Voortgangsverslag Honoursproject

Thomas van den Berg en Maarten van der Velden

10 november 2009

Doelstellingen

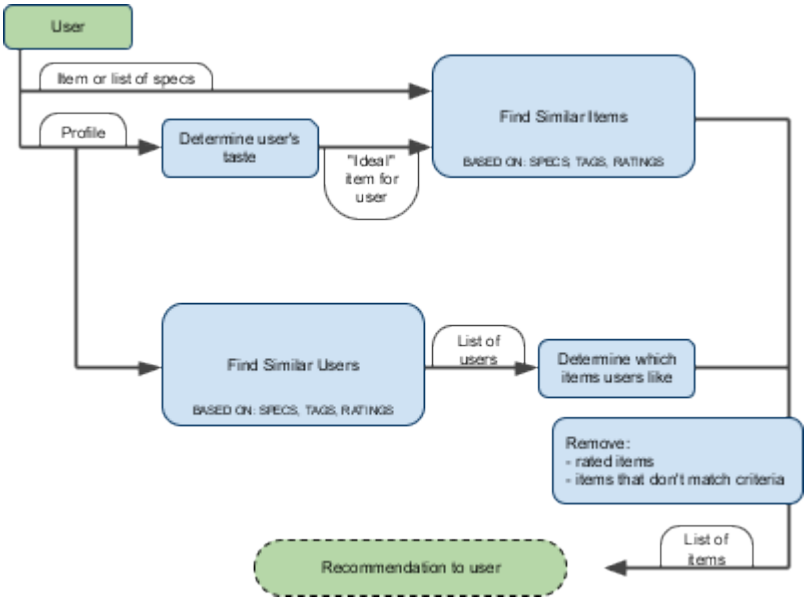
Het doel van dit honoursproject is om een opzet te maken voor een algemeen *recommender*systeem. Dit systeem moet aanbevelingen kunnen doen voor verschillende soorten items/producten op basis van een *user community.* De vraag hierbij is welke methoden er zijn om dit te doen en wat deze methoden kenmerkt. Door een standaardmodel voor *recommender*systemen te ontwikkelen willen we een overzicht creëren van wat er zoal op de markt is en daarnaast een gefundeerde keuze maken in onze eigen aanpak.

Aanpak

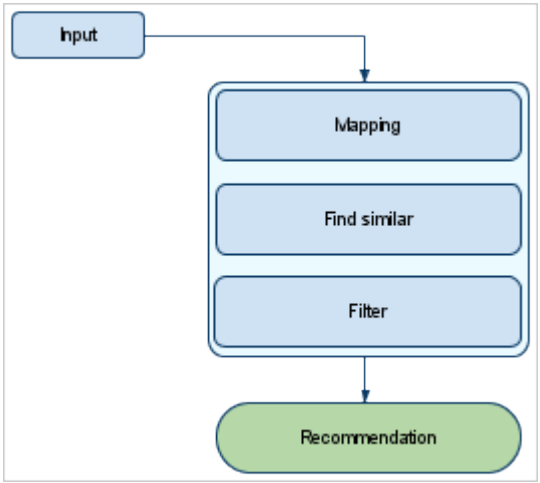
In de eerste weken van het project hebben we ons verdiept in bestaande toepassingen en technieken voor het maken van aanbevelingen. We hebben deze technieken met elkaar vergeleken en gezocht naar fundamentele overeenkomsten en verschillen. Een aantal methodes blijken bijvoorbeeld data te abstraheren tot getallen in een matrix, maar uiteindelijk komt het er vaak op neer dat een aanbeveling item based of user based gedaan wordt, zoals hieronder in het schema te zien is.

Je begint altijd bij een gebruiker die een aanbeveling wil (of krijgt). In een item based methode (Amazon) geeft de gebruiker criteria voor het object dat hij aanbevolen wil krijgen, of wordt een mapping gedaan op basis van de gegevens van die gebruiker om te zien wat het 'ideale' object van deze gebruiker is. Met dit model-object wordt een match gemaakt in de database met objecten om te zien welke objecten er het meest op lijken. Hiervoor worden specificaties vergeleken en vooral oordelen van gebruikers met tags en/ of ratings. De meest lijkende objecten worden dan aanbevolen na een eventuele filter-stap om objecten te verwijderen die de gebruiker al kent.

Een *user based* (last.fm) systeem doet een *match* op basis van het gebruikersprofiel om te zien welke andere gebruikers het meest op deze gebruiker lijken. Dit wordt gedaan op basis van beoordelingen van objecten door de gebruikers, gegeven tags en/of objectieve eigenschappen zoals leeftijd, e.d. Van de meest gelijkende gebruikers wordt dan bekeken welke objecten zij het best vinden, waarna de objecten die de gebruiker al kent eruit worden gefilterd.



