Afrondingsproject Movielens 20m met LINQ

# Project omschrijving

## Aanleiding

Het project waar u over gaat lezen is gemaakt om de laatste studiepunten te behalen voor de studie van Casper Zoeteman (0831596). Door middel van een vervangende opdracht (die is besproken en is goedgekeurd door John Grobbe) zijn laatste 3 modules (INFSEN01-1, INFDTA01-1 en INFDEV03-5) kan halen.

## Doel

Het te behalen doel van deze opdracht is het aantonen van de volgende vaardigheden:

* Beheersing van complexere SQL opdrachten
* Beheersing van complexe databases in elkaar zetten
* Beheersing van Lambda functies en function programmeren door middel van het toepassen van LINQ queries.
* Begrijpen en toepassen van Data Sience/machine learning aan de hand van het toepassen van verschillende Data Sience formules

## **Opdracht & Verplichte onderdelen**

De opdracht heeft de volgende vorm gekregen:

*“De opdracht bevat moderne databasetechnieken, datamining technieken, onderzoek en moderne softwaretechnieken. Hierbij is geprobeerd tot een logisch geheel te komen.*

*De opdracht zal met een aantal sprints uitgevoerd worden, waarbij tweewekelijks overleg over de voortgang zal plaatsvinden. Uitgangspunt is de Movielens 20M dataset (*[*https://grouplens.org/datasets/movielens/*](https://grouplens.org/datasets/movielens/)*) Deze set moet na opgeschoond te zijn geïmporteerd worden in een relationele database. Hiervoor zal een genormaliseerd databasemodel ontworpen moeten worden. Uit de database zullen een aantal overzichten moeten komen die op verschillende technieken gebaseerd zijn:*

•    *Basis tabel overzichten (zoeken, filteren, sorteren)*

•    *Statistische overzichten (mogelijkheden die SQL biedt)*

•    *Trends*

•    *Clustering (Nearest Neighbour zal efficiënt geïmplementeerd moeten worden)*

•    *Visualisaties*

*Bij de communicatie met de database moet LINQ gebruikt worden.*

*Er zal onderzoek gedaan moeten worden naar:*

•    *Visualisaties in C#*

•    *Nearest neighbour algorithme*

•    *Evt. technieken ter verbetering van de performance*

*Aan het eind van het project moet de oplossing getoond worden waarbij de verschillende deelaspecten toegelicht worden."*

* Gequote van Meneer Grobben & mevrouw Van IJperen

## Zelf toegevoegde onderdelen

* Efficient en correct de SQL database vullen met de door MovieLens 20m geleverde data
* Stored procedures
* MVC5 met Razor, bootstrap met het DataBase First principe
* MMVC (Model-Model-View-Controller)

# Software omschrijving

De software bestaat twee complete lostaande projecten die gebruik maken van dezelfde Microsoft SQL server database.

Het eerste project is de backend bedoelt om de data efficient in te lezen, weg te schrijven en zodanig aan te vullen met een leeg passend dataveld. Het lege passende dataveld wordt later aangevuld wanneer dit nodig is. De code wordt weggeschreven naar een Microsoft SQL server database.

Het tweede project is de frontend in MVC5 met Razor en LINQ om de database op te halen.

# Structuur

## Model – View – Controller

Het model-view-controller principe is een designpattern waarbij de programmeur verantwoordelijkheden strict scheidt. De models bevatten alle informatie zoals deze ook in de database staat. De Controller haalt de data op en wordt aangeroepen als een API. De data die doorgegeven wordt door de Controller aan de View wordt ingeladen. Op die manier kan de data gericht opgehaald worden en getoont worden in de View. Dit zorgt voor minder laadtijd en minder aanroepen voor de database; dus win je er efficientie mee.

De Models bestaan uit een DatabaseContext aangezien voor dit project een Database First principe is gekozen. Dit is in overleg geweest met meneer Grobben omdat dit een nieuwe structuur is voor Casper Zoeteman en daarom ook leerzaam.

