



Faculteit Bedrijf en Organisatie

Personalized Search: Graph vs. Elasticsearch

Shauni Van de Velde

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Guy Dekoning
Co-promotor:
Nicolas Lierman

Instelling: MultiMinds

Academiejaar: 2019-2020

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Personalized Search: Graph vs. Elasticsearch

Shauni Van de Velde

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Guy Dekoning
Co-promotor:
Nicolas Lierman

Instelling: MultiMinds

Academiejaar: 2019-2020

Tweede examenperiode

Woord vooraf

Deze bachelorproef met als titel "Personalized Search: Neo4j vs Elasticsearch" werd geschreven in het kader van het behalen van een diploma als bachelor in de Toegepaste Informatica aan de Hogeschool Gent.

Dit onderwerp werd mij voorgesteld door MultiMinds, en sprak mij aan omdat dit ondanks de recente wetgeving rond GDPR nog steeds een actueel onderwerp is. Ik had reeds een idee in mijn hoofd van hoe personalisatie in e-commerce toepassingen in zijn werk ging, en ik wou mij hier met plezier in verdiepen. Dit onderzoek was een leerzame ervaring voor mij, en als ik erop terugblik zou ik geen ander onderwerp gekozen hebben.

Bij deze zou ik graag nog even enkele personen bedanken die mij geholpen en gesteund hebben bij het maken van deze bachelorproef, zonder hun zou dit onderzoek niet tot stand kunnen gekomen zijn.

Als eerste zou ik graag mijn copromotor, Nicolas Lierman van MultiMinds, bedanken voor het aanbieden van dit interessante onderwerp, alsook de technische ondersteuning en het nazien van dit werk.

Ten tweede zou ik ook graag mijn promotor, Guy Dekoning van Hogeschool Gent, bedanken voor de vlotte en zeer duidelijke feedback, en het blijven steun tonen in mijn onderzoek.

Ten slotte wil ik ook graag mijn ouders bedanken, om mij in alle rust te laten werken ondanks dat ik hele dagen thuiszit; mijn medestudenten, die mijn rubberen eendje wouden spelen; en mijn vrienden, in het bijzonder Angelo Carly, om mij morele steun te bieden en inhoudelijke feedback te geven van het perspectief van een buitenstaander.

Samenvatting

In samenwerking met MultiMinds werd besloten onderzoek te voeren naar de haalbaarheid van het gebruik van twee vooraf bepaalde opties voor het implementeren van een gepersonaliseerde zoekfunctie bij e-commerce toepassingen, de opties die gekozen werden door de opdrachtgever zijn Neo4j en Elasticsearch.

De reden voor dit onderzoek is dat zo goed als alle gekende webshops gebruik maken van een zoekfunctie, maar dat deze vaak beperkt blijft tot de productcatalogus. In dit onderzoek wordt dus nagegaan of het mogelijk is deze zoekfunctie uit te breiden met data van gebruikers om zo een meer gepersonaliseerd resultaat te kunnen opleveren.

Enkele voorbeelden van dergelijke gebruikersdata zijn geslacht, leeftijd, locatie, etc., maar ook bijvoorbeeld familiale verbanden, zo is het bijvoorbeeld wenselijk dat in het geval van twee samenwonenden Persoon A en persoon B, er merken worden aanbevolen aan Persoon A waarvan Persoon B regelmatig producten koopt.

In eerste instantie werd er een literatuurstudie gedaan naar de reeds bestaande vormen van personalisatie, al dan niet binnen e-commerce toepassingen als marketingvorm.

Ten tweede werden de verschillende types van aanbevelingssystemen onderzocht. Daarbij werden twee belangrijke families gevonden: Collaborative Filtering-algoritmen, en Content Based-algoritmen. Voor de implementatie van dergelijke algoritmen zijn ook hybride vormen mogelijk, deze hybride vormen werden gebruikt in dit onderzoek.

Ten derde werd er ook een literatuuronderzoek gedaan naar de twee platformen, Neo4j en Elasticsearch. In dit deel van het onderzoek werd bekeken of deze platformen de mogelijkheden hadden om een aanbevelingssysteem te implementeren dat voldoende accurate resultaten zou opleveren. Uit dit onderzoek bleek dat Neo4j sterk uitblinkt in

het gebruiken van data over de relaties tussen twee personen, waar Elasticsearch eerder uitblinkt in het zeer snel opleveren van resultaten die aan bepaalde criteria voldoen.

Als vierde puntje in het onderzoek werd ook verder ingegaan op een nadeel dat het personaliseren van e-commerce toepassingen met zich mee kan brengen, namelijk een 'Filter Bubble', waarbij gebruikers enkel nog resultaten te zien krijgen die volledig passen bij hun profiel. Bijgevolg zullen gebruikers weinig of zelfs geen resultaten mogen verwachten die door hun nog niet gezien werden.

Als laatste deel van de literatuurstudie werd bekeken of de implementatie van een systeem dat op deze manier gebruik maakt van persoonlijke data en gegevens wel als legaal aanschouwd kan worden onder de wetgeving van de GDPR. Onderzoek wees uit dat het systeem hier geen problemen van zou mogen ondervinden.

De resultaten van dit onderzoek wezen uit dat beide platformen hun eigen voor- en nadelen met zich meebrachten, verwijzend naar de verschillende functionaliteiten die onze zoekfunctie vereist. Een belangrijke factor bij het maken van de keuze is ook de eenvoud van de implementatie, waarbij al snel bleek dat het zeer moeilijk zal zijn om de sterktes van Neo4j te repliceren in Elasticsearch, of omgekeerd.

De resultaten zetten aan tot verder onderzoek naar de mogelijkheid om deze twee systemen parallel te gebruiken, en de resultaten ervan te combineren. Zo kan Neo4j gebruikt worden als een 'Knowledge Graph', die dient als aanvulling voor de data die Elasticsearch tot zijn beschikking heeft.

Inhoudsopgave

1	Inleiding	17
1.1	Probleemstelling	17
1.2	Onderzoeksvraag	18
1.3	Onderzoeksdoelstelling	18
1.4	Opzet van deze bachelorproef	19
2	Stand van zaken	21
2.1	Personalisatie	21
2.1.1	Personalisatie op basis van e-mail en sociale media	21
2.1.2	Personalisatie op basis van geografische locatie	22
2.1.3	Personalisatie op basis van IP-adres	22
2.1.4	Verwante inhoud personalisatie	23

2.2	Recommender Systems	24
2.2.1	Haalbaarheid	24
2.2.2	Algoritmen voor aanbevelingssystemen	25
2.3	Wat is Graph?	27
2.3.1	Graph	27
2.3.2	Knowledge Graphs	29
2.3.3	Neo4j	31
2.4	Wat is ElasticSearch?	33
2.5	Filter Bubble	38
2.6	GDPR	39
2.6.1	Gegevensverwerking	39
2.6.2	Rechten van de gebruikers	40
2.6.3	Effect van de GDPR op e-commerce	41
3	Methodologie	43
3.1	Uitwerking ElasticSearch	43
3.1.1	Producten	43
3.1.2	Gebruikers	44
3.1.3	Graph API Plug-in	46
3.1.4	Conclusie ElasticSearch	46
3.2	Uitwerking Neo4j	47
3.2.1	Model	47
3.2.2	Ophalen van data	49
3.2.3	Search	50
3.2.4	PageRank	51

3.2.5	Personalized PageRank	51
3.2.6	Community Detection	53
4	Conclusie	55
4.1	GDPR	55
4.2	ElasticSearch	56
4.3	Neo4j	56
4.4	Resultaat	56
A	Onderzoeksvoorstel	59
A.1	Introductie	59
A.2	State-of-the-art	59
A.3	Methodologie	60
A.4	Verwachte resultaten	61
A.5	Verwachte conclusies	61
	Bibliografie	63

Lijst van figuren

3.1	Overzicht van de beschikbaarheid van de Graph plugin voor de Elastic licenties	46
3.2	Overzicht van de ingevoerde data in een Neo4j graaf	49
3.3	Resultaat PageRank algoritme	52
3.4	Resultaat Personalized PageRank algoritme	52
3.5	Resultaat Louvain algoritme	53

Lijst van tabellen

2.1 Source: https://neo4j.com/news/how-much-faster-is-a-graph-database-really/	33
---	----

Listings

2.1	Tekst-gebaseerd zoeken: Query om een simpele zoekopdracht uit te voeren met de term 'deodorant'	33
2.2	Tekst-gebaseerd zoeken: Query om een simpele zoekopdracht uit te voeren waar de categorie van het product overeen komt met 'Audio'	33
2.3	Tekst-gebaseerd zoeken: Query die de resultaten sorteert op basis van hoe vaak het woord 'accepted' voorkomt in het veld 'message' van een document	34
2.4	Bereik zoeken: Een query om alle documenten op te halen waarbij het veld 'price' tussen 30 en 50 ligt	34
2.5	Combineren: Voorbeeld van een 'bool' query	35
2.6	Scoring: voorbeeld van een function_score query	36
2.7	Scoring: voorbeeld van een script_score query	37
2.8	Decay: voorbeeld van een decay functie	38
3.1	Query om één enkel product aan te maken	43
3.2	Query om één enkel product op te halen	44
3.3	Query om één enkele gebruiker aan te maken	44
3.4	Query om een zoekopdracht met term 'Deodorant' uit te voeren, met filters en een score op basis van informatie van een gebruiker	44

3.5	Neo4j query voor het aanmaken van producten en klanten	47
3.6	Neo4j query voor het aanmaken van een relatie tussen een product en een klant	48
3.7	Neo4j query voor het aanmaken van een relatie tussen een twee klanten. . . .	48
3.8	Neo4j query die alle likes en aankopen van de familie van een bepaalde gebruiker c1 weergeeft	49
3.9	Neo4j query die de aankopen van gebruikers weergeeft waar een gebruiker c1 gemeenschappelijke aankopen mee heeft	50
3.10	Neo4j query die de aankopen en likes van gebruikers weergeeft waar een gebruiker c1 gemeenschappelijke aankopen mee heeft	50
3.11	Een index creëren binnen Neo4j om zoekopdrachten op basis van tekst uit te voeren	50
3.12	Een zoekopdracht uitvoeren op basis van tekst	50
3.13	Een genoemde graaf creëren om graafalgoritmen op uit te voeren	51
3.14	PageRank algoritme uitvoeren	51
3.15	Personalized PageRank algoritme	52
3.16	Personalized PageRank algoritme	53

1. Inleiding

1.1 Probleemstelling

Elke webshop of e-commerce toepassing die vandaag de dag gekend is maakt gebruik van een zoekfunctie die gebruikers toelaat om snel en eenvoudig producten te kunnen opzoeken. Deze zoekfunctie is beperkt tot de beschikbare informatie, en bijgevolg dus in de meeste gevallen gelimiteerd tot de productcatalogus.

Rekeninghoudend met de algemene trend rond personalisering van de *customer experience* zou het ook aangewezen zijn om ook de zoekfunctionaliteit op e-commerce websites te personaliseren. Dit zou resulteren in een betere customer experience voor de klant en een hogere conversie voor het bedrijf.

Het personaliseren wordt in deze context bedoeld als het gebruiken van persoonlijke data om de resultaten van een zoekopdracht te verfijnen en zo producten te kunnen aanbieden die passen bij de voorkeuren en het gedrag van de gebruiker.

Hierop biedt dit onderzoek een mogelijke oplossing om dergelijke functionaliteiten te bereiken. Er zullen twee vooraf bepaalde platformen gebruikt worden om dit uit te werken, namelijk Neo4j en Elasticsearch. Het onderzoek zal uitwijzen welk platform het meest geschikt is om een gepersonaliseerde zoekfunctie mee te implementeren.

Belangrijk hierbij is om na te gaan of de wetgeving omtrent GDPR het gebruik van de persoonlijke data als legaal aanhoudt in de context van zoekfuncties, meer specifiek op welke manier deze data gebruikt zal worden binnen de uitwerking van de twee voorgenoemde platformen.

1.2 Onderzoeksvraag

De onderzoeksvraag bestaat eruit om te ontdekken welk van de twee vooraf gekozen platformen of welke technologie de beste oplossing biedt om in real-time op grote schaal gepersonaliseerde zoekresultaten te kunnen leveren.

Belangrijke criteria hierbij zijn snelheid, performantie, kwaliteit van de resultaten, en of het al dan niet mogelijk is om familierelaties te kunnen verwerken. Met performantie wordt hier geduid op de rekenkracht die een zoekoperatie zal nodig hebben, dit is gerelateerd aan de kost.

