



J-Fall

11 oktober 2007 Spant! - Bussum



Taste - collaborative filtering for Java

Pieter Kuijpers

luminis 



Amazon.com: Recommended for You

http://amazon.com/gp/yourstore/ref=pd_ys_tdy_recs/002-

amazon.com Pieter's Amazon.com See all 42 Product Categories Your Account | Cart | Your Lists | Help |

Your Browsing History | Recommended For You | Rate These Items | Improve Your Recommendations | Your Profile | Learn More

Search Amazon.com New Search GO Gift Certificates Web Search GO

Pieter's Amazon.com™ > Recommended for You
(If you're not Pieter, [click here.](#))

Recommendations by Category

- [Apparel & Accessories](#)
- [Baby](#)
- [Beauty](#)
- [Books](#)
- [Camera & Photo](#)
- [Computers & PC](#)
- [Hardware](#)
- [DVD](#)
- [Electronics](#)
- [Gourmet Food](#)
- [Grocery](#)
- [Health & Personal Care](#)
- [Home Improvement](#)
- [Industrial & Scientific](#)
- [Jewelry & Watches](#)
- [Kitchen & Dining](#)
- [MP3 Downloads](#)
- [Magazine Subscriptions](#)
- [Music](#)
- [Patio & Garden](#)
- [Software](#)
- [Sports & Outdoors](#)
- [Toys & Games](#)
- [Video](#)
- [Video Games](#)

Recommendations Based on Your Tags
([What's this?](#))

You currently have no tagged items

These recommendations are based on [items you own](#) and more.

view: **All** | [New Releases](#) | [Coming Soon](#) [More results](#)

- 

Golden Fool (The Tawny Man, Book 2)
by Robin Hobb (Dec 2, 2003)
Average Customer Review: ★★★★★ (70)
In Stock
Price: \$7.99
[87 used & new from \\$1.66](#) [Add to cart](#) [Add to Wish List](#)

I own it Not interested ☆☆☆☆☆ Rate it
Recommended because you rated **Fool's Errand (Tawny Man, Book 1)** and more ([Fix this](#))
- 

Fool's Fate (The Tawny Man, Book 3)
by Robin Hobb (Nov 23, 2004)
Average Customer Review: ★★★★★ (104)
In Stock
Price: \$7.99
[63 used & new from \\$4.01](#) [Add to cart](#) [Add to Wish List](#)

I own it Not interested ☆☆☆☆☆ Rate it
Recommended because you rated **Fool's Errand (Tawny Man, Book 1)** and more ([Fix this](#))
- 

Ship of Magic (The Liveship Traders, Book 1)
by Robin Hobb (Feb 2, 1999)
Average Customer Review: ★★★★★ (182)
In Stock
Price: \$7.99
[132 used & new from \\$0.75](#) [Add to cart](#) [Add to Wish List](#)

I own it Not interested ☆☆☆☆☆ Rate it
Recommended because you rated **Fool's Errand (Tawny Man, Book 1)** and more ([Fix this](#))

Klaar 4.134s 3 Errors

Zodra je een keer een aankoop bij Amazon.com hebt gedaan, houdt Amazon dat bij en begint je aanbevelingen te geven voor boeken die je misschien ook interesseren. Amazon geeft aanbevelingen op basis van je aankopen, maar je kan ook andere boeken een waardering geven van 1 tot 5 sterretjes die dan gebruikt worden voor de aanbevelingen.



A screenshot of the Last.fm 'My Recommendation Dashboard' in a web browser. The browser address bar shows 'http://www.last.fm/dashboard/music/?recs_sysdefault=1'. The dashboard has a navigation menu with 'Overview', 'Music', 'Videos', 'Events', 'Users', 'Reading', and 'Reply Tracker'. The 'Music' tab is active. The main content area is titled 'Manage your music recommendations' and includes checkboxes for 'From Last.fm', 'From your friends', and 'From your groups (filter groups)'. Below this is a 'Play my Recommendations' button. The 'Recommendations from Last.fm' section features a slider between 'obscurer' and 'popular' and a list of 14 recommended artists with their similar artists. The sidebar on the left contains sections for 'Shortcuts', 'New To My Profile', 'Last Scrobbled', 'Most Recent Shout', and 'Recently Loved'. The status bar at the bottom shows 'Klaar' and a system clock of '4.707s'.

Een ander voorbeeld is Last.fm. Met Last.fm kan je registreren welke muziek je draait, met iTunes of welke player je ook gebruikt, en gebruikt dat om andere gebruikers te zoeken die dezelfde smaak in muziek hebben als jij. En daar worden dan ook weer aanbevelingen gedaan voor muziek die je nog niet kent.



Google News Nederland – Aanbevolen

pieter.kuijpers@gmail.com | Zoekgeschiedenis | Mijn account | Afmelden

Het Internet Afbeeldingen Nieuws Maps Nieuw! Discussiegroepen meer »

Google Nieuws zoeken Het web doorzoeken

Nieuws Nederland BETA Zoek en blader in 400 continu bijgewerkte nieuwsbronnen.