We kwamen al snel tot de conclusie dat deze twee benaderingen eigenlijk meer overeenkomsten dan verschillen hebben. Het begint altijd met een gebruiker en eindigt met een aanbeveling, er vindt altijd een *match* plaats, er zit altijd ergens een filter in en er moet ergens een *mapping* gedaan worden. De plaats waar de *mapping* plaatsvindt bepaald het type *match*: Eerst een mapping van gebruiker naar object dat deze gebruiker representeert om de *match* mee te doen, of eerst een *match* en dan een *mapping* om van een groep users naar de ideale objecten te komen, zie illustratie hieronder.



Bij last.fm begint het proces met matchen: van een user wordt gekeken naar zijn buren, de mensen die het meest naar dezelfde muziek luisteren, vervolgens is er de mapping stap, van al deze buren wordt gekeken naar welke muziek *zij* luisteren, en tot slot vindt de filterstap plaats, namelijk het weglaten van de muziek die door de user al eens geluisterd is.

Bij een Amazon-achtig systeem begint het proces met een mapping, vanuit de gebruiker wordt gekeken naar welke boeken hij het leukst vindt (dat wordt geoperationaliseerd door welke boeken hij *gekocht* heeft), vervolgens wordt gekeken welke boeken lijken op deze boeken, de filterstap is natuurlijk weer het weghalen van boeken die de gebruiker al gekocht heeft.

Deze twee aanpakken geven eigenlijk precies het verschil tussen item-based en user-based filtering aan.

Stand van Zaken

We hebben gekozen voor een expliciete representatie van de data, zodat je zo min mogelijk gegevens verliest in de opslag. We hebben een *match*-functie bedacht die zowel gebruikers als objecten kan matchen. Het idee hiervan is dat elk *item* (object of gebruiker) gedefinieerd wordt door een aantal eigenschappen. Door het aantal overeenkomende eigenschappen te tellen kan de overeenkomst bepaald worden. Zo kan het systeem in principe ook leren welke (soorten) eigenschappen veel vertellen over de match en welke niet, waarna je de eigenschappen een bepaald gewicht kunt meegeven.

Met dit idee zijn we begonnen aan een implementatie om verschillende mogelijkheden te kunnen testen. We hebben ervoor gekozen om met één objectdomein te beginnen om te testen of de implementatie überhaupt werkt. Hiervoor hebben we de Movielens-dataset gebruikt. We implementeren het geheel in de Google App Engine om gebruik te kunnen maken van de opslag en vooral rekencapaciteit van Google, en omdat het veel rompslomp scheelt in de implementatie, zoals gebruikersbeheer en dergelijke. Het staat meteen online zodat het makkelijk te delen en te testen is.

Vervolg

Het doel is om eind december een werkende *recommender* te hebben, die in elk geval voor één domein goede aanbevelingen kan doen. Dit willen we gaan testen aan de hand van de dataset van Movielens. Hierbij zijn de *mapping*- en *matching*-stappen uiteraard het belangrijkst.
Met een beetje geluk kunnen we er een tweede domein aan toevoegen om te zien of het interessante resultaten oplevert. Het nadeel is dat er geen geschikte datasets beschikbaar zijn over meerdere product-domeinen. Het liefst verschillen ze écht veel van elkaar (bijvoorbeeld stofzuigers).
Een ander punt is dat de huidige implementatie leidt tot aanbevelingen die vooral de 'diepte' ingaan. Dat wil zeggen, een gebruiker zal vooral objecten aanbevolen krijgen die in het verlengde liggen van zijn huidige smaak. Dit kan leiden tot een soort van tunnelvisie. Het zou leuk zijn om impliciet of expliciet ervoor te zorgen dat de gebruiker aanbevelingen krijgt die juist niet in het verlengde liggen van zijn voorkeur, maar verbreden op gebieden waarover nog niet veel bekend is van de gebruiker in het systeem.

Naast deze uitwerking zou het interessant zijn om verschillende *match*- en *mapping*-methoden te vergelijken of misschien te combineren. Het is met name met tags ook interessant om te kijken of je het semantisch kunt oplossen.
Wat tags betreft kan het ook interessant zijn om te leren begrijpen in welke omstandigheden een gebruiker een tag juist *niet* geeft. Vergeet een gebruiker dat of doet hij het bewust niet? In het laatste geval kan het informatief zijn om dat ook expliciet op te slaan.
Aan deze laatste dingen zullen we waarschijnlijk niet toekomen, maar aan de hand van ons werk zou iemand later kunnen proberen om dit uit te zoeken.