Nog een niet te missen detail is de invloed van de wetgeving rond GDPR, en of een gepersonaliseerde zoekfunctie op basis van persoonlijke data al dan niet als legaal wordt aanzien onder deze wetgeving.

Concreet omvat dit onderzoek volgende vier onderzoeksvragen:

- Is het implementeren van een gepersonaliseerde zoekfunctie op basis van persoonlijke data legaal onder de wetgeving rond GDPR?
- Welke technologie biedt de mogelijkheid om een gepersonaliseerde zoekfunctie te implementeren?
- Welke technologie biedt de beste resultaten op basis van snelheid, performantie en kwaliteit?
- Laten deze technologieën toe om rekening te houden met factoren die niet te maken hebben met historisch koopgedrag (bv. leeftijd, geslacht, gezinssamenstelling)?

Deze vragen zullen doorheen dit onderzoek een antwoord krijgen.

1.3 Onderzoeksdoelstelling

Het onderzoek heeft als doel om te ontdekken in hoeverre de persoonlijke data van een specifieke gebruiker en contextuele data kan ingeschakeld en gecombineerd worden met de data uit een productcatalogus om persoonlijke en relevante zoekresultaten te genereren.

Dit zal verwezenlijkt worden door middel een prototype op te stellen van beide vooraf gekozen platformen, zijnde Neo4j en Elasticsearch, en de mogelijkheden ervan te ontdekken. De ondervindingen van het onderzoek, naast de informatie die een literatuurstudie biedt, zullen een antwoord geven op de onderzoeksvragen.

Bij het gebruik van contextuele data wordt hier geduid op vooral historische koopdata van de persoon zelf, maar ook demografische data zoals geslacht, leeftijd, gezinssamenstelling kunnen hierbij belangrijke factoren zijn.

Het is niet enkel de mogelijkheid om dergelijk zoekstelsel te verwezenlijken die van belang is voor dit onderzoek, ook het gebruiksgemak en eenvoud van installatie speelt hierbij een rol, en zullen ondanks een eerder subjectieve voorwaarde te zijn toch effect

hebben op de conclusie.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 4, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2. Stand van zaken

2.1 Personalisatie

Personalisatie van webapplicaties en websites draait erom de bezoekers een op maat gemaakte ervaring aan te bieden. Dit kan op verschillende manieren toegepast worden, en kan meerdere doelen hebben.

In dit hoofdstuk wordt er verder ingegaan op welke manieren personalisatie van websites wordt toegepast, en wat dit betekent voor zowel de bezoeker als het bedrijf zelf.

2.1.1 Personalisatie op basis van e-mail en sociale media

Zowat iedereen heeft wel te kampen met een overvloed aan e-mails in hun postvak van allerlei websites waar ze hun e-mailadres ooit hebben vrijgegeven. Het is een courante denkwijze om er van uit te gaan dat deze e-mails door de meeste mensen simpelweg verwijderd worden, maar e-mailmarketing blijft een van de meest succesvolle marketingstrategieën (Dehkordi, Rezvani, Rahman, Nahid & Jouya, 2012).

E-mailmarketing is relatief makkelijk te implementeren en vereist weinig technische investering, meestal wordt dit verwezenlijkt via systemen van derden zoals bijvoorbeeld MailChimp.

Een nadeel van deze marketingvorm is dat het bedrijf continu bezig moet zijn met nieuwe inhoud te creëren voor deze e-mails, ook moet de website continu geüpdatet worden met relevante informatie en pagina's die relevant zijn voor de e-mails die ze versturen.

2.1.2 Personalisatie op basis van geografische locatie

Geografische personalisatie is het aanpassen van de website op basis van de locatie van de gebruiker. Gebruikers uit België die naar de website van een internationaal bedrijf surfen, zullen dan worden omgeleid naar een Nederlandse of Franse versie van die website.

Geografische personalisatie kan ook gebruikt worden om de inhoud van een pagina aan te passen aan de hand van de locatie van de gebruiker, of om vertalingen aan te bieden.

Een nadeel hiervan is dat mensen die op reis gaan het soms moeilijk zouden kunnen hebben om naar de juiste versie van de website te navigeren, aangezien het systeem de gebruiker zal willen omleiden naar de pagina of inhoud die voorzien is voor het land waar zij zich momenteel in bevinden. Eenzelfde probleem kan zich voordoen bij bedrijven die hun webverkeer omleiden via een ander land door middel van bijvoorbeeld een VPN.

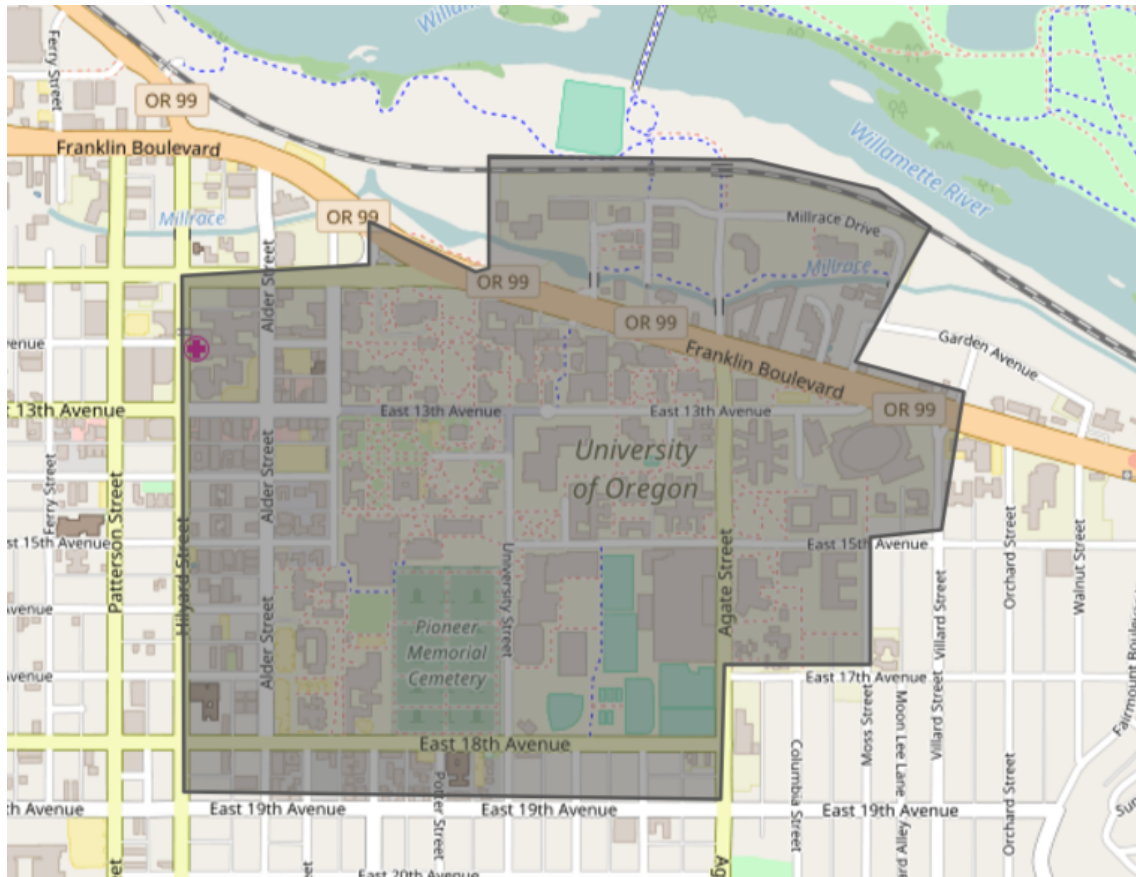
Een mogelijke oplossing voor dit probleem is het bijhouden van cookies. Als de geografische informatie over een gebruiker daarin wordt opgeslagen, kan deze via de cookie opgehaald worden en zal een gebruiker alsnog de website met de correcte inhoud en taal aangeboden krijgen. Als de gebruiker geen cookies toelaat, of zijn cookies verwijdert, wordt deze mogelijke oplossing tenietgedaan.

Geografische personalisatie is ook relatief eenvoudig te implementeren en kan een grote troef zijn op de internationale markt.

2.1.3 Personalisatie op basis van IP-adres

Deze methode van personalisatie is wat minder opvallend, aangezien het bij de gemiddelde internetgebruiker weinig tot nooit zal voorkomen, dit omdat zij het internet gebruiken via een serviceprovider zoals Telenet of Proximus.

Deze vorm van personalisatie wordt gebruikt om zakelijke gebruikers en bedrijven te kunnen identificeren op basis van hun IP-adres. Zo kan men zien of een bezoeker bij een bepaald bedrijf werkzaam is om deze direct aan te spreken op bijvoorbeeld de homepage.



Net zoals bij personalisatie op basis van locatie kan dit misleidende resultaten opleveren, bijvoorbeeld als de werknemer van thuis werkt of het IP-adres niet duidelijk aantoont vanuit welk bedrijf het webverkeer van de bezoeker afkomstig is. Ook voor prestatie kan dit negatieve gevolgen hebben, aangezien deze vorm van personalisatie afhankelijk is van systemen van derden.

Verder moet er ook inhoud gecreëerd worden voor elk bedrijf dat men specifiek wil aanspreken. Dit is een tijdrovend proces, maar aangezien deze vorm van personalisatie weinig voorkomt, is het wel een troef waardoor het bedrijf zich kan onderscheiden van de meerderheid en zich kan laten opvallen.

2.1.4 Verwante inhoud personalisatie

Dit is de vorm van personalisatie die een grote meerwaarde zal leveren aan dit onderzoek. De meeste mensen hebben deze vorm al ondervonden op een webshop zoals Amazon of Bol.com. Deze vorm van personalisatie draait erom de gebruikers artikelen of producten aan te raden op basis van items die ze al eerder bekeken hebben, alsook het gedrag van andere gebruikers.

De werking van het aanbevelingssysteem van Amazon is gebaseerd op enkele complexe algoritmen. (Linden, Smith & York, 2003) Dit is natuurlijk verantwoord omdat zij een gigant zijn in de e-commerce industrie. Amazon werkt op zeer grote schaal werken en

heeft veel geld geïnvesteerd in de ontwikkeling van hun systeem.

In de realiteit hoeven de technologieën voor aanbevelingen van producten niet zo complex te zijn voor gewone webshops en bedrijven, vaak is het voldoende om relaties te creëren tussen artikels en op basis van deze relaties nieuwe artikels aan te raden aan de gebruikers.

Een voorbeeld van een relatie tussen twee artikels is de welbekende 'Anderen bekeken ook' blok die vaak zichtbaar is bij het bekijken van een detailpagina van een product.

Een simpelere methode van dergelijke relaties is het aanbieden van verwante producten op basis van categorieën of tags. Tags zijn een manier om kenmerken van een product weer te geven die specifieker zijn dan een categorie. Een categorie kan dan 'schoenen' zijn, terwijl een tag 'lage sneakers' is.

Dit is een klassiek voorbeeld van de probleemstelling van dit onderzoek. De zoekfunctie is dus beperkt tot de productcatalogus en de informatie die beschikbaar is over deze producten. De website maakt dus geen gebruik van persoonlijke informatie om deze zoekresultaten te personaliseren.

Over de mogelijkheden van het personaliseren van zoekresultaten gaan we dieper in in het volgende hoofdstuk van deze literatuurstudie.

2.2 Recommender Systems

Recommender Systems (Resnick & Varian, 1997) zijn aanbevelingen vanuit het systeem die rekening houden met de beschikbare informatie van gebruikers en hun voorkeuren om zo een filter te plaatsen op de informatie die weergegeven wordt. Verder zullen we deze benoemen aan de hand van hun Nederlandse naam 'aanbevelingssystemen'.

Aanbevelingssystemen worden vooral gebruikt in een e-commerce toepassingen waar een zeer groot en verscheiden aanbod aan producten is, en het al vaak lastig wordt om precieze aanbevelingen aan de klant te geven. Hierbij wordt allerlei informatie van een gebruiker verzameld, zoals historische aankopen, items op het verlanglijstje, items waar de gebruiker op geklikt heeft, etc.

In dit onderdeel zullen we kort wat dieper ingaan op de werking van dergelijke aanbevelingssystemen en de achterliggende algoritmen.