De Models die in de map ‘Model’ staan zijn ViewModels. Dit zijn tussen oplossingen om extra data aan een model meer informatie meegegeven vanuit de bijbehorende Controller.

## Strategy pattern

Het strategy pattern is toegepast om de DataSience/machinelearning algoritmen aan elkaar te knopen. De algoritmen die gebruikt zijn, zijn losse wiskundige formules die elk in hun eigen SOLID Class zitten. Dit zorgt ervoor dat de verantwoordelijkheid zoveel mogelijk singulair gescheiden word ten het onderhouden van deze classes minder tijd vergt.

## Singleton

Het database filler project bestaat uit allemaal losse SOLID singletons. Elke actie voor het toevoegen van een losse dataset is een Singleton op zich. Om dit helemaal netjes af te maken zou van de mainclasse ook een factory gemaakt kunnen worden; zodat elke singleton een-voor-een aangeroepen kan worden in plaats van dat het handmatig gebeurd.

# Software keuzeverantwoording

LINQ is een functioneel gerichte taal om SQL methodisch op de frontend in te laden. Razor is een frontend oplossing van Microsoft om modellen te binden aan een HTML pagina.

# Recommender system implementatie

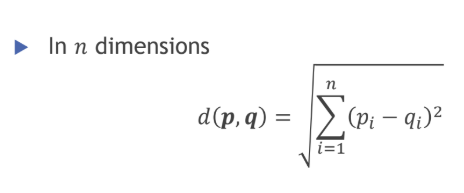
## Item-Item

Met een Item-item voorspellings- en aanbiedingssysteem worden voorspellingen bedoelt om aan de hand van al bekende cijfers gebruikers met elkaar te vergelijken aan de hand van wiskunde. De vergelijkbaarheid wordt uitgedrukt in een cijfer tussen de -1 (helemaal niet) en 1 (helemaal wel). De vergelijking vindt plaats tussen een target user en alle andere users binnen een cluster of de gehele user database.

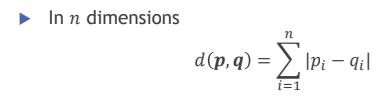
Het ‘item’ of het aantal ‘items’ dat uiteindelijk aangeboden wordt zijn films die de target user nog niet gewaardeerd heeft maar vergelijkbare users met een matchingsscore van 80% of hoger wel hebben gewaardeert. Zo kan het zijn dat iemand die enkel nog Fantasy films heeft gewaardeerd ineens een Lego Movie aangeboden krijgt omdat de target user deze nog niet heeft gezien maar al zijn matchende en vergelijkbare users wel.

Om een zo eerlijke en exact mogelijke benadering van een matchende user te bekenen zijn verschillende wiskundige formules nodig die allemaal een afstand berekenen tussen de users. Hoe korter de afstand, uitgedrukt in vectoren, hoe meer de users op elkaar zullen lijken op statistisch niveau. Voor dit project is gebruik gemaakt van de Euclidian Distance, Manhattan Distance en de Pearson Correlation Coefficient om de vergelijkbaarheid te berekenen.

### Euclidian Distance

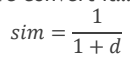


### **Manhattan Distance:**



Zowel de Euclidian Distance en Manthattan Distance berekenen een afstand. Beide formules zijn gemaakt voor gevallen waarbij sprake is van dense data. Dense data is een term die wordt gebruikt om de hoeveelheid ingevulde gebruikers data aan te duiden. Denk bij gebruikersdata bijvoorbeeld aan het geven van cijfer aan een film.

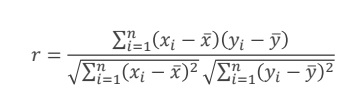
Om van een afstand een gelijkheid te kunnen maken, moet er een kleine conversie plaatsvinden voor de afstand. Hiervoor is een formule opgesteld:



Bij een uitkomst van Sim =1 is er een perfecte match. Een directe match vind bijna nooit plaats aangezien bovenstaande formule een mooi voorbeeld is van een asymptoot. De 1 zal bijna nooit exact geraakt worden maar het kan wel een benadering van 1 zijn.