Voorpagina


- Aanbevolen
- Arnhem
- Nederland
- Economie
- Wetenschap
- Buitenland
- Meest gelezen


Nieuwsmeldingen

- Mobiel nieuws
- Informatie over Google Nieuws

Aanbevolen

 **Hirsi Ali keert terug naar Nederland**
NRC Handelsblad - 12 uur geleden
Amsterdam, 1 okt. Ayaan Hirsi Ali keert terug naar Nederland omdat minister Hirsch Ballin (Justitie, CDA) niet meer bereid is de kosten voor haar beveiliging in de Verenigde Staten te betalen. De bescherming van het voormalige VVD-Kamerlid in Amerika ...
[Hirsi Ali werft fondsen in binnen- en buitenland](#) Volkskrant
[Balkenende: 'Hirsi Ali had eerder eigen initiatief moeten nemen'](#) Stentor
[DePers.nl](#) - [Trouw](#) - [BN/De Stem](#) - [Elsevier](#)
[alle 468 soortgelijke »](#)

 **Bankentrio zal bij ABN op eieren moeten lopen**
De Financiële Telegraaf - 10 uur geleden
AMSTERDAM (DFT) - De definitieve eindstand laat nog even op zich wachten, maar de wedstrijd is gespeeld. Het bankentrio Fortis, Royal Bank of Scotland en Santander heeft het maandenlange gevecht om ABN Amro gewonnen en komt volgende week met de ...
[Strijd ABN Amro beslecht](#) Parool
[Barclays trekt bod op ABN Amro in wegens onvoldoende steun](#) Volkskrant
[Provinciale Zeeuwse Courant](#) - [Elsevier](#) - [Algemeen Dagblad](#) - [BN/De Stem](#)
[alle 363 soortgelijke »](#)

 **Oranjeverenigingen boos om uitspraken Máxima**
Elsevier - 11 uur geleden
Prinses Máxima heeft 'een grote vergissing' begaan door de Nederlandse identiteit en de Nederlandse cultuur te relativeren. Dat vindt de Bond van Oranjeverenigingen, meldt De Telegraaf vandaag. Volgens voorzitter Michiel Zonneville viel de achterban ...
['Máxima vervreemd van gewone mensen'](#) Trouw
[Máxima betreurt ophef over speech](#) Volkskrant
[Stentor](#) - [De Telegraaf](#) - [RTL Z](#) - [NRC Handelsblad](#)
[alle 28 soortgelijke »](#)

Klaar 1.507s

Nog een voorbeeld is Google News: die kan je ook aanbevelingen doen op basis van de nieuwlinks waar je eerder op hebt geklikt.



Collaborative filtering

Het algemene probleem is dat een gebruiker is geïnteresseerd in informatie, boeken die hij wil lezen, muziek die hij wil luisteren, nieuws dat hij wil lezen. Als hij een goed idee heeft van waar hij naar op zoek is, gebruikt hij gewoon Google. Maar in deze situaties weet hij niet precies wat hij wil. De zoekvraag is dan eigenlijk, 'Laat mij de dingen zien die voor mij interessant zijn'. Traditionele search engines kunnen die vraag niet beantwoorden. Collaborative filtering is een manier om dergelijke vragen toch te kunnen beantwoorden.

'Filtering' slaat op het filteren van informatie, dus het selecteren van de juiste informatie uit een grote verzameling. 'Collaborative' slaat op het feit dat de informatie die wordt gebruikt voor het filteren, door alle gebruikers van het systeem samen wordt aangeleverd. In het voorbeeld van Amazon zijn dat de aankopen die jij en andere klanten hebben gedaan, en de waardering die je aan boeken hebt gegeven. Bij last.fm is het de muziek die jij en de andere gebruikers hebben gedraaid.



Nodig voor collaborative filtering:

- grote hoeveelheid informatie-items
- voldoende grote groep gebruikers
- manier om te meten welke informatie de gebruikers interessant vinden
- een correlatie tussen de items

Collaborative filtering kan gebruikt worden in situaties waar er sprake is van:

- een grote hoeveelheid informatie-items
in ieder geval te groot voor een gebruiker om alles zelf te bekijken
- een voldoende grote groep gebruikers
zo groot dat er altijd overlap is tussen de voorkeuren van gebruikers
- een manier om te meten welke informatie de gebruikers interessant vinden
bijvoorbeeld expliciete beoordelingen, of impliciete beoordelingen: als een klant een boek heeft besteld bij Amazon, vindt hij dat boek blijkbaar interessant. Als een gebruiker in google news op een bepaald nieuws item heeft geklikt, vindt hij dat item schijnbaar interessant. De kwaliteit van deze meetgegevens heeft natuurlijk effect op de kwaliteit van de aanbevelingen: expliciete beoordelingen zijn beter dan impliciete. Iemand kan een boek kopen bij Amazon dat hij na lezen toch niet goed vindt, of aan de andere kant: gebruikers van Google News lezen voor hen interessante koppen op de homepage van Google News, zonder erop te klikken.