2.2.1 Haalbaarheid

Een degelijk systeem biedt een grote meerwaarde voor een bedrijf, zo heeft Netflix een competitie gehouden die 1 miljoen dollar bood aan degene die een aanbevelingssysteem kon maken dat 10% beter presteerde dan hun bestaande systeem.

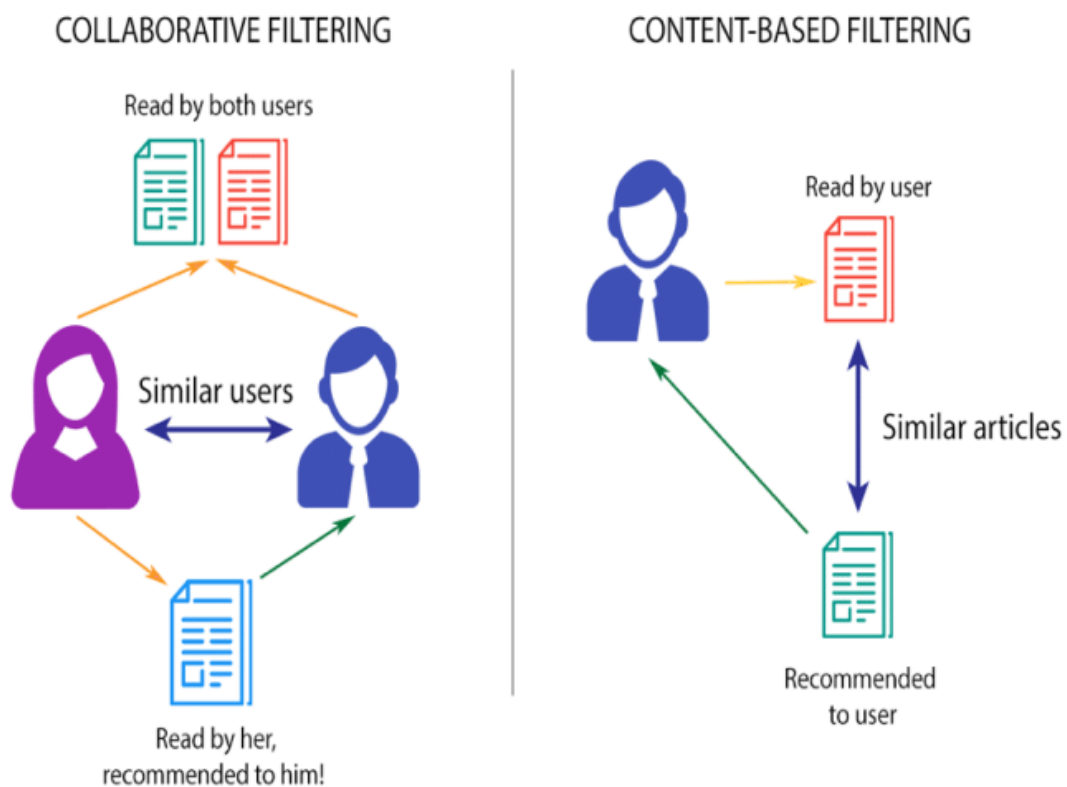
Deze wedstrijd liep van oktober 2006 tot minstens oktober 2011. De wedstrijd werd de

Netflix, 2009 genoemd, Netflix stelde hiervoor een dataset beschikbaar. Enkele groepen hebben het doel behaald, maar het algoritme van de winnende groep is uiteindelijk nooit in productie gebracht, omdat de mogelijke opbrengst niet kon opwegen tegen de extra gevraagde rekenkracht.

Een aanbevelingssysteem moet dus niet enkel de juiste waarden kunnen aangeven, het moet ook realistisch en haalbaar zijn qua rekenkracht en extra bijkomende kosten voor onderhoud. Het ontwerpen van dergelijk systeem is dus geen eenvoudige klus.

2.2.2 Algoritmen voor aanbevelingssystemen

We spreken over twee grote categorieën in algoritmen van aanbevelingssystemen: 'collaborative filtering'-algoritmen en 'content based'-algoritmen. (Adomavicius & Tuzhilin, 2005) Door de evolutie en groeiende ontwikkeling van beter presterende systemen zijn er ondertussen ook enkele algoritmen die niet echt binnen een van deze twee koepels vallen. In de onderstaande figuur wordt een simpele representatie gegeven van de werking van deze soorten algoritmes.



Collaborative Filtering

Het kernidee bij Collaborative Filtering (Schafer, Frankowski, Herlocker & Sen, g.d.) is om aanbevelingen te maken gebaseerd op de voorkeuren van een andere gebruiker met een gelijkaardig gedrag. Zoals de figuur hierboven aantoont dat als gebruiker 1 en 2 hetzelfde

artikel gelezen hebben, gebruiker 2 een artikel als aanbeveling zal krijgen dat gelezen werd door gebruiker 1. Hetzelfde idee kan toegepast worden op allerlei interacties van de gebruikers met een website zoals likes, shares, verlanglijstjes, etc.

In de praktijk geeft deze techniek zeer goede resultaten, maar zoals verwacht brengt deze techniek ook enkele problemen met zich mee. Het meest merkwaardige probleem is een zogenaamde cold start, dit komt onder andere voor bij de eerste interactie van een nieuwe gebruiker met een applicatie die aanbevelingen biedt. Het algoritme heeft dan onvoldoende informatie om een correcte en nuttige aanbeveling op te leveren aan de gebruiker. Een andere oorzaak kan zijn dat er een nieuw product wordt toegevoegd aan het systeem, er kan dan nog niet geweten zijn welk type gebruiker hierin geïnteresseerd zou kunnen zijn.

Een andere factor voor het succes van een Collaborative Filtering algoritme is het aantal gebruikers van een systeem, met andere woorden, hoe meer gebruikers er zijn, hoe correcter de aanbevelingen aan een specifieke gebruiker zal zijn. (Sarwar, Karypis, Konstan & Riedl, 2001). Een gebruiker met ongewone interesses zal logischerwijs in dergelijk klein systeem weinig gelijkaardige gebruikers hebben, en zal dus ook geen optimale aanbevelingen krijgen.

Een groot voordeel van Collaborative Filtering is dat er absoluut geen kennis hoeft te zijn van de toepassing van het systeem, de aanbevelingen worden gegenereerd op basis van het gedrag van de gebruikers en zijn interesses. De producten of hun attributen moeten dus niet gekend zijn om aanbevelingen te kunnen geven, dat maakt het eenvoudiger om een aanbevelingssysteem met de techniek van Collaborative Filtering te implementeren.

Content Based

Content Based aanbevelingssystemen (Lops, de Gemmis & Semeraro, 2011) maken, in tegenstelling tot Collaborative Filtering, wel gebruik van de specifieke producten binnen het systeem en hun attributen. Op basis van deze attributen en de interesse van de gebruiker daarin, wordt per gebruiker een profiel opgezet, elk attribuut krijgt dan een score toegekend, een hogere score betekent grotere interesse. Een attribuut van een product kan dan bijvoorbeeld 'schoenen' of 'PS4 games' zijn, of zelfs een filmgenre.

Een probleem van Content Based is, net zoals bij Collaborative Filtering, het cold start probleem. Als een nieuwe gebruiker het systeem gebruikt, is er voor deze gebruiker nog geen profiel opgesteld en kunnen er ook geen aanbevelingen gemaakt worden.

Een ander probleem van Content Based wordt overspecialisatie genoemd, dit treedt op wanneer het systeem eigenlijk té accurate aanbevelingen doet. Het gevolg hiervan is dat slechts enkele producten voldoen aan de verwachtingen van het systeem, waardoor er geen nieuwe aanbevelingen aan de gebruiker naar voor gebracht worden, en de gebruiker enkel producten zal zien die hij reeds bekeken heeft.

Het andere probleem dat optreedt bij Collaborative Filtering, namelijk dat bij het toevoegen van nieuwe producten niet geweten kan zijn welke gebruikers hierin geïnteresseerd zouden zijn, is niet van toepassing bij Content Based. Het systeem maakt gebruik van de attributen

van producten, dus nieuwe producten kunnen meteen belanden in de aanbevelingen van gebruikers die reeds interesse getoond hebben in andere producten met die attributen. Ook het aantal gebruikers binnen een systeem vormt om dezelfde reden geen probleem bij Content Based aanbevelingssystemen.

Een grote boosdoener bij Content Based kan zijn dat producten slecht gelabeld zijn, en hun attributen onvoldoende passen bij wat het product effectief is. Hierdoor kan het systeem deze producten niet goed vergelijken met andere. Dit is vooral een probleem wanneer de attributen van de producten van verschillende bronnen afkomstig zijn, of manueel slecht opgesteld zijn.

Hybrides

Beide van de voorgenoemde technieken hebben elk hun eigen voor- en nadelen, alsook sterke en zwakke punten. Content Based heeft te kampen met overspecialisatie, maar is wel in staat om nieuwe producten meteen aan te bevelen aan de gebruikers. Collaborative Filtering heeft moeite met het aanbevelen van nieuwe producten en een cold start, maar heeft geen problemen in de aard van overspecialisatie.

De logische redenering is dan natuurlijk om deze twee soorten systemen te gaan combineren, kwestie van het beste van twee werelden te proberen bekomen. Dit worden hybride aanbevelingssystemen (Çano, 2017) genoemd.

De manier van het opbouwen van een hybride systeem kan zijn dat beide methoden afzonderlijk worden uitgevoerd, en hun resultaten op het einde samengebundeld worden. Dit is de meest eenvoudige implementatie.

Een andere manier van combineren kan zijn door de informatie binnen een Collaborative Filtering systeem aan te vullen met informatie uit de gebruikersprofielen. Hierdoor wordt de gelijkaardigheid van twee producten bepaald door zowel de inhoud en welke soorten typische gebruikers deze producten bekijken, kopen, leuk vinden, etc. De informatie van deze gebruikers kan dan bijvoorbeeld leeftijdsgroep, woonplaats, gezinssamenstelling, etc. voorstellen.

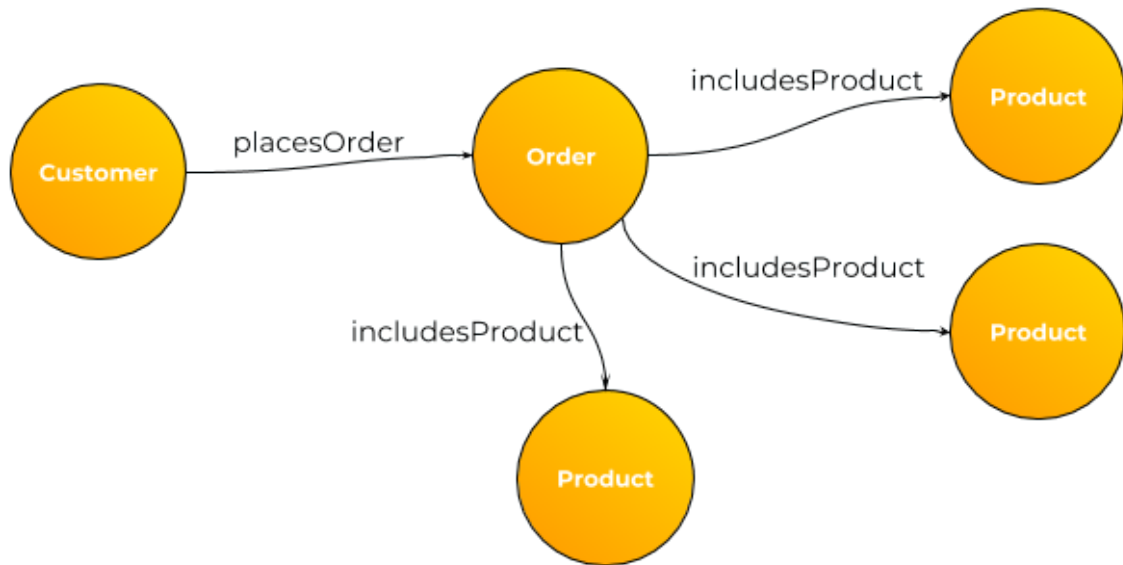
2.3 Wat is Graph?

In de context van deze bachelorproef zal er met 'Graph' steeds verwezen worden naar een Graph databank. In dit onderdeel zullen we wat dieper ingaan op wat dit soort databank precies inhoudt om een volledig begrip van de precieze werking te verzekeren.

2.3.1 Graph

Graph is de databankstructuur die verder in dit onderzoek gebruikt zal worden. Deze structuur maakt gebruik van een wiskundige graaf om data op te slaan. Een graaf bestaat

uit een aantal knopen (genaamd nodes) die al dan niet verbonden zijn.



