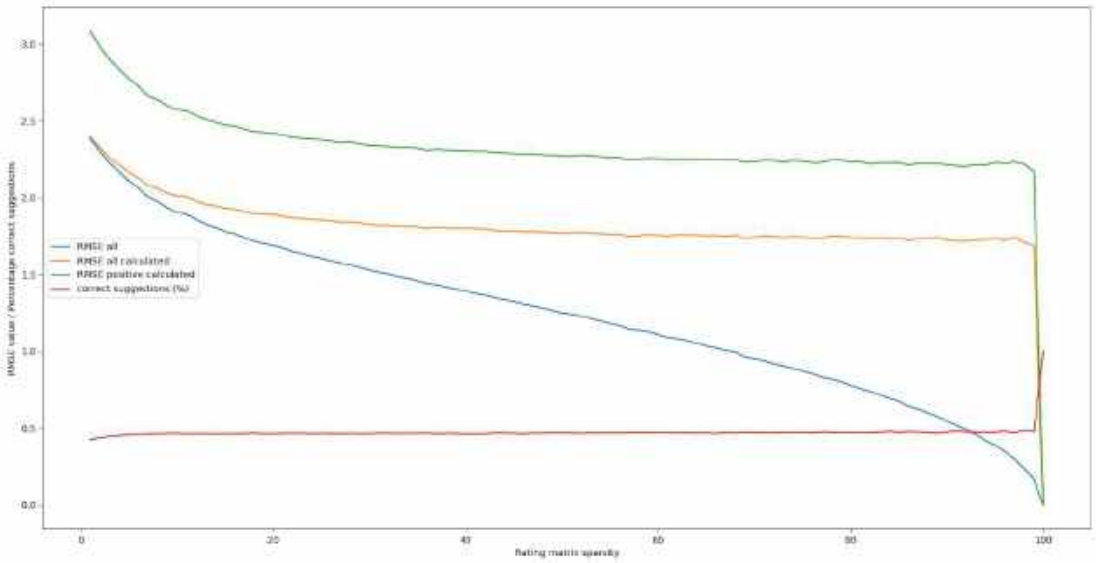
4.4.3 Content based filtering

Grafiek 9: - Content-based filtering | Jaccard similarity



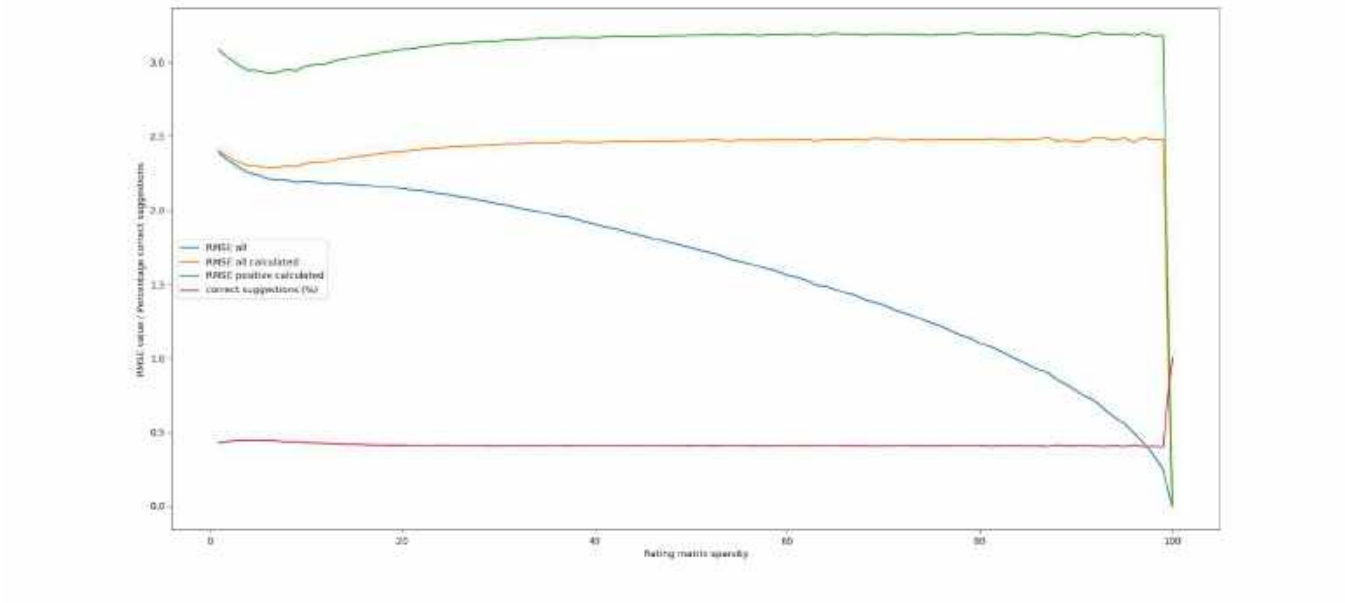
Outliners mee opgenomen in het resultaat
Geen global baseline gebruikt
Alle gebruikers met positieve similarity in de catalogus als neighbours beschouwd