- een correlatie tussen de items

De items moeten iets met elkaar te maken hebben. Het moet mogelijk zijn om een gebruiker die item A interessant vindt, item B aan te bevelen omdat item B in een of ander opzicht op item A lijkt. Je hoeft niet precies te weten wat die correlatie is! Die kan je ook afleiden uit de voorkeuren van andere gebruikers. Maar als er helemaal geen correlatie is, als iedere gebruiker een willekeurige set van items interessant vindt, valt er niets aan te bevelen. En er geldt: hoe hoger de correlatie, hoe makkelijker het is om goede aanbevelingen te doen.



Toepassingen

- Aanbevelingen doen
- ‘Buren’ zoeken
- Vergelijkbare items zoeken

Mogelijke toepassingen voor collaborative filtering zijn:

- Aanbeveling van items die gebruiker nog niet kent, zoals Amazon
- Zoeken van 'buren' -> andere gebruikers met vergelijkbare interesse, zoals Last.fm
- Zoek naar vergelijkbare items, zoals amazon doet met 'Better together' => bij aanschaf boek een speciale aanbieding voor een boek dat erop lijkt.
- Andere kant op: zoeken van gebruikers die waarschijnlijk geïnteresseerd zijn in een bepaald item. Kan interessant zijn voor bijvoorbeeld de marketing afdeling.



Probleem: Voorspellen beoordeling

Het probleem dat een collaborative filtering algoritme probeert op te lossen is

voorspel de beoordeling van items voor een bepaalde gebruiker, gebaseerd op een verzameling van beoordelingen van andere gebruikers

Het gaat dus om het voorspellen van de beoordeling van een combinatie van gebruiker en item. Als je die hebt kan je lijsten maken van de hoogste verwachte beoordelingen per gebruiker, of de gebruikers met naar verwachting de hoogste beoordeling voor een bepaald item.



	Gladiator	Rocky II	The Sixth Sense	Closer
Jaap	5	2	5	4
Fred	2	5		3
Hans	2	2	4	2
Ton	5	1	5	?

Voorbeeld:

Geef een voorspelling voor de beoordeling van 'Closer' voor Ton, gegeven de beoordelingen van Ton voor Gladiator, Rocky II en The Sixth sense, en gegeven de beoordelingen van Jaap, Fred en Hans.



Algoritmen

Het is een onderwerp waar behoorlijk wat onderzoek naar gedaan is, als je bijvoorbeeld zoekt op scholar.google.com vind je duizenden research papers. En zolang je zelf niet een recommendation engine wil gaan maken hoef je niet in detail te weten hoe ze allemaal werken. Maar toch is het belangrijk om een globaal idee te hebben van hoe ze werken, omdat je voor je eigen toepassing ten eerste een algoritme moet kiezen, en vervolgens moet fijnafstellen op je eigen data. Bovendien hebben de verschillende algoritmes andere eigenschappen voor wat betreft performance, startup tijd, omgaan met nieuwe data enzovoort.



Neighborhood-based

Een veel voorkomende klasse van algoritmen is Neighborhood-based. Een neighborhood-based algoritme werkt door het kiezen van een groep gebruikers op basis van hun gelijkheid met de actieve gebruiker. Het gewogen gemiddelde van de waarderingen van die groep wordt dan gebruikt voor het voorspellen van een waardering van een bepaald item voor de actieve gebruiker.



	Gladiator	Rocky II	The Sixth Sense	Closer
Jaap	5	2	5	4
Fred	2	5		3
Hans	2	2	4	2
Ton	5	1	5	?

In dit voorbeeld zie je dat Jaap de meeste overeenkomsten heeft met Ton voor wat betreft de beoordelingen voor Gladiator, Rocky II en The Sixth Sense. Het oordeel van Jaap voor Closer zal daarom het zwaarst meewegen bij de voorspelling van de beoordeling van Closer voor Ton. De beoordelingen van Fred en Hans worden ook meegenomen, maar wegen minder zwaar in de voorspelling voor Ton.



Neighborhood-based

1. Bepaal overeenkomst gebruikers

Het algoritme bestaat uit de volgende stappen:

1. Bepaal de overeenkomst van alle gebruikers met de huidige gebruiker

Hiervoor wordt vaak het gemiddelde verschil in beoordeling gebruikt voor alle items die beide gebruikers hebben beoordeeld. Als jij dezelfde items een hoge waardering geeft als gebruiker B, dan heeft B een hoge overeenkomst met jou, en andersom als gebruiker C lage waarderingen geeft aan items die jij hoog hebt gewaardeerd, heeft gebruiker C een lage overeenkomst.

Vaak wordt daarbij ook meegenomen wat de gemiddelde beoordeling voor alle items van een bepaald persoon is: er zijn namelijk gebruikers die nooit de beoordeling 1 en 5 gebruiken (in een schaal van 1 tot 5) en andere gebruikers die juist de hele range gebruiken. Een 4 van die eerste persoon heeft dan eigenlijk een vergelijkbare waarde als de 5 van de andere persoon. Je kan dat oplossen door niet te kijken naar de absolute waardering van de personen, maar de afwijking van zijn eigen gemiddelde waardering.