Relaties hebben de prioriteit bij een graph databank, dit betekent dat relaties tussen knopen gepersisteerd worden in de databank, en niet enkel tijdelijk berekend voor een enkele query, dit is de reden waarom er geen complexe JOIN query's of foreign keys nodig zijn.

Graph databanken hebben dus een voordeel voor functionaliteiten zoals netwerken, aanbevelingen, etc. maar ook fraudedetectie.

Voordelen

Een groot voordeel van de graafstructuur die deze databanken gebruiken is dat deze sneller zijn dan relationele databanken omdat ze op zeer korte tijd naar een bepaalde node kunnen verwijzen, waarbij we bij een relationele databank een JOIN zouden moeten gebruiken.

Graph databanken zijn ook zeer flexibel in gebruik. Bij het veranderen van een model kan er steeds verder gebouwd worden op het bestaande model, zonder dat hierdoor reeds verwezenlijkte functionaliteiten verloren zou kunnen gaan. Dit is ook een voordeel bij het opstarten, er moet niet op voorhand grondig gediscussieerd worden over hoe het model er uiteindelijk uit zal moeten zien.

Deze troeven zijn niet te vinden bij traditionele databankstructuren die met tabellen werken, daarbij moet er alvorens het opzetten van de databank al een zekerheid zijn van hoe het model eruit zal zien, want het model zal de mogelijke functionaliteiten bepalen. Als er dan achteraf nog veranderingen nodig zijn, zal heel het model herzien moeten worden.

Nog een belangrijk voordeel van graph databanken is dat hiermee een *Knowledge Graph* kan opgebouwd worden, hier wordt in een verder hoofdstuk van de literatuurstudie dieper op ingegaan.

Nadelen

Graph databanken hebben niet enkel voordelen, deze zijn vooral afhankelijk van de verwachtingen van de gebruikers en wat zij nodig hebben om hun doeleinden te bereiken. Graph databanken bestaan niet om relationele databanken te vervangen, zij bieden slechts een oplossing voor de beperkingen waar relationele databanken mee kampen.

Graph databanken zijn geen goede keuze wanneer er data moet opgeslagen worden waarbij geen verbanden of connecties zijn tussen deze data, hier blinkt een relationele databank dan weer in uit.

Hetzelfde geldt voor wanneer de databank slechts als opslag gebruikt zal worden, of er slechts simpele query's nodig zullen zijn die in een relationele databank zonder JOIN statement kunnen gebeuren. Dit is niet zozeer een nadeel, maar benadrukt wel dat graph databanken geen universele *one size fits all* oplossing zijn. Graph kan dit, maar is hiervoor niet geoptimaliseerd.

Een voordeel dat eigenlijk als nadeel kan aanzien worden is dat de data niet consistent hoeft te zijn. Het model dat opgebouwd wordt is niet hetzelfde als een schema bij relationele databanken. Even een kort voorbeeld:

Bij relationele databanken zijn deze objecten gebonden aan een schema van tabellen waar de relaties vooraf gedefinieerd zijn. Hierin wordt dus afgedwongen dat een Persoon werkzaam kan zijn bij een bedrijf, en een persoon een huisdier kan hebben. Het is dus niet mogelijk om een huisdier in te voeren als werkzaam zijnde bij een bedrijf. Een graph databank zal het toestaan om een relatie van het huisdier als 'werknemer' te leggen naar het bedrijf, aangezien dit soort databank niet gebonden is aan een schema. Dit is natuurlijk niet wenselijk.

De voor- en nadelen van graph databanken worden nog even kort opgesomd in volgende tabel:

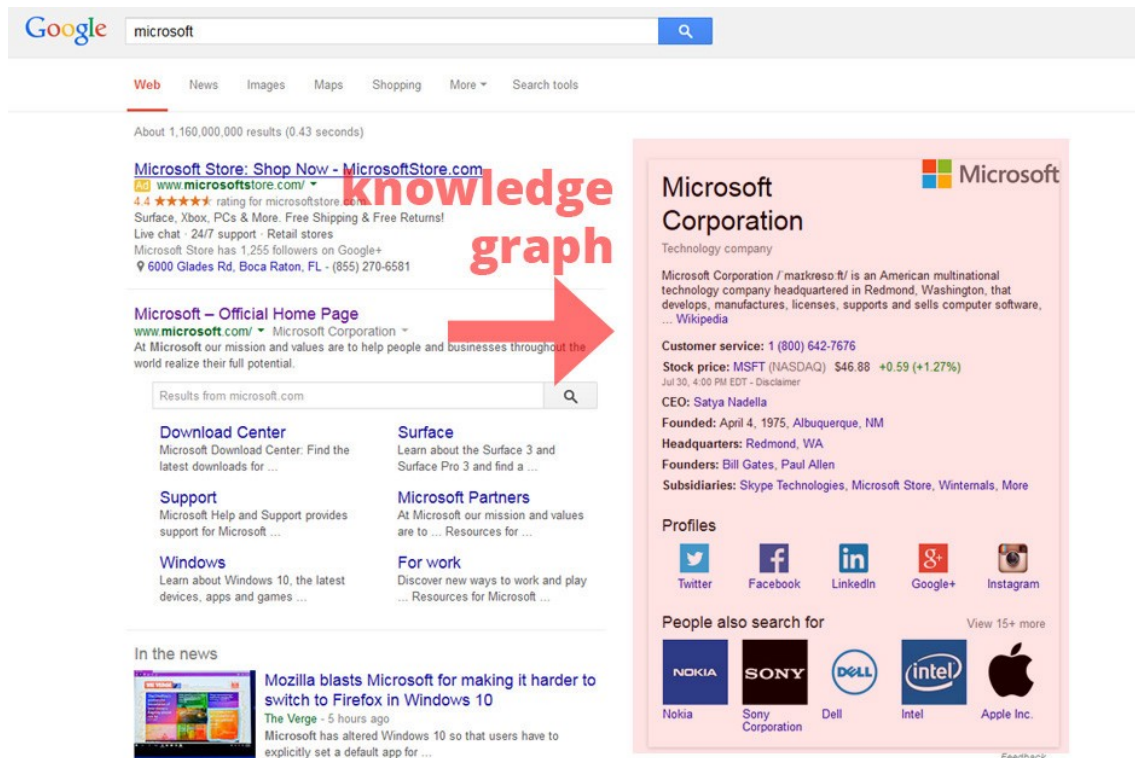
Voordelen	Nadelen
<ul style="list-style-type: none">- Sneller dan relationele databanken bij het queryen met meerdere tabellen- Flexibel- Makkelijk te begrijpen structuur- Opbouwen van Knowledge Graph	<ul style="list-style-type: none">- Gaat slecht om met data waar weinig of geen connecties zijn- Data is niet gebonden aan een schema- Niet geoptimaliseerd voor simpele query's waar in een relationele databank geen JOIN nodig is

2.3.2 Knowledge Graphs

Google startte in mei 2012 een initiatief genaamd *Knowledge Graph* (**GoogleKnowledgeGraph**). Deze graaf wordt door Google gebruikt om de resultaten van de Google zoekmachine aan te vullen met informatie uit verscheidene bronnen, om zo beknopt aan de gebruikers de

meest belangrijke informatie te kunnen tonen zonder dat zij zelf op verschillende resultaten moeten klikken om hun antwoord te vinden.

De informatie uit deze Knowledge Graph is terug te vinden in een kader rechts van de gewone resultaten, zoals te zien is in volgende figuur. Dit kader werd door Google ook wel het 'Knowledge Panel' genoemd.



Sinds de uitvoering van de Google Knowledge Graph is deze term zelf bekend geraakt, en wordt deze algemeen gebruikt voor het representeren van kennis die gebaseerd is op een graaf.

Binnen de informatica worden knowledge graphs ook wel ontologieën genoemd, dit soort graaf houdt eigenlijk een verzameling van onderling gelinkte objecten en feiten bij, die zowel door mensen als computers eenvoudig en ondubbelzinnig te begrijpen zijn.

Op dezelfde manier dat Google zijn eigen Knowledge Graph gebruikt om haar zoekresultaten te verbeteren, kan er ook een graaf opgesteld worden voor andere toepassingen zoals een webshop. In de context van e-commerce toepassingen kan deze graaf gebruikt worden om relaties tussen producten te ontdekken.

Aangezien het doorlopen van een graaf zoals eerder besproken zeer snel kan gebeuren, is dit een ideale troef om gepersonaliseerde zoekresultaten te kunnen aanbieden aan gebruikers. **(E-CommerceKnowledgeGraph)**

Door middel van een graaf kan geanalyseerd worden welke soorten producten vaak samen gekocht of bekeken worden, alsook ontdekken in welke producten gebruikers van e-

commerce toepassingen geïnteresseerd zijn. Op basis hiervan kunnen aanbevelingen berekend en aan de gebruiker getoond worden.

In de context van aanbevelingssystemen worden knowledge graphs vooral gebruikt om extra kennis toe te voegen aan een graaf met relaties tussen klanten en producten, waardoor er onderling meer connecties kunnen gelegd worden tussen klanten en producten, hierdoor zijn de mogelijkheden voor aanbevelingen plots enorm veel groter.

Het cold-start probleem dat eerder bij het puntje aanbevelingssystemen besproken werd, wordt hier ook deels mee opgelost. Als er weinig relaties zijn tussen klant en product, of in het geval van een cold-start zelfs geen, kan de knowledge graph, die kennis heeft over de producten en hoe populair deze zijn, hier een handje bij toesteken om toch succesvol aanbevelingen aan de gebruiker te bieden.

In het onderzoek van **Grad-Gyenge2015** wordt een model voorgesteld waar attributen van bijvoorbeeld een persoon of een film wordt voorgesteld als een extra knoop in een graaf. Zo kunnen films gelinkt worden aan genres, jaar van release, acteurs, etc. en personen kunnen gelinkt worden aan een leeftijd, geslacht, locatie, etc. Zo kunnen er bogen in de graaf toegevoegd worden die personen met bijvoorbeeld dezelfde interesses en leeftijd groeperen.

Het is ook een mogelijkheid om gewichten toe te kennen aan deze relaties (bogen), zodat bijvoorbeeld interesse in genres harder doorweegt dan gelijkaardige leeftijd wanneer er een verband gelegd wordt tussen twee personen, op deze manier kunnen de aanbevelingen op maat gegenereerd worden voor andere toepassingen, zoals e-commerce, waar in sommige sectoren leeftijd misschien een belangrijkere relatie is dan regio.

2.3.3 Neo4j

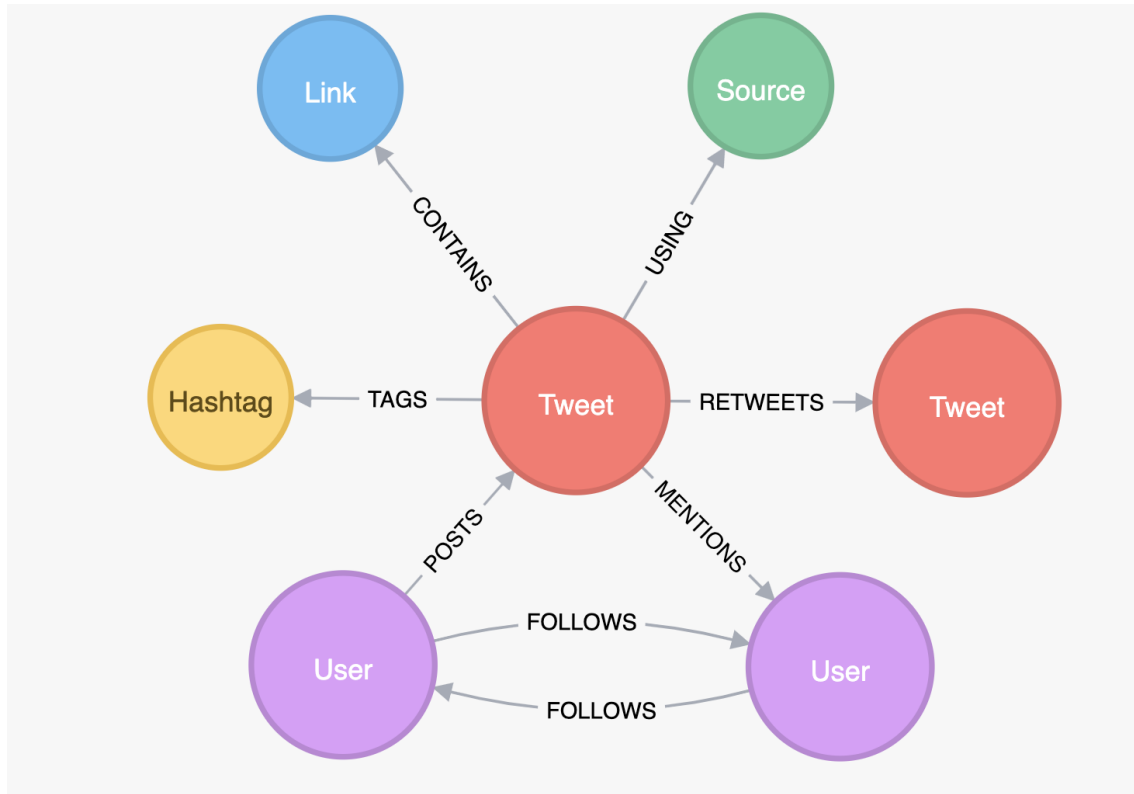
Er bestaan verschillende platformen voor SQL en relationele databanken, bijvoorbeeld Microsoft SQL Server, MySQL, Oracle, etc.