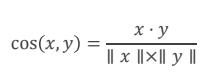
De conclusie is dat de Euclidian Distance en Manthattan Distance beiden efficiente, lichte en makkelijk toe te passen berekeningen voor het berekenen van gelijkheid tussen gebruikers aan de hand van waarderingen op films. Het grote nadeel is dat dit wel een dense data set vergt, wat over het algemeen zeldzaam is.

### Pearson correlation coefficient



De pearson coefficient geeft een correlatie weer tussen de -1 en 1. De correlatie houdt in hoe gelijk gebruiker A is aan B, wanneer de getallen erg verspreid van elkaar liggen. De pearson geeft voor de use case “MovieLens 20m” de beste uitkomst aangezien de meeste gebruikers ver uit elkaar liggen. Dit geeft een eerlijkere uikomst wanneer beide users een heel ander beeordelingsgedrag vertonen.

### Cosine similarity



De Cosine Similarity is het beste om toe te passen waneer er sprake is van Dense Data. Dense Data houdt in dat de data lege vakken bevat. Waarbij de Pearson, Manhattan en Euclidian goed werken bij Dense Data werkt de Cosine juist goed voor Sparse Data.

Aangezien de “MovieLens 20m” erg veel lege vakken bevat is dit de Cosine de beste optie om te kiezen voor ‘t berekenen van de Similarities tussen gebruikers.

## User-Item

User item wordt gebruikt om een waardering voor een user te voorspellen.

## Voorspelling aan de hand van Tags

Afgezien van een film aanraden aan de hand van User Ratings kan een film ook aangeraden worden door middel van de User Tags te vergelijken. Een Tag is een beschrijving, samenvattend woord of korte recensie voor een film gegeven door een specifieke gebruiker.

Om een goede inschatting en voorspelling te maken hoe een gebruiker een door die person nog niet beoordeelde film zal omschrijven. Zal eerst gekeken moeten worden naar de films die dan al wel door hem beoordeelt zijn in text. Deze woorden worden dan met vergelijkbare users (sim >= 0,75) vergeleken. Het vergelijken wordt gedaan aan de hand van alle woorden per film per unieke gebruiker in een .TXT document te zetten. Elke regel bevat een FilmID en daarbij de tag. De title van ‘t .txt bestand is de userID (dit kan natuurlijk ook allemaal in SQL zelf door middel van verschillende views, maar voor het gemak is het voorbeeld uitgelegd in losse bestanden).

Wanneer van alle vergelijkbare users een dergelijk bestand is opgemaakt kunnen overeenkomende woorden opgesteld worden. Wanneer er in zo’n geval veel woorden als “Fantasy, Trolls, Wizards, Knights” voorkomen en de gebruiker dan enkel nog maar Camelot (1982) en Lord of the Rings (Fellowship of the ring) heeft gezien zal hij waarschijnlijk Harry Potter and the chamber of secrets ook leuk vinden. Aangezien deze film binnen deze beschrijving valt en daarbij een grote match heeft (sim>82).

Deze manier van voorspellen is enkel in theorie opgenomen in dit verslag om aan te tonen dat er meer manieren mogelijk zijn om met de MovieLens 20m voorspellingen te doen. Aangezien dit buiten de scope viel is dit buitenwege gelaten maar wel onderzocht. De inschatting in uren uitgedrukt om dit te implementeren is 40 uur; vanwege de hoge complixiteit en het aantal uur dat nodig zal zijn om alles te refactoren.

# Custom dataloader voor SQL Server

**De MovieLens 20M database staat letterlijk voor 20Miljoen records aan user informatie voor films**

## **Grote datadumps**

## Data chunken

## Efficient data omzetten en verschrijven in een Relationele SQL database