Neighborhood-based

1. Bepaal overeenkomst gebruikers
2. Kies een groep gebruikers

2. Kies een groep gebruikers die dient als voorspeller

Meestal is dat de top zoveel van dichtsbijzijnde gebruikers, dus de gebruikers die het meest overeenkomen met de actieve gebruiker. Maar wat ook kan is om een bepaalde grenswaarde te nemen voor de overeenkomst, en alle gebruikers te nemen die boven die grens liggen.



Neighborhood-based

1. Bepaal overeenkomst gebruikers
2. Kies een groep gebruikers
3. Voorspel beoordeling voor actieve gebruiker

3. Gebruik een gewogen combinatie van de beoordelingen van de geselecteerde groep om de beoordeling voor de actieve gebruiker te voorspellen

Hoe groter de overeenkomst tussen een gebruiker en de actieve gebruiker is, des te zwaarder weegt de beoordeling van die gebruiker voor een item.

Alle drie de stappen kunnen natuurlijk op verschillende manieren worden uitgevoerd, en dat is ook waarin de verschillende neighborhood-based algoritmes van elkaar afwijken. Zo kan je de overeenkomst tussen gebruikers bepalen zoals ik al zei, door het nemen van het gewogen gemiddelde van de verschillen in beoordeling, maar een andere manier die ook wel wordt gebruikt is om de bekende beoordelingen van twee users als een vector te beschouwen, en dan de hoek tussen die vectoren uit te rekenen.

Voor het kiezen van de groep voorspellende gebruikers noemde ik al twee verschillende methoden: de n dichtsbijzijnde burens en de burens met een afstand boven een bepaalde grens.

En voor het berekenen van de voorspelde beoordeling zelf kunnen een aantal optimalisaties worden toegepast. Zo kan je beoordelingen voor items die relatief weinig door andere gebruikers beoordeeld zijn zwaarder laten meewegen dan items die door veel gebruikers zijn beoordeeld. Of je kan extreme beoordelingen, zoals een 1 of een 5, zwaarder laten meewegen dan gemiddelde beoordelingen.



Item-based

Normaal gesproken werkt het algoritme op basis van gelijkenis tussen verschillende gebruikers. Een variant is om in plaats van de overeenkomsten tussen gebruikers, overeenkomsten tussen items te nemen. De werking van zo'n item-based algoritme is vergelijkbaar met het user-based algoritme, maar je kijkt op een andere manier tegen je data aan. In plaats van het zoeken van de groep meest gelijkende gebruikers bij een gebruiker, zoek je de meest gelijkende groep items bij een item. Voordeel hiervan is dat het soms makkelijker is om de gelijkenis tussen items te berekenen: in het geval van boeken zou je bijvoorbeeld kunnen kijken naar schrijver en genre. Het is ook beter schaalbaar bij grote aantallen gebruikers: de performance is niet meer afhankelijk van het aantal gebruikers, maar van het aantal items. En het aantal items is vaak een minder veranderlijk dan het aantal gebruikers, zodat het een optie is om de gelijkenissen tussen items vooraf te berekenen, zodat dat op runtime niet meer hoeft.



Nadelen neighborhood-based:

- Complex
- Veel beoordelingen nodig
- Runtime performance

Een neighborhood algoritme heeft een aantal nadelen: het is vrij complex om te implementeren en te onderhouden, het werkt niet zo goed als je maar weinig beoordelingen hebt, en het is op runtime niet zo snel, omdat alle gewichten elke keer opnieuw berekend moeten worden.



Slope one

Een ander algoritme dat probeert die nadelen te compenseren is het Slope One algoritme.



'Popularity differential'

Het slope one algoritme is gebaseerd op het idee van 'popularity differential' – het verschil in voorkeur tussen twee items. Stel dat een gebruiker twee films heeft beoordeeld, Titanic en Spiderman, en dat hij Titanic een waardering van 3 heeft gegeven, en Spiderman een waardering van 4. Een andere gebruiker heeft alleen Titanic beoordeeld met een 4. Omdat Spiderman gemiddeld 1 punt meer heeft dan Titanic, is de voorspelde beoordeling van de tweede gebruiker voor Spiderman dus een 5.

Dat klinkt heel flauw voor het geval van twee gebruikers en twee films, maar het wordt al anders als je een groot aantal films en een groot aantal gebruikers gaat gebruiken. Wat je dan als eerste gaat doen is het bepalen van het gemiddelde verschil in voorkeur tussen alle combinaties van items i en j over alle gebruikers die beide items hebben beoordeeld. Het resultaat is een matrix met voor elke combinatie i en j .

De verwachte beoordeling voor item i gegeven item j is dan de de beoordeling voor film i + de waarde uit de matrix bij (i,j) . Dat doe je voor alle items die de gebruiker heeft beoordeeld.

De uiteindelijke voorspelling is dan het gemiddelde van al die voorspellingen.