Zo zijn er ook verschillende platformen voor de hierboven beschreven NoSQL databanken, afhankelijk van welk type databank gezocht wordt. Neo4j is een van de meer bekende platformen voor graph databanken, dit platform zal gebruikt worden in dit onderzoek.

Walmart maakt gebruik van Neo4j om de aanbevelingen voor hun klanten op hun online webservices te optimaliseren (NeoTechnology, 2014). Zij gebruiken dit omdat graph databanken zeer snel over een gebruiker zijn koophistorie kunnen traverseren, en ook direct nieuwe mogelijke interesses kunnen halen uit het gedrag van de gebruiker. Daarmee wordt bedoeld dat er in real-time nieuwe connecties worden gelegd tussen de gebruiker en de producten, en hij de nieuwe aanbevelingen meteen zal zien, en niet enkele dagen of uren later. Er wordt dus historische data gematcht met real-time data, hier blinkt Neo4j in uit.

Neo4j maakt dus gebruik van wiskundige grafen om data weer te geven samen met hun onderlinge relaties. Een graaf kan gericht of ongericht zijn, ongericht wil zeggen dat er geen richting is waarin de relaties lopen, dus deze zijn onderling uitwisselbaar.

Een gerichte graaf is dan een graaf waarbij de relaties in een specifieke richting lopen, zoals volgers op Twitter: Persoon A volgt persoon B, maar persoon B volgt niet persoon A. Voor een voorbeeld van hoe een graaf voor Twitter er in simpele vorm uit zou kunnen zien, is zichtbaar in volgende figuur:



Performantie

In in hoofdstuk één van het boek 'Neo4j in Action' (**Vukotic2014**) wordt een onderzoek gedaan naar de snelheid van het uitvoeren van bepaalde query's zoals 'zoek vrienden van vrienden' op een dataset. De resultaten hiervan wijzen erop dat graph databanken beduidend sneller zijn in het traverseren van een graaf, bijvoorbeeld voor het zoeken op een bepaalde diepte vanaf een startpunt.

Het experiment bestaat eruit 'vrienden van vrienden' op te halen uit een databank, dit in zowel Neo4j als MySQL. De databank bestaat voor beide platformen uit 1 000 000 gebruikers, *Execution Time* wordt uitgedrukt in seconden per 1000 gebruikers.

In Tabel 2.1 worden de resultaten van dit experiment samengevat. De *Depht* slaat hier op hoeveel vrienden een persoon van een andere persoon verwijderd zal zijn, voor een diepte drie is dit dus 'vrienden van vrienden van vrienden'.

Uit deze resultaten wordt al snel duidelijk dat Neo4j drastisch beter presteert bij het zoeken binnen netwerken van entiteiten waar veel connecties tussen voorkomen, wat nogmaals bevestigt dat voor dit soort use-cases, graph databanken de beste keuze zijn.

Depth	Execution Time - MySQL	Execution Time - Neo4j
2	0.016	0.010
3	30.267	0.168
4	1,543.505	1.359
5	Not Finished in 1 Hour	2.132

Tabel 2.1: Source: <https://neo4j.com/news/how-much-faster-is-a-graph-database-really/>

2.4 Wat is Elasticsearch?

Elasticsearch op zichzelf is eigenlijk een zoekmachine die data kan analyseren, of dit nu nummers, tekst, gestructureerd of ongestructureerd is.

Elasticsearch is een component van de Elastic stack, ook wel ELK-Stack genoemd. Dit staat voor Elasticsearch, Logstash en Kibana. Logstash wordt gebruikt om data te verwerken uit meerdere bronnen en te versturen naar Elasticsearch. Kibana is een tool om de gebruikers een visueel beeld te geven van de data in de vorm van grafieken of tabellen.

Data wordt in Elasticsearch opgeslagen als een document in JSON-formaat in een index. Aangezien dit een zoekmachine is, is Elasticsearch dus gespecialiseerd in het zoeken van data, bijvoorbeeld een query om alle producten op te halen waar het woord 'switch' in voorkomt zal zeer snel resultaten opleveren.

Elasticsearch ondersteunt ook meerdere methodes voor het zoeken, eentje ervan is zoals hierboven beschreven het zoeken op een stukje tekst, waarbij alle documenten geretourneerd worden waar de tekst in een van de velden voorkomt, dit wordt tekst-gebaseerd zoeken genoemd. Hieronder enkele voorbeelden van zoekmethodes:

De zoekmethode die hier als voorbeeld genomen werd maakt gebruik van de parameter *query_string*. Als een van de velden van een document overeenkomt met de ingevoerde tekst, zal dit document in de lijst van resultaten terechtkomen.

```
1 {  
2   "query" :  
3   {  
4     "query_string":  
5     {  
6       "query": "deodorant"  
7     }  
8   }  
9 }
```

Listing 2.1: Tekst-gebaseerd zoeken: Query om een simpele zoekopdracht uit te voeren met de term 'deodorant'

Er kan ook op specifieke termen gezocht worden via *term*. Daarbij kan dan gespecificeerd worden welk veld moet overeenkomen met de ingevoerde tekst. Indien men wil zoeken op meerdere termen, kan terms vervangen worden door de parameter *terms*, en zal er een array mee moeten gegeven worden als input in plaats van 'Audio'

```
1 {  
2   "query" :  
3   {  
4     "term":  
5     {  
6       "category": "Audio"  
7     }  
8   }  
9 }
```

Listing 2.2: Tekst-gebaseerd zoeken: Query om een simpele zoekopdracht uit te voeren waar de categorie van het product overeen komt met 'Audio'

Een andere manier om te zoeken op basis van tekst is *match*. Hierbij worden alle woorden van alle velden van een document bekeken, en wordt er een score toegekend voor elke keer dit woord in het document voorkomt, op deze manier zullen resultaten waarbij dit woord het vaakst voorkomt, bovenaan in de lijst van resultaten terechtkomen.

```
1 {  
2   "query" :  
3   {  
4     "match":  
5     {  
6       "message": "accepted"  
7     }  
8   }  
9 }
```

Listing 2.3: Tekst-gebaseerd zoeken: Query die de resultaten sorteert op basis van hoe vaak het woord 'accepted' voorkomt in het veld 'message' van een document

Als men wil zoeken binnen een bepaalde prijsklasse, bijvoorbeeld tussen 30 en 50 euro, kan dit met de parameter *range*, bijvoorbeeld met volgende query:

```
1 {  
2   "query" :  
3   {  
4     "range":  
5     {  
6       "price": {  
7         "gte": 30,  
8         "lte": 50  
9       }  
10    }  
11  }  
12 }
```

Listing 2.4: Bereik zoeken: Een query om alle documenten op te halen waarbij het veld 'price' tussen 30 en 50 ligt

Het is ook mogelijk om query's te combineren, dit kan verwezenlijkt worden door het gebruik van een *bool query*. De parameters die hieronder gebruikt kunnen worden zijn:

- “*must*“ : De meegegeven waarde moet in het document voorkomen en de score van

deze query zal meetellen voor de totale score. Dit is een strikte voorwaarde, als hier niet aan voldaan wordt zal het document niet opgenomen worden in de resultaten.

- “*filter*“ : De meegegeven waarde moet in het document voorkomen. De score van deze query zal niet meetellen voor de totale score. Alle documenten die niet aan de voorwaarden van de filter voldoen, zullen net zoals bij ‘must’ niet opgenomen worden in de resultaten.
- “*should*“ : Als de gegeven waarde voorkomt in het document, zal de score van het document verhoogd worden, maar dit is geen strikte voorwaarde. De parameter “*minimum_should_match*“ geeft aan aan hoeveel voorwaarden voldaan moet worden om het document op de nemen in de resultaten. De “*boost*“ parameter laat toe bepaalde overeenkomsten een hogere score te laten genereren.
- “*must_not*“ : Het omgekeerde van ‘must’. De meegegeven waarde mag niet voorkomen in het document. Net zoals bij ‘filter’ zal de score voor deze query niet meetellen.

```
1 {
2   "query" :
3   {
4     "bool":
5     {
6       "must": {
7         "term" : { "name" : "Kim" }
8       },
9       "must_not": {
10        "term": { "lastname" : "Johnson" }
11      },
12      "filter": {
13        "term": { "gender" : "female" }
14      },
15      "should" : [
16        {
17          "range" : {
18            "age" : { "gte" : 20, "lte" : 30 }
19          }
20        },
21        {
22          "range" : {
23            "height" : { "gte" : 150, "lte" : 170 } ,
24            "boost" : 3
25          }
26        },
27        {
28          "term" : { "country" : "Belgium" }
29        },
30      ],
31      "minimum_should_match": 2
32    }
33  }
34 }
```

Listing 2.5: Combineren: Voorbeeld van een ‘bool’ query

Bovenstaande query zal alle documenten retourneren waar:

- De voornaam 'Kim' is
- De achternaam NIET 'Johnson' is
- Het geslacht 'female' is.

En er aan minstens twee van deze drie voorwaarden voldaan wordt:

- De leeftijd tussen 20 en 30 jaar is
- De lengte tussen 150 en 170 is
- Het land 'België' is.

In Elasticsearch is het dus mogelijk gewichten toe te kennen aan de resultaten van een zoekopdracht, zo kunnen meer relevante resultaten hoger in de lijst staan, dit is een belangrijke factor in dit onderzoek, namelijk of deze technologie gebruikt kan worden in E-Commerce toepassingen.

Een alternatieve en meer uitgebreide manier om scores toe te kennen aan resultaten is via een *“function_score”*. Het bovenste deel van deze query (*“query”*) zal bepalen welke documenten opgenomen moeten worden, en het onderste deel (*“functions”*) zal een lijst van functies vormen.

Aan de hand van deze functies zal een score voor een resultaat worden berekend. Aan elk van deze functies kan een gewicht (*“weight”*) worden toegekend om bepaalde overeenkomsten meer te laten doorwegen dan andere en deze dus een hogere score toe te kennen.

Er komen nog twee extra parameters kijken bij deze functies, namelijk *“score_mode”* en *“boost_mode”*. Elk document dat wordt opgehaald zal een score toegewezen krijgen.

De parameter *score_mode* bepaalt hoe de scores van deze functies gecombineerd worden. Mogelijke opties hiervoor zijn *“multiply”*, *“sum”*, *“avg”*, *“first”*, *“max”*, en *“min”*

De parameter *boost_mode* bepaalt hoe de score van de query en de functies gecombineerd moet worden. Mogelijke opties hiervoor zijn: *“multiply”*, *“replace”*, *“sum”*, *“avg”*, *“max”*, en *“min”*.

De meeste opties verklaren zichzelf, *“replace”* zal de score die toegekend werd aan het document bij het ophalen (*“query”*) overschrijven door de score die berekend werd door de functies.

Hieronder een voorbeeld van een *function_score* query waar alle documenten opgehaald worden, daarna zal er een score van 4 toegekend worden aan de documenten indien 'name' overeenkomt met 'Kim', en een score van 2 indien 'age' overeenkomt met '23'.