Grafiek 10: - Content-based filtering | Cosine similarity



Outliners mee opgenomen in het resultaat
Geen global baseline gebruikt
Alle gebruikers met positieve similarity in de catalogus als neighbours beschouwd

Grafiek 11: - Content-based filtering | Centered cosine similarity



Outliners mee opgenomen in het resultaat
Geen global baseline gebruikt
Alle gebruikers met positieve similarity in de catalogus als neighbours beschouwd

4.5 Andere dataset

De metingen zijn specifiek voor de gebruikte dataset. Om een vergelijking te kunnen maken werd ook een test gedaan met een externe dataset. Hier werd geopteerd om de Jester dataset[52] te gebruiken van Ken Goldberg, professor aan de Berkley Universiteit in Californië. De geselecteerde variant is 'Dataset 2' [53].

Onderstaande test werd gedaan aan de hand van een user-user collaborative filtering, gebruikmakende van de cosine similarity. Als default waarde werd 0.0 gebruikt, het middelpunt van het rating interval. Er werd geen global baseline mee opgenomen in de berekening. Als neighbours werden enkel de top 5 beste matches beschouwd. At random werden enkele positieve beoordelingen weerhouden uit de dataset *(leave one out methode)* en opnieuw berekend. De gegenereerde waarden werden vergeleken met de effectieve waarden aan de hand van RMSE. Het resultaat van deze test vindt u in de schermafbeelding hieronder:

```
0.585 vs 3.78125 -> 3.19625
4.4062 vs 1.15625 -> 3.24995
3.4659 vs 0.3125 -> 3.1534
0.8077 vs 1.78125 -> 0.97355
1.4423 vs 6.0625 -> 4.6202
1.3667 vs 3.625 -> 2.2582999999999998
0.1021 vs 6.15625 -> 6.05415
0.7744 vs 1.65625 -> 0.8818500000000001
0.0 vs 5.0625 -> 5.0625
0.0 vs 7.5 -> 7.5
0.0 vs 5.96875 -> 5.96875
3.2493 vs 0.90625 -> 2.34305
2.0501 vs 7.4375 -> 5.3873999999999995
2.1113 vs 7.78125 -> 5.66995
0.0 vs 1.25 -> 1.25
0.0 vs 1.1875 -> 1.1875
-0.3884 vs 0.21875 -> 0.6071500000000001
0.9673 vs 6.5 -> 5.5327
1.3521 vs 3.25 -> 1.8979
0.6426 vs 3.84375 -> 3.20115
1.8375 vs 3.78125 -> 1.94375
-2.953 vs 0.375 -> 3.328
4.0305 vs 7.40625 -> 3.37575
0.0111 vs 8.75 -> 8.7389
0.7682 vs 9.9375 -> 9.1693
1.5972 vs 2.21875 -> 0.6215499999999997
-2.4934 vs 2.03125 -> 4.52465
0.3598 vs 0.21875 -> 0.14104999999999995
0.0 vs 3.53125 -> 3.53125
-1.1948 vs 9.25 -> 10.4448
0.4843 vs 4.15625 -> 3.67195
0.0 vs 8.8125 -> 8.8125
3.3294 vs 9.0 -> 5.6706
1.9175 vs 3.4375 -> 1.5199999999999998
0.0 vs 2.90625 -> 2.90625
2.2175 vs 1.625 -> 0.5925
1.1539 vs 4.8125 -> 3.6586000000000003
1.1164 vs 4.71875 -> 3.60235
0.0 vs 2.65625 -> 2.65625
1.9726 vs 7.125 -> 5.1524
----------------
Mean 4.6185 RMSE
----------------
```

[52] De Jester Collaborative Filter dataset kan teruggevonden worden op: ieor.berkeley.edu/~goldberg/jester-data

[53] De link naar deze dataset is: eigentaste.berkeley.edu/dataset

Opvallend is dat de RMSE waarde hier veel hoger ligt dan bij de evenementen dataset *(deze RMSE waarde is vergelijkbaar met de groene lijn in de grafieken)*. Een mogelijke verklaring hiervoor is dat er minder sterke user-user relaties in de dataset zijn of omdat de het rating interval groter is dan bij de evenementen. Bij de evenementen dataset is het rating interval [1, 5], terwijl dat bij de Jester dataset [-10, 10] is. De item catalogus is hier ook vele malen groter dan in de evenementen dataset. In de evenementen dataset waren er 33 items, terwijl dat er in de Jester dataset 150 zijn. Daar de meeste gebruikers de meeste items niet beoordeeld hebben kan het hier moeilijker zijn om een overlap te vinden in ratingen.