Als er genoeg data is, dwz voor (bijna) alle paren (i,j) is er minstens 1 beoordeling, is een goede benadering van de voorspelling voor de beoordeling van j :
de gemiddelde beoordeling van alle items door de user beoordeeld + (de gemiddelde dev_{i_j} over alle beoordeelde items i)



	Gladiator	Rocky II	The Sixth Sense	Closer
Jaap	5	2	5	4
Fred	2	5		3
Hans	2	2	4	2
Ton	5	1	5	?

Voorbeeld: het verschil in beoordeling tussen The Sixth Sense en Closer is -1 voor Jaap en -2 voor Hans, gemiddeld dus $-1,5$. Op basis van alleen deze twee films is de voorspelling voor Ton dus 5 (beoordeling The Sixth Sense van Ton) $- 1,5$ (gemiddelde verschil) is $3,5$.

Hetzelfde kan je doen voor Gladiator en Rocky II, om uiteindelijk op de gemiddelde voorspelde waardering te komen.



Differential matrix

Het mooie van dit algoritme is dat de matrix vooraf berekend kan worden, en daarna voor elke voorspelling opnieuw gebruikt worden. Je kan je voorstellen dat het bepalen van een beoordeling vrij snel uitgevoerd kan worden als je eenmaal die matrix met verschillen hebt berekend, en dat blijkt in de praktijk ook zo te zijn. En ook is het een minder complex om te bouwen, en het is niet moeilijk om nieuwe beoordelingen in de matrix te verwerken zonder alles opnieuw te hoeven berekenen. Een ander voordeel is dat het ook nog goed blijkt te werken als een gebruiker nog maar weinig beoordelingen heeft gedaan.

Uit tests met de bekende MovieLens data blijkt dat de kwaliteit van de voorspellingen met het slope one algoritme vergelijkbaar is met de beste neighborhood-based algoritmes. Het is niet beter, maar ook niet slechter, terwijl het wel de voordelen heeft die ik net noemde, zoals snelheid en eenvoud.



- Voordelen slope one:
 - Eenvoud
 - Snelheid
 - Weinig beoordelingen nodig
- Nadelen:
 - Geheugengebruik
 - Opbouw matrix kost tijd

Zijn er dan ook nog nadelen aan dit algoritme? Ja, je vermoedt het waarschijnlijk al, het kost een hoop tijd om de matrix initieel op te bouwen. Dat hoeft nog niet zo'n probleem te zijn, want daarna is het behoorlijk snel. Maar ook kost het een hoop geheugen. Om snel te kunnen werken moet je eigenlijk de hele matrix in het geheugen houden, en als je veel items hebt kan dat aardig op gaan lopen. Je kunt natuurlijk de matrix in een file of database opslaan, maar dat gaat dan weer ten koste van de performance.



Model-based

Naast de algoritmes waar we het over hebben gehad bestaat er ook nog de klasse van de model-based algoritmes. Daarbij wordt vooraf een model gemaakt van de beoordelingen, en dat model wordt vervolgens gebruikt voor het voorspellen. Voorbeelden zijn algoritmes die gebruik maken van een Bayesian networks of een neurale netwerk. Ze hebben net als het slope one algoritme een tijd nodig om in te leren, namelijk om het model op te stellen, maar als dat eenmaal is gebeurd zijn ze behoorlijk snel. Er is geen groot verschil in de kwaliteit van de voorspellingen tussen de beste model-based algoritmes en de neighborhood based algoritmes. Maar aangezien deze algoritmes nog niet beschikbaar zijn in de Java library waar ik het zo over ga hebben, ga ik er hier verder niet op in.



Effectiviteit?

Het vergelijken van de verschillende algoritmes is een dankbaar onderwerp voor research, en er zijn dan ook verschillende onderzoeken gedaan naar de effectiviteit van de verschillende methoden. Daaruit komt dat de Correlatie algoritmes het over het algemeen goed doen: ze voorspellen de beoordeling op een schaal van 1 tot 5 met een gemiddelde fout van ongeveer 0,7. Met bepaalde optimalisaties kan je het iets verbeteren, maar dat gaat dan eerder om hondersten. Het slope one algoritme en model-based algoritmes op basis van Bayesian networks geven een vergelijkbaar resultaat.

Zwak punt van al deze onderzoeken is dat de meeste zijn uitgevoerd op 1 set testdata, namelijk de GroupLens database met filmbeoordelingen. Die dataset kan je namelijk gratis downloaden, en bevat een lekker grote hoeveelheid data (1 miljoen beoordelingen). Het kan dus zo zijn dat de onderzoeksresultaten anders zijn als ze op een andere dataset toegepast worden.