De scores van deze functies zullen worden opgeteld, en het resultaat hiervan zal de score die toegekend werd aan het resultaat bij het ophalen van het document overschrijven.

```
1 {
2   "query" : {
3     "function_score" : {
4       "query" : { "match_all" : {} },
5       "boost" : 2,
6       "functions" : [
7         {
8           "filter" : { "match" : { "name" : "Kim" } },
9           "weight": 4
10        },
11        {
12          "filter" : { "match" : { "age" : 23 } },
13          "weight": 2
14        }
15      ],
16      "score_mode" : "sum",
17      "boost_mode" : "replace"
18    }
19  }
20 }
```

Listing 2.6: Scoring: voorbeeld van een function_score query

De “*script_score*” functie laat toe om een score voor een document te berekenen aan de hand van de waarden van de velden in het document.

Onderstaande query zal alle documenten ophalen waarbij het veld ‘naam’ overeenkomt met ‘Kim’, en zal een score berekenen op basis van de lengte. In dit geval (300 - lengte) zal een lagere waarde voor dit veld resulteren in een kleinere score.

```
1 {
2   "query" : {
3     "script_score" : {
4       "query" : {
5         "match" : { "name" : "Sasha" }
6       },
7       "script": {
8         "
9           height = doc['height'].value;
10          return 300 - height;
11        "
12      }
13    }
14  }
15 }
```

Listing 2.7: Scoring: voorbeeld van een script_score query

Decay functies kunnen gebruikt worden om de score van een document aan te passen naar gelang een opgegeven veld afwijkt van een bepaalde waarde. Dit is gelijkaardig aan de ‘range’ query, waar met intervallen zou kunnen werken, maar loopt geleidelijk mee naargelang de afstand tussen de waarde van het veld en de opgegeven waarde groter wordt.

Er zijn drie mogelijke decay functies, namelijk “*linear*”, “*exp*”, en “*gauss*”. De velden die hierbij horen zijn:

- “*origin*” : Het startpunt voor het berekenen van de afstand. De waarde hiervan moet numeriek, een datum of een geo-punt zijn.
- “*scale*” : Het verschil dat nodig is om de score te doen afnemen met de waarde aangegeven door de decay.
- “*offset*” (Optioneel): Als deze waarde gedefinieerd wordt, zal de decay functie enkel berekend worden als de afstand groter is dan deze waarde. Default is 0.
- “*decay*” (Optioneel): Geeft aan met welke waarde de score van een document moet afnemen. Default is 0.5

```
1 {  
2   "query" : {  
3     "function_score" : {  
4       "exp" : {  
5         "date" : {  
6           "origin": "2013-09-17",  
7           "scale" : "10d",  
8           "offset" : "5d",  
9           "decay" : 0.5      }  
10        }  
11      }  
12    }  
13  }
```

Listing 2.8: Decay: voorbeeld van een decay functie

De bovenstaande query zal de score van een document laten afnemen met 0.5 naar gelang de waarde van het veld 'datum' verder ligt van de opgegeven waarde (2013-09-17). De waarde zal niet afnemen indien deze binnen de 5 dagen verwijderd ligt van de opgegeven waarde.

In het artikel van (Vavliakis, Katsikopoulos & Symeonidis, 2019) wordt beweerd dat het systeem dat zij implementeerden op een performante manier de gewenste resultaten gaf, alsook dat dit een oplossing kan zijn voor real-time zoekopdrachten in commerciële toepassingen. Dit sluit dus al uit dat Elasticsearch geen oplossing zou kunnen bieden voor onze onderzoeksvraag.

2.5 Filter Bubble

Een filter bubble, ook wel informatieluchtbel genaamd, is een gevolg van het personaliseren van zoekopdrachten. Personalisatie is het proberen bepalen welke zaken een gebruiker zou willen zien, aan de hand van een algoritme.

Een filter bubble betekent dat door deze personalisatie, een bijvoorbeeld geen producten zal zien die hem niet interesseren, in dit geval is dat een voordeel. Als we kijken in de context van controversiële onderwerpen, zorgt deze bubbel ervoor dat de gebruiker eigenlijk geen

informatie zal zien die niet bij zijn standpunt past, de gebruikers worden dus afgesloten in luchtbel bestaande uit enkel hun visie of standpunt. (Pariser, 2011)

Pariser geeft als voorbeeld in zijn boek een scenario waar een gebruiker de zoekterm “BP” invoerde, en resultaten kreeg in verband met investeringsnieuws over British Petroleum. Een tweede gebruiker gaf dezelfde zoekterm in en kreeg informatie over Deepwater Horizon als resultaat. Dit is dus een merkwaardig verschil.

Pariser beweert dat het effect van de filter bubble negatieve gevolgen kan hebben voor de manier waarop mensen hun mening vormen, dit aangezien zij geen argumenten te zien krijgen die hun mening tegenspreken.

De relevantie van dit fenomeen voor dit onderzoek valt op het feit dat bij de Content Based algoritmen die eerder besproken werden, we met een gelijkaardig probleem te maken hadden. Bij deze aanbevelingssystemen kan er overspecialisatie optreden, waardoor de gebruiker enkel producten aanbevolen krijgt die reeds gezien zijn, of té gelijkaardig zijn, en er dus geen nieuwe verrassende producten aangeboden worden.

2.6 GDPR

Op 25 mei 2018 is de *General Data Protection Regulation* (GDPR) ingevoerd. Dit is een verzameling van regels die bestaat om de persoonlijke data van inwoners van de EU te beschermen. Met persoonlijke data wordt verwezen naar data die direct of indirect kan gelinkt worden aan een persoon, het gaat hierbij over data zoals IP-adressen en cookies.

Onder deze regulatie is het verplicht voor bedrijven om toestemming te vragen aan hun gebruikers om persoonlijke data te mogen opslaan en verwerken. Meestal gebeurt dit in de vorm van een knop waar de gebruiker op moet klikken om cookies toe te staan, en zelfs daar kan de gebruiker soms nog zelf enkele opties aanvinken over welke data hij wil vrijgeven aan het bedrijf. (Goddard2017)

Enkele onderdelen die van belang zijn voor e-commerce toepassingen worden verder besproken in de volgende hoofdstukken.

2.6.1 Gegevensverwerking

Bedrijven mogen onder deze regulatie niet zomaar eender welke persoonlijke gegevens van hun gebruikers verwerken. Er werden enkele regels opgesteld die bedrijven moeten volgen indien zij toch wensen om persoonlijke data te verwerken, en er moet aangetoond kunnen worden dat deze regels nageleefd en gerespecteerd worden. Deze regels zijn terug te vinden in artikel 5 van de GDPR, genaamd “Beginselen inzake verwerking van persoonsgegevens” (Article5GDPR2018).

Het verwerken van persoonsgegevens moet rechtmatig, behoorlijk en transparant gebeuren. Dit wil zeggen dat er aan minstens één van de voorwaarden in Artikel 6 van de GDPR

voldaan moet worden.

- Een persoon geeft expliciet toestemming voor het verwerken van zijn gegevens.
- De verwerking is noodzakelijk in het kader van een wettelijke plicht of om de vitale belangen van een persoon te beschermen
- De verwerking is noodzakelijk voor het algemeen belang of de uitoefening van het openbaar gezag
- De verwerking is noodzakelijk voor een gerechtvaardigd belang.

(Article6GDPR2018)

De regulatie verplicht ook dat slechts het minimum aan persoonlijke data van een gebruiker wordt opgeslagen, en moet dus beperkt blijven tot de data die van belang is voor de doelen waarvoor die data gebruikt zal worden.

Als er wordt gesproken over transparantie, wil dit zeggen dat er geen twijfel mogelijk mag zijn over hoe de persoonlijke data van een gebruiker verwerkt wordt. Ook dat enkel en alleen de verwerkingen waar een gebruiker mee instemt effectief uitgevoerd worden, en niets meer.

2.6.2 Rechten van de gebruikers

Indien een persoon de persoonlijke data wenst op te vragen, verplicht de wetgeving rond GDPR bedrijven om deze data, samen met andere informatie op te leveren, al dan niet in digitaal formaat.

Deze oplevering van persoonlijke data moet binnen een redelijke tijdspanne gebeuren, en er mogen dus geen onrealistische vertragingen plaatsvinden hierbij. Enige uitzondering hiervoor is wanneer een van de condities in Artikel 17, punt 3 van toepassing zijn.

Een volledige beschrijving van deze informatie is terug te vinden in Artikel 15 van de GDPR, “recht van inzage van de betrokkene“. **(Article15GDPR2018)**

Een persoon heeft het recht om de persoonlijke data te laten wissen. Dit moet zonder onredelijke vertraging kunnen gebeuren, tenzij een van de uitzonderingen in Artikel 17, punt 3 van toepassing zijn.

Persoonlijke gegevens moeten gewist worden wanneer:

- Deze gegevens niet langer nodig zijn voor de doelen waarvoor deze verzameld zijn.
- De betrokken persoon trekt zijn toestemming voor de verwerking in.
- De betrokken persoon maakt bezwaar tegen de verwerking.
- De gegevens zijn onrechtmatig verwerkt.
- Om te voldoen aan een recht van de betrokken persoon.

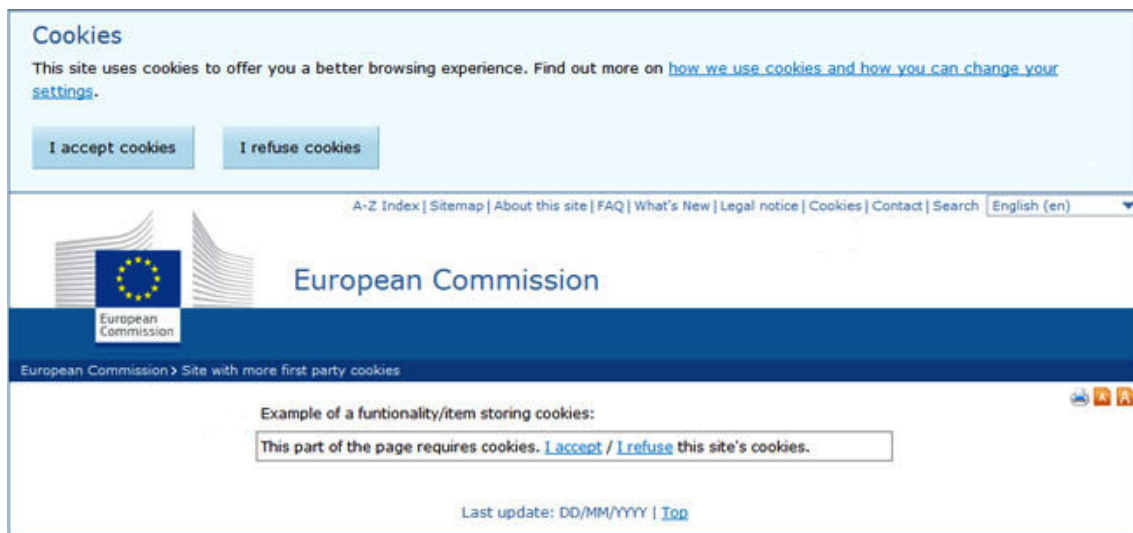
(Article17GDPR2018)

2.6.3 Effect van de GDPR op e-commerce

In het geval van e-commerce toepassingen zijn de voorwaarden over wettelijke plicht of een gerechtvaardigd belang niet van toepassing. Indien een e-commerce website de data van een gebruiker zou willen verwerken, zal deze expliciete toestemming moeten vragen aan de gebruiker.

Een gebruiker moet op transparante wijze ingelicht worden over hoe zijn gegevens verwerkt zullen worden, en dus voor welke doeleinden deze gegevens zullen dienen. Er zal ook geen data mogen bijgehouden worden die niet gebruikt wordt.

Een evidente keuze hiervoor is om gebruik te maken van cookies, en hiervoor expliciete toestemming te vragen aan de gebruiker. Er zal dan een privacybeleid moeten worden opgesteld waar in duidelijke en heldere taal beschreven staat welke data verwerkt zal worden, en voor welke doeleinden.



Er mogen geen dubbelzinnigheden of onduidelijkheden terug te vinden zijn, en er mag geen informatie achterwege gelaten worden. Ook informatie over welke data naar derden gestuurd wordt, en voor welke redenen moet hier duidelijk vermeld worden.

3. Methodologie

Rekeninghoudend met de bevindingen uit de literatuurstudie, zal hier verder besproken worden hoe we een systeem gaan opstellen dat gepaste aanbevelingen kan maken. Gezien we twee verschillende werkwijzen en technologieën zullen vergelijken, zal voor beide systemen een werkwijze voorgesteld worden.