5. Conclusie

De kracht en noodzaak van recommender systems is duidelijk. Deze algoritmes kunnen nog steeds verbeterd worden om meer diverse aanbevelingen te geven aan gebruikers welke ook heel unieke gebruikersprofielen kunnen incorporeren. Ook de performantie van zulke systemen mag niet vergeten worden. Datasets worden alsmaar groter en hoewel ook computerkracht blijft toenemen, zijn snelle algoritmes, die zuinig met geheugen omspringen, een must. Gebruikers zijn gewoon aan real time interactie met platformen. Voor recommendation systemen is er geen tijd om enkele leesverzoeken naar een mechanische harde schijf te sturen, hoogstens een handvol naar een solid state oplossing. Om minimale latency te garanderen moeten algoritmes caching toelaten en zuinig omspringen met geheugen. Naar de toekomst toe zal het ook voor kleinere digitale platformen interessanter worden om recommender systemen te gaan inzetten. Het onderzoek naar continue verbetering van recommendation systemen is zeker nog niet afgelopen, een domein binnen de informatica waar zeker nog veel vooruitgang geboekt kan worden.

Dankwoord

Graag zou de auteur van deze bachelorproef ook enkele personen bedanken om de realisatie van deze paper mogelijk te maken.

- In de eerste plaats gaat mijn dank naar promotor prof. Jan Van den Bussche (Universiteit Hasselt) voor zijn begeleiding tijdens het onderzoek, het aanreiken van kwalitatieve literatuur en om de auteur in contact te brengen met onderzoekers binnen dit domein.

- Als tweede gaat mijn dank naar dhr. Olivier Jeunen (Universiteit Antwerpen) voor zijn ondersteuning op de technische vragen en de evaluatie van recommender systems bij te sturen.

- Ook een dankbetuiging aan dhr. Brecht Vandevoordt (Universiteit Hasselt) om de implementatie van similarity functies te willen bijsturen.

- Tenslotte een woord van dank aan Frans Vervoort en Yana Vanderperren om deze paper te willen nalezen en schrijffouten te verbeteren.

Formules gegenereerd met behulp van: codecogs.com/latex/eqneditor.php
3D-grafieken gegenereerd met: academo.org/demos/3d-surface-plotter
2D-grafieken gegenereerd met de library: matplotlib.org

Bronnen

Doorheen de tekst wordt er verwezen naar externe bronnen aan de hand van voetnoten. Echter zijn er ook algemene bronnen die geraadpleegd zijn om tot de globale inzichten in recommender systems en de algemene structuur van dit document te komen. Deze bronnen worden hieronder opgelijst:

Boek: Recomender Systems
Aggarwal, Charu C - 2016
Uitgegeven door: Springer

Artikel: Recommender System
Wikipedia

Online video cursus: Machine Learning
Andrew Ng, Stanford University, Coursera - 2013
coursera.org/learn/machine-learning

Hoorcollege: 8 Recommender Systems - Machine Learning Class 10-701
Prof. Alex Smola, Carnegie Mellon University - 2015
alex.smola.org

Cursus: Recommender Systems
Jeffrey D. Ullman, Stanford University
infolab.stanford.edu/~ullman/mmds/ch9.pdf

Blog post: How do you build a “People who bought this also bought that”-style recommendation engine
Roi Reshef, Data Science Made Simple - 16/12/015
datasciencemadesimpler.wordpress.com/tag/alternating-least-squares - Geraadpleegd op: 22/03/2018

Paper: Recommender systems: An overview
Guido de Nooij, Amsterdam University - 2008
beta.vu.nl/nl/Images/werkstuk-nooij_tcm235-91406.pdf - Geraadpleegd op: 15/05/2018

Paper: Knowledge-based recommender systems
Robin Burke, University of California
pdfs.semanticscholar.org/1b50/ff4e5d8420f3ffae7de2a060b5a6fd4b8023.pdf
Geraadpleegd op: 15/05/2018