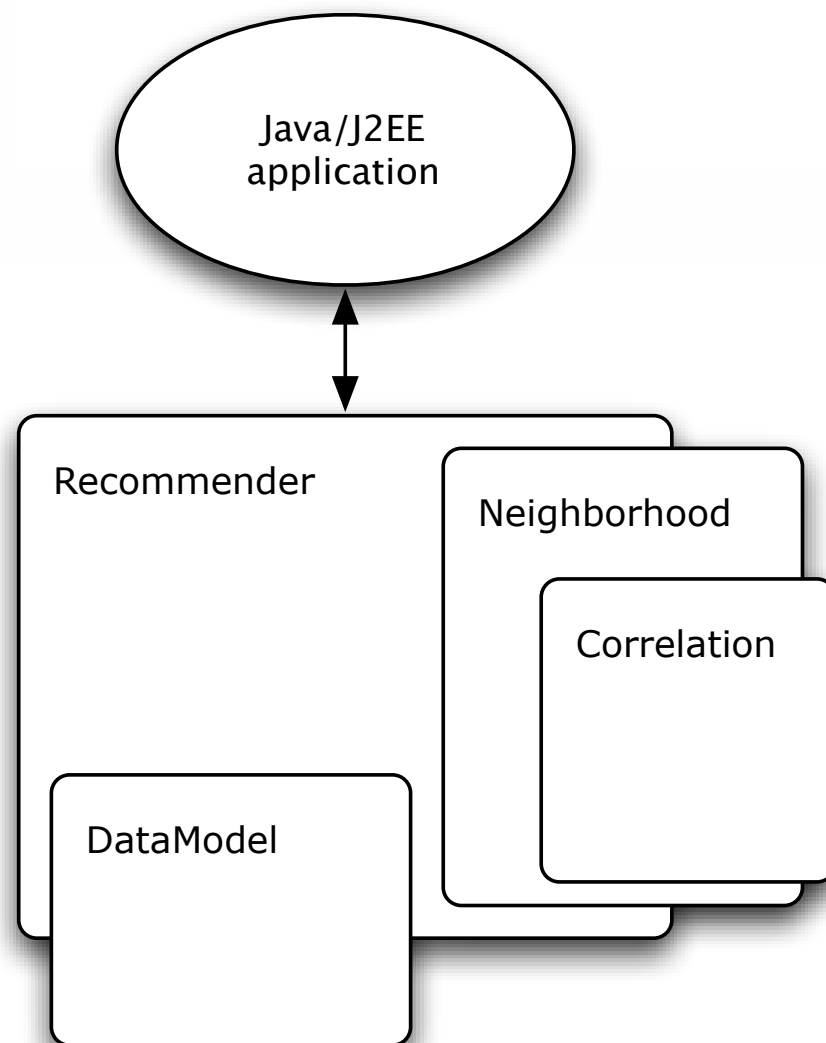
Het goede nieuws is dat het niet moeilijk is om zelf een aantal verschillende algoritmes en optimalisaties uit te proberen, en de resultaten te vergelijken. Ik zal dat straks laten zien. En dat is dan dus ook het advies: als je collaborative filtering op je eigen dat wil toepassen, doe dan experimenten met verschillende algoritmes en verschillende optimalisaties, en neem degene die voor jou het beste werkt.



Taste

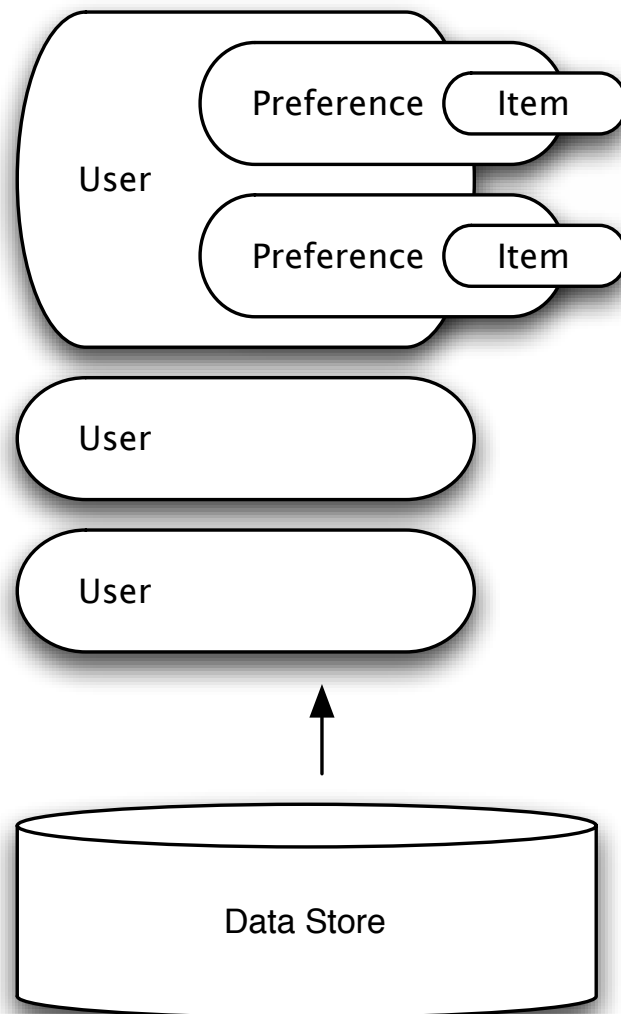
<http://taste.sourceforge.net>

Genoeg theorie, we gaan het nu hebben over hoe je dit allemaal in je eigen code kan toepassen. Er is een open source library voor collaborative filtering: 'Taste'. Het staat sinds mei 2005 op sourceforge, en ze zijn inmiddels bij versie 1.61. Je kan het gebruiken om een eigen Recommender te maken waarbij je de keuze hebt uit verschillende algoritmes.