3.1 Uitwerking Elasticsearch

Voor de uitwerking van Elasticsearch wordt een instantie opgezet waarmee we zullen communiceren, op deze instantie draait Elasticsearch.

In Elasticsearch worden items opgeslagen als een document met enkele waarden, typisch aan een product. In dit onderzoek zijn deze waarden; 'naam', 'merk', 'categorieën', en 'prijs'.

3.1.1 Producten

Voor het aanmaken van de product data zullen enkele query's uitgevoerd worden die er als volgt uitzien:

```
1 POST http://35.233.112.106:9200/products/product/1
2 {
3   "name": "Logitech G930",
4   "brand": "Logitech",
5   "categories": ["Headset", "Wireless headset", "Headphones"],
6   "price": "190.00"
```

```
7 }
```

Listing 3.1: Query om één enkel product aan te maken

Op deze manier worden enkele producten in de databank gezet, met enkele verwante velden zoals gelijkaardige categorieën, zodat we hiermee aanbevelingen kunnen maken. Deze verzameling van producten wordt een index genoemd.

Vervolgens kunnen we een zoekterm ingeven via volgende query, deze zal alle resultaten weergeven waarvan een van de velden voldoet aan de meegegeven waarde, namelijk "deodorant".

```
1 GET http://35.233.112.106:9200/products/product/1
2 {
3   "query" : {
4     "query_string": {
5       "query": "deodorant"
6     }
7   }
8 }
```

Listing 3.2: Query om één enkel product op te halen

3.1.2 Gebruikers

Er zullen enkele vooraf gedefinieerde gebruikers opgesteld worden, waarmee bepaald zal worden of de zoekresultaten aan de verwachtingen voldoen. Een voorbeeld van een gebruiker is te zien in volgende query:

```
1 POST http://35.233.112.106:9200/users/user/1
2 {
3   "name": "Louise",
4   "age": "21",
5   "address": {
6     "city": "Aalst",
7     "street": "Molendries 4",
8     "province": "Oost-Vlaanderen"
9   },
10  "categories": ["Electronics", "Apple"],
11  "searches": ["iphone", "deodorant", "uncommon product"],
12  "brands": ["Nivea", "Apple"]
13 }
```

Listing 3.3: Query om één enkele gebruiker aan te maken

Bij deze gebruiker kunnen we bijvoorbeeld verwachten dat als deze 'deodorant' opzoekt, zij die van het merk Nivea bovenaan de resultaten zal zien. Een voorbeeld van zo'n zoekopdracht, waarbij sommige velden ingevuld zijn op basis van onze persoon van de query hierboven, gaat als volgt:

```
1 POST http://35.233.112.106:9200/products/_search
2 {
3   "query": {
```

```
4   "function_score": {
5     "query": {
6       "query_string": {
7         "query": "Deodorant"
8       }
9     },
10    "functions": [
11      { "filter" :
12        { "terms" :
13          { "brand" : ["Niveau", "Apple"]  }
14        },
15        "weight": 3
16      },
17      { "filter" :
18        { "terms" :
19          { "categories" : ["Electronics", "Apple"] }
20        },
21        "weight": 2
22      },
23      { "filter" :
24        { "terms" :
25          { "searches" : ["iphone", "deodorant", "uncommon
product"] }
26        },
27        "weight": 1
28      }
29    ],
30    "score_mode": "sum",
31    "boost_mode": "replace"
32  }
33 }
34 }
```

Listing 3.4: Query om een zoekopdracht met term 'Deodorant' uit te voeren, met filters en een score op basis van informatie van een gebruiker

In het bovenste deel van de query, waar de *query_string* wordt aangeduid. Deze string zou dan overeenkomen met wat een gebruiker zou invoeren in een zoekbalk op een website.

In de realiteit zal de informatie over 'brand', 'categories', en 'searches' opgehaald worden uit het model van de persoon in kwestie.

In deze query wordt een score toegekend aan de resultaten die voldoen aan bovenstaande verwachting, namelijk dat het woord 'deodorant' terug te vinden moet zijn in een van de velden van het product (naam, merk, categorie)

Die score wordt toegekend op basis van de informatie over een persoon, hieruit verstaan we dat dit gaat over de merken die deze persoon reeds gekocht heeft, in welke categorieën deze persoon reeds gekocht heeft, en een historiek van zoekopdrachten. Deze hebben elk een gewicht toegekend gekregen, respectievelijk drie, twee en één.

In de query zijn drie filters te zien die de score van een resultaat zullen beïnvloeden. Bij elk van deze filters wordt het score vermenigvuldigd met het gewicht. *score_mode* : *sum*

zorgt ervoor dat de resultaten van de scores opgeteld worden.

boost_mode : replace zorgt ervoor dat de score die verkregen wordt bij het gelijkaardig zijn aan de zoekterm vervangen wordt door de nieuw berekende score van de functies.

Deze query is een vorm van collaborative filtering, toegepast op zichzelf. De query zal dus gaan zoeken naar hoe verwant een resultaat is aan een persoon, op basis van enkele variabelen, en zal deze een hogere score toekennen afhankelijk van hoe relevant het product is voor een gebruiker. Resultaten met een hogere score zullen dus bijgevolg ook hoger in de lijst van resultaten komen te staan.

3.1.3 Graph API Plug-in

Elasticsearch biedt ook een plug-in aan die de functionaliteiten van een graph database voorziet. Met deze plug-in bekomen we dus eigenlijk een Knowledge Graph zoals eerder in de literatuurstudie vermeld werd.

Deze plug-in laat toe om connecties te vinden tussen objecten in de databank, en wordt geadverteerd als ook bruikbaar te zijn voor aanbevelingen.

Jammer genoeg is deze plug-in enkel beschikbaar wanneer we beschikken over een Platinum of Enterprise licentie. Door deze vereiste valt deze functionaliteit dus buiten de scope van het onderzoek.

	OPEN SOURCE	BASIC	GOLD	PLATINUM	ENTERPRISE
SEARCH & ANALYSIS					
Graph exploration	—	—	—	✓	✓

Figuur 3.1: Overzicht van de beschikbaarheid van de Graph plugin voor de Elastic licenties

Source: <https://www.elastic.co/subscriptions>

3.1.4 Conclusie ElasticSearch

Elasticsearch is zeer snel in het vinden van producten die relateren aan de gebruiker zelf, maar is wat moeilijker in omgang met het gebruik maken van data die personen aan elkaar linkt. Binnen dit onderzoek is er geen manier ontdekt waarbij er rekening kan gehouden worden met familiale verbanden.

Het rekening houden met andere variabelen van een persoon zoals geslacht, leeftijdscategorie, etc. zijn wel mogelijk bij het genereren van aanbevelingen bij een zoekopdracht. Dit kan door middel van gebruikers te vergelijken op basis van bijvoorbeeld hun gemeenschappelijke zoekopdrachten.

3.2 Uitwerking Neo4j

Voor dit systeem wordt er een model opgesteld in een Graph databank, de gekozen technologie hiervoor is Neo4j. Op deze graaf kunnen dan zoekalgoritmes uitgevoerd worden, om zo tot correcte aanbevelingen te komen. In dit hoofdstuk zal verder uitgewerkt worden hoe dit precies in elkaar zit.

3.2.1 Model

Relaties tussen producten en klanten zullen als volgt worden opgeslagen worden in deze Graph databank:

Er zijn 2 soorten nodes, dit zijn de knopen van een graaf, namelijk 'Gebruiker' en 'Product'. De relaties, aangeduid door lijnen tussen de knopen, zullen dan voorstellen wat de interactie van een gebruiker met een product is. Dit kan 'LIKES', 'BOUGHT', of 'LIVES_TOGETHER' zijn. 'LIVES_TOGETHER' zal dan aanduiden of 2 gebruikers familie of samenwonend zijn.

De attributen van een Product zijn als volgt:

- Product ID
- Name
- Brand

De attributen voor een Gebruiker zijn als volgt:

- Name
- Address

De verschillende soorten relaties zijn als volgt:

- 'BOUGHT' -> Gebruiker heeft Product gekocht
- 'LIKES' -> Gebruiker heeft Product leuk gevonden
- 'LIVES_TOGETHER' -> Gebruiker 1 heeft hetzelfde adres als Gebruiker 2

De query die gebruikt wordt om de producten en gebruikers aan te maken:

```
1 CREATE
2 (shauni:Customer {name: 'Shauni', address: 'Exterkenstraat 14'}),
3 (lynn:Customer {name: 'Lynn', address: 'Exterkenstraat 14'}),
4 (angelo:Customer {name: 'Angelo', address: 'Arbeidstraat 14'}),
5 (nicolas:Customer {name: 'Nicolas', address: 'Kouter 3'}),
6 (wendy:Customer {name: 'Wendy', address: 'Arbeidstraat 14'}),
7 (prod1:Product{id: '1', name:'Hairbrush', brand:'Syoss'}),
8 (prod2:Product{id: '2', name:'Instant Chocolate Milk', brand:'
  Nesquik'}),
9 (prod3:Product{id: '3', name:'Toilet Paper', brand: 'Boni'}),
10 (prod4:Product{id: '4', name:'Pen', brand:'Stabilo'}),
11 (prod5:Product{id: '5', name:'Roller Deodorant', brand: 'Sanex'}),
```

12 ...

Listing 3.5: Neo4j query voor het aanmaken van producten en klanten

Om een relatie aan te maken tussen een product en een gebruiker zal volgende query gebruikt worden. Deze zal dus uitgevoerd worden elke keer een gebruiker een actie uitvoert met een product. Aangezien een product leuk vinden en een product kopen niet aanzien worden als evenwaardig, zullen er gewichten toegekend worden aan de interacties tussen gebruikers en producten. Dat kan door middel van volgende query:

```
1 MATCH (c:Customer),(p:Product)
2 WHERE c.name = 'Shauni' AND p.id=3
3 CREATE (c)-[r:BOUGHT]->(p)
4 SET r.score = 3
```

Listing 3.6: Neo4j query voor het aanmaken van een relatie tussen een product en een klant

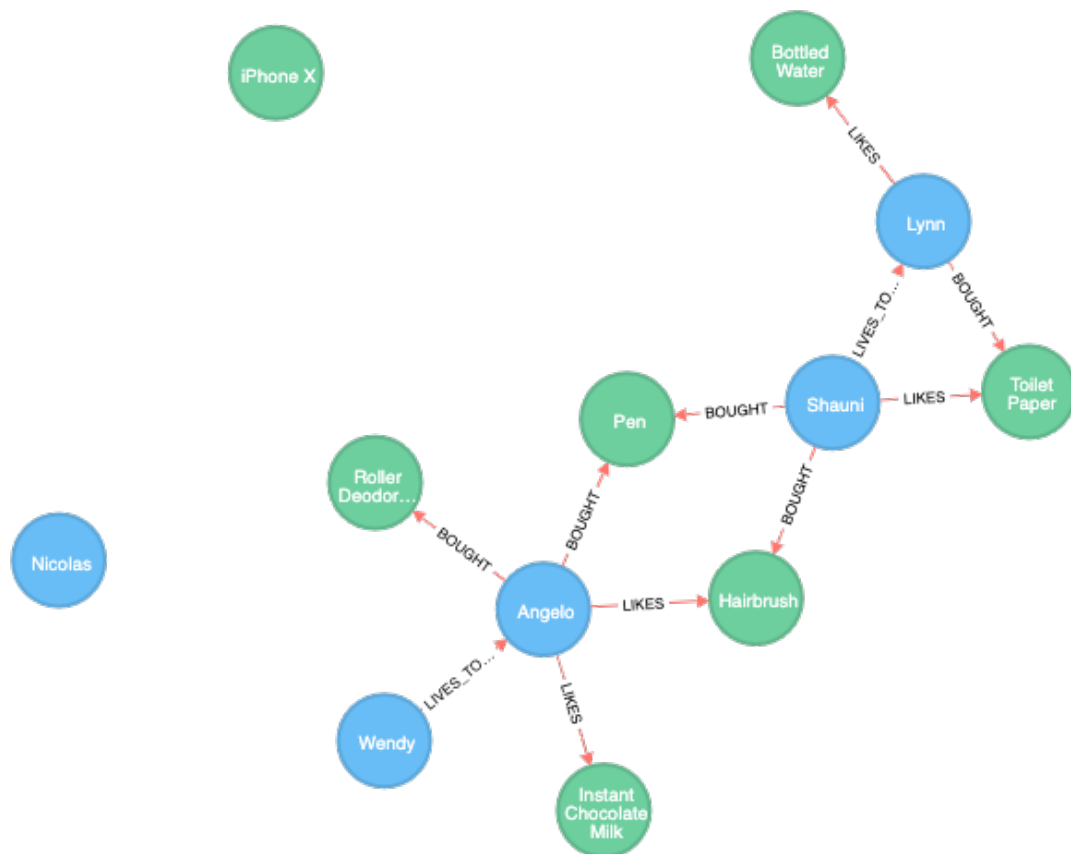
Een product leuk vinden krijgt een score 2 toegekend, en een product kopen zal de score 3 krijgen, om de relatie 'LIKES' aan te maken zal in bovenstaande query de relatie 'BOUGHT' en de score moeten aangepast worden.