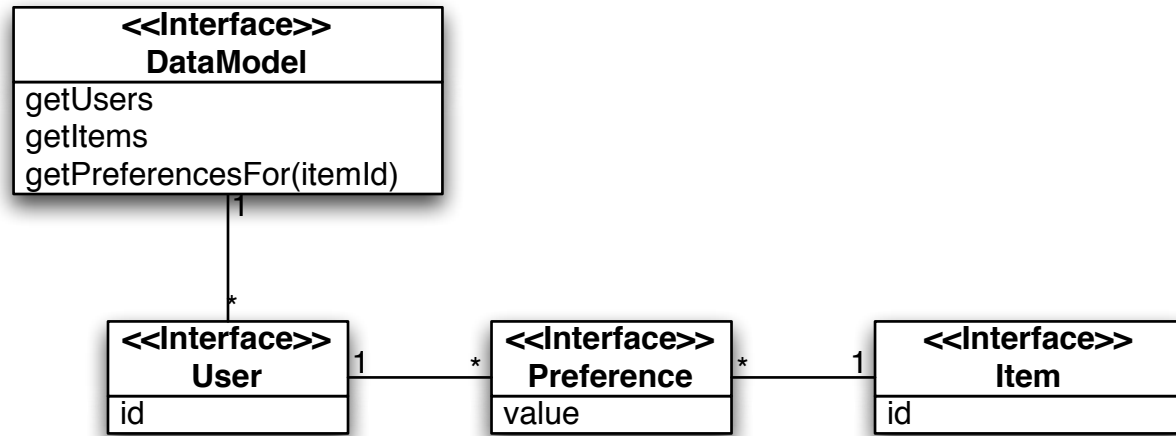


De belangrijkste abstracties zijn een aantal interfaces:

- Recommender is de recommender zelf. Dit is de interface die je aanspreekt vanuit je applicatie om aanbevelingen te krijgen en dergelijke.
- DataModel is een interface waar je je eigen implementatie aan moet geven. De data waar de aanbevelingen mee worden gedaan wordt met deze interface opgehaald.
- Neighborhood is de implementatie van het neighborhood algoritme dat je gebruikt, bijvoorbeeld dichtsbijzijnde N gebruikers.
- Correlation is de implementatie van het correlation algoritme dat je wil gebruiken.



Het DataModel bestaat uit Users, Preferences en Items. Elke User heeft een of meer Preferences, die gelden voor een Item. Het belangrijkste werk dat je zelf moet doen is het implementeren van het DataModel voor je eigen data. In de meeste gevallen zal je een database hebben waar je de data uit haalt. Reken er in dat geval op dat er veel gelijktijdige reads worden uitgevoerd, en zorg ervoor dat je een connection pool gebruikt. En natuurlijk indexen op de juiste plaatsen. Buiten die overwegingen is het vrij eenvoudig om het DataModel te implementeren.





Recommender

- **GenericUserBasedRecommender**
 - UserNeighborhood
 - UserCorrelation
- **GenericItemBasedRecommender**
 - ItemCorrelation
- **SlopeOneRecommender**
 - DiffStorage

Als je je DataModel hebt geïmplementeerd, is de volgende stap het instantiëren van een Recommender. Taste heeft een aantal implementaties beschikbaar, onder andere:

- GenericUserBasedRecommender
 - UserNeighborhood: NearestUserNeighborhood, ThresholdUserNeighborhood
 - UserCorrelation: PearsonCorrelation, SpearmanCorrelation
- GenericItemBasedRecommender
 - ItemCorrelation: PearsonCorrelation, GenericItemCorrelation

In de praktijk gebruik je de GenericItemCorrelation, omdat je dan de voordelen van de item based recommender benut door het gebruik van vooraf berekende overeenkomsten.

- SlopeOneRecommender
 - DiffStorage: MemoryDiffStorage, AbstractJDBCDiffStorage,

MySQLJDBCDiffStorage

De slope one recommender is het algoritme waarbij vooraf de matrix met voorkeurverschillen wordt opgebouwd. Je kan bij deze implementatie kiezen welke strategie wordt gebruikt voor het opslaan van die matrix: in memory, of in een database.



Recommender benchmark

- ItemAverageRecommender
- ItemUserAverageRecommender

En als benchmark voor het testen van je recommender:

- ItemAverageRecommender

Geeft als voorspelde beoordeling altijd de gemiddelde beoordeling voor een item over alle gebruikers

- ItemUserAverageRecommender

Geeft ook de gemiddelde beoordeling, maar dan gecorrigeerd voor de gemiddelde beoordeling voor die users.



```
DataModel datamodel = new GroupLensDataModel();

UserCorrelation correlation
    = new SpearmanCorrelation(datamodel);

UserNeighborhood neighborhood
    = new NearestUserNeighborhood(20, correlation, datamodel);

Recommender recommender
    = new GenericUserBasedRecommender(datamodel,
                                       neighborhood,
                                       correlation);

String userId = "10";
List<RecommendedItem> items = recommender.recommend(userId, 3);

printRecommendations(items);
```

Dit is een voorbeeld van hoe je een Recommender gebruikt als je eenmaal je DataModel hebt geïmplementeerd. Ik gebruik hier het GroupLensDataModel die als voorbeeld bij Taste zit gebundeld. Het gebruikt de MovieLens dataset met 1.000.000 beoordelingen voor films.



```
5-okt-2007 21:48:31 com.planetj.taste.impl.model.file.FileDataModel
<init>
INFO: Creating FileDataModel for file /tmp/taste.ratings.txt
5-okt-2007 21:48:31 com.planetj.taste.impl.model.file.FileDataModel
processFile
INFO: Reading file info...
5-okt-2007 21:48:35 com.planetj.taste.impl.model.file.FileDataModel
reload
INFO: Applying transforms...
3364Asphalt Jungle The (1950) Crime|Film-Noir (5.0)
3569Idiots The (Idioterne) (1998) Comedy|Drama (5.0)
13 Balto (1995) Animation|Children's (5.0)
```

Na wat logging zie je de drie films die worden aanbevolen.



Recommender evaluatie

Om de juiste combinatie van algoritmes voor je recommender te vinden wil je natuurlijk de kwaliteit van de aanbevelingen kunnen meten. Gelukkig is daar in Taste een makkelijke manier voor in de vorm van de RecommenderEvaluator. Die voert een soort automatische test op je recommender uit op basis van je data. De manier waarop hij dat doet is door je data te verdelen in een trainingsset en een testset. De trainingset gebruikt de recommender om aanbevelingen te doen. Die aanbevelingen worden vergeleken met de werkelijke beoordelingen uit de testset. Uiteindelijk berekent de RecommenderEvaluator een waarde uit die de kwaliteit van de recommender aangeeft, waarbij 0 perfect is.



```
RecommenderBuilder builder = new RecommenderBuilder() {
    public Recommender buildRecommender(DataModel dataModel)
        throws TasteException {

        return new GroupLensRecommender(dataModel);

    }
};

RecommenderEvaluator evaluator
    = new AverageAbsoluteDifferenceRecommenderEvaluator();

double eval = evaluator.evaluate(builder, MODEL, 0.9, 1.0);

System.out.println("Evaluation GroupLensRecommender is " + eval);
```

Hier zie je de code voor het uitvoeren van een RecommenderEvaluator. De RecommenderEvaluator verwacht een implementatie van een RecommenderBuilder, die de recommender maakt. In dit voorbeeld gebruik ik de Recommender die als voorbeeld bij Taste wordt meegeleverd, en die werkt op de MovieLens data met filmbeoordelingen. Erachter zit een Slope One recommender.



Evaluation GroupLensRecommender is 0.7097983224225899

Evaluation ItemAverageRecommender is 0.7828029973000493

Evaluation ItemUserAverageRecommender is 0.7335411613364233

Als je de recommender uitvoert krijg je dit resultaat. De gemiddelde afwijking tussen de voorspelde beoordeling en de echte beoordeling is 0.7. Ter vergelijking heb ik dezelfde test uitgevoerd met de ItemAverageRecommender, die als voorspelling altijd de gemiddelde beoordeling voor een item geeft. Je ziet dat de GroupLensRecommender iets beter werkt. De ItemUserAverageRecommender doet het ook aardig goed, terwijl die ook de gemiddelde beoordeling geeft, maar dan gecorrigeerd voor de gemiddelde beoordeling per gebruiker.

Door deze tests op je eigen data uit te voeren kan je dus een verstandige keuze maken voor een algoritme, of misschien concluderen dat het niet gaat werken voor jouw data.

Je weet nu voldoende om zelf aan de slag te gaan met Taste, en om een verstandige keuze te maken over hoe het in je architectuur past. Denk eraan dat Taste profijt heeft van veel geheugen, en dat het een processor intensieve taak is. Je bent het hopelijk met me eens dat het gebruik van de library niet moeilijk is als je iets weet van de achtergrond, en dat je genoeg mogelijkheden hebt om te experimenteren met verschillende algoritmes. Uiteindelijk is het grootste probleem dat je tegenkomt misschien wel de kwaliteit en kwantiteit van de onderliggende data, als je die eenmaal hebt dan kom je een heel eind.



Performance

Je grootste uitdaging zal zijn om een goede performance te halen met je Recommender en DataModel. Zolang het hele datamodel in het geheugen past is er niet zoveel aan de hand, maar bij grote hoeveelheden data is dat niet meer haalbaar. Dan zal je de gegevens uit een database moeten gaan halen, waarbij de betere database tuning skills van pas komen.



Vragen?