Om de relatie van Gebruikers die op eenzelfde adres wonen aan te maken, gebruiken we volgende query:

```
1 MATCH (a:Customer), (b:Customer)
2 WHERE EXISTS (a.address) AND EXISTS (b.address) AND a.address=b.
   address AND id(a)<id(b)
3 CREATE (a)-[:LIVES\_TOGETHER]->(b);
```

Listing 3.7: Neo4j query voor het aanmaken van een relatie tussen een twee klanten.

De uiteindelijke graaf die voor dit zeer klein voorbeeld opgebouwd werd, zal al snel een beter inzicht geven in hoe deze data aan elkaar vast hangt. Dit is ook een van de troeven van Neo4j, het voorstellen van duidelijke visuele beelden die het eenvoudig maken om te begrijpen wat er zich op de achtergrond afspeelt. De graaf is te zien in Figure 3.2



Figuur 3.2: Overzicht van de ingevoerde data in een Neo4j graaf

3.2.2 Ophalen van data

Voor deze uitwerking werden enkele query's opgesteld die op basis van de relaties producten retourneren die zouden kunnen dienen als aanbevelingen.

Zo kan er bijvoorbeeld een lijst van producten gegenereerd worden die leuk gevonden of gekocht werden door de familieleden van een bepaalde gebruiker, dit gebeurt op basis van de relatie `LIVES_TOGETHER`.

```

1 MATCH (c1: Customer)
2 WHERE c1.name = 'Shauni'
3 MATCH (c1) - [:LIVES_TOGETHER] -> (c2 : Customer) - [:BOUGHT] -> (p
  :Product)
4 RETURN p as Product
5 UNION
6 MATCH (c1) - [:LIVES_TOGETHER] -> (c2) - [:LIKES] -> (p:Product)
7 RETURN p as Product

```

Listing 3.8: Neo4j query die alle likes en aankopen van de familie van een bepaalde gebruiker `c1` weergeeft

Ook van belang is het ophalen van producten die gebruikers kopen waar een gebruiker `c1` gemeenschappelijke aankopen mee heeft. Dit kan via volgende query:

```

1 MATCH (c1: Customer)
2 WHERE c1.name = 'Shauni'
3 MATCH(c1) - [:BOUGHT] -> (p1 : Product) <- [:BOUGHT] - (c2:
   Customer) - [:BOUGHT] -> (p2: Product)
4 RETURN p2

```

Listing 3.9: Neo4j query die de aankopen van gebruikers weergeeft waar een gebruiker *c1* gemeenschappelijke aankopen mee heeft

Om hetzelfde te bereiken maar dan inclusief de likes van deze gebruikers waar een gebruiker *c1* gemeenschappelijke aankopen mee heeft, kan volgende query gebruikt worden:

```

1 MATCH (c1) - [:BOUGHT] -> (p1 : Product) <- [:BOUGHT] - (c2:
   Customer) - [:BOUGHT] -> (p2: Product)
2 RETURN p2 as Product
3 UNION ALL
4 MATCH (c1) - [:BOUGHT] -> (p1) <- [:BOUGHT] -(c2) - [:LIKES] -> (p2
   )
5 WHERE id(c1)<id(c2)
6 RETURN p2 as Product

```

Listing 3.10: Neo4j query die de aankopen en likes van gebruikers weergeeft waar een gebruiker *c1* gemeenschappelijke aankopen mee heeft

3.2.3 Search

Binnen neo4j is het ook mogelijk om zoekopdrachten uit te voeren op basis van een tekstveld. Hiervoor moet op voorhand een index aangemaakt worden waarin beschreven staat welke attributen van welke types van nodes gebruikt mogen worden om de ingevoerde tekst mee te vergelijken.

```

1 CALL db.index.fulltext.createNodeIndex("namesAndCategories",["
   Product"],["name", "category"])

```

Listing 3.11: Een index creëren binnen Neo4j om zoekopdrachten op basis van tekst uit te voeren

In dit voorbeeld gebruiken we de attributen *name* en *categories* van het nodetype *Product*, en nemen we als zoekterm “deodorant”. Op deze manier kan er dus eenvoudig gekozen worden op welke velden van de objecten er moet vergeleken worden.

```

1 CALL db.index.fulltext.queryNodes("namesAndCategories", "deodorant
   ") YIELD node, score
2 RETURN node.name, node.category, score

```

Listing 3.12: Een zoekopdracht uitvoeren op basis van tekst

Er zal een lijst geretourneerd worden met de attributen (*node.name*), en een score die door het algoritme werd toegekend die aantoont hoe sterk het resultaat gelijkend is aan de ingevoerde tekst.

3.2.4 PageRank

Voor het vinden van resultaten die belang hebben voor de gebruiker, wordt het PageRank algoritme gebruikt. Dit algoritme berekent de belangrijkheid van een knoop op basis van zijn burens. Hoe meer connecties er naartoe, hoe sterker het resultaat. Op deze manier worden populaire resultaten hoger gerangschikt.

Om via dit algoritme aanbevelingen te verkrijgen, moet er eerst een nieuwe graaf worden opgesteld waar enkel de informatie in zit die nodig is voor de berekening, dit gebeurt via volgende query, waarin de interacties 'BOUGHT' en 'LIKES' opgenomen worden, samen met hun gewicht.

```

1 CALL gds.graph.create.cypher(
2   'graph1',
3   'MATCH (p:Product) RETURN id(p) AS id',
4   'MATCH (p2:Product)<-[:BOUGHT|LIKES]-(c:Customer)-[:BOUGHT|LIKES]
5   ->(p3:Product)
6   RETURN
7     id(p2) AS source,
8     id(p3) AS target,
9     r.weight AS weight',
10  {
11    readConcurrency: 4
12  })

```

Listing 3.13: Een genoemde graaf creëren om graafalgoritmen op uit te voeren

Vervolgens kunnen we het pageRank algoritme uitvoeren op deze graaf, die een lijst van productnamen zal weergeven met daarnaast een score.

```

1 CALL gds.pageRank.stream('graph1', { maxIterations: 10,
2   dampingFactor: 0.85 })
3 YIELD nodeId, score
4 RETURN gds.util.asNode(nodeId).name AS name, score
5 ORDER BY score DESC, name ASC

```

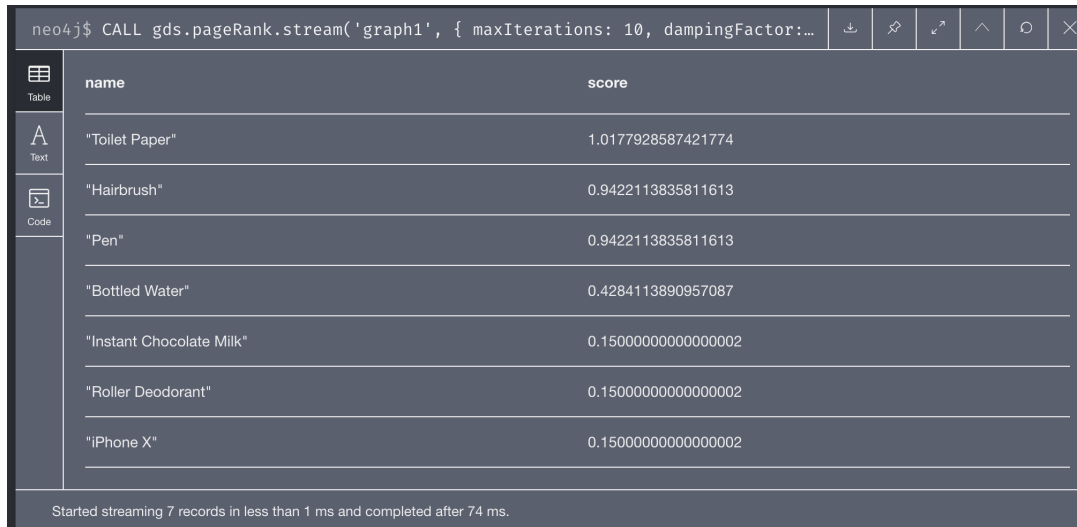
Listing 3.14: PageRank algoritme uitvoeren

Het resultaat van dit algoritme (Figure 3.3) toont ons dat “Toilet Paper” de belangrijkste knoop is in deze graaf, met andere woorden dat als er door de graaf gelopen wordt, deze knoop de grootste kans heeft om bereikt te worden.

3.2.5 Personalized PageRank

Om de resultaten te baseren vanuit een bepaald startpunt, bijvoorbeeld een Customer, kan er een variant op PageRank gebruikt worden. Het algoritme zal dan starten in een bepaalde knoop (de gebruiker in kwestie) en zal op deze manier de meest relevante knopen vinden.

We gebruiken opnieuw de genoemde graaf uit de simpele PageRank, en voeren daar volgend algoritme op uit:



name	score
"Toilet Paper"	1.0177928587421774
"Hairbrush"	0.9422113835811613
"Pen"	0.9422113835811613
"Bottled Water"	0.4284113890957087
"Instant Chocolate Milk"	0.15000000000000002
"Roller Deodorant"	0.15000000000000002
"iPhone X"	0.15000000000000002

Started streaming 7 records in less than 1 ms and completed after 74 ms.

Figuur 3.3: Resultaat PageRank algoritme

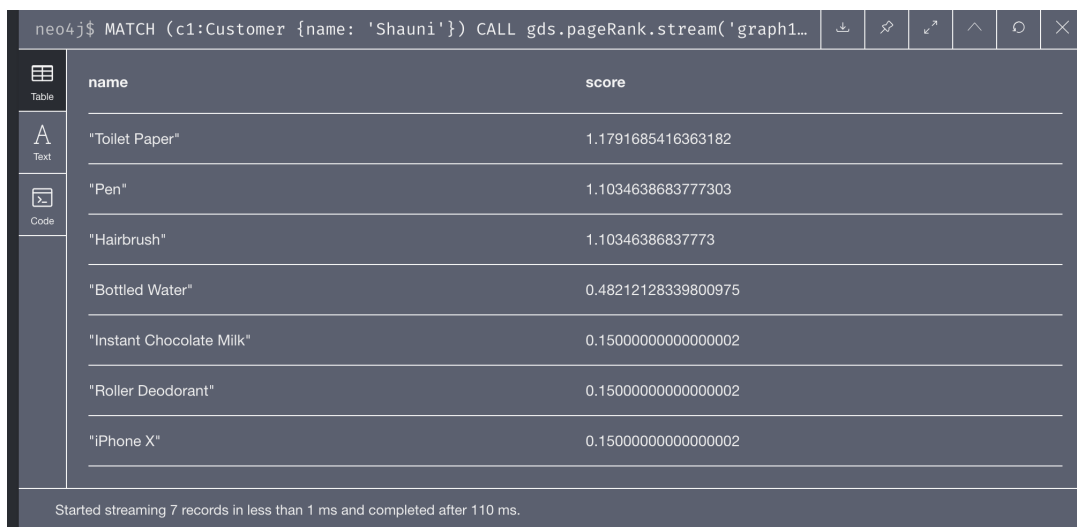
```

1 MATCH (c1:Customer {name: 'Shauni'})
2 CALL gds.pageRank.stream('graph1', {
3   maxIterations: 20,
4   dampingFactor: 0.85,
5   sourceNodes: [c1]
6 })
7 YIELD nodeId, score
8 RETURN gds.util.asNode(nodeId).name AS name, score
9 ORDER BY score DESC, name ASC

```

Listing 3.15: Personalized PageRank algoritme

In de resultaten is nu zichtbaar dat voor deze specifieke klant nog steeds “Toiler Paper” een aanbeveling is, maar dat het product met naam “Hairbrush” een hogere score krijgt dan “Pen”, wat in het algemene PageRank algoritme niet zo is.



name	score
"Toilet Paper"	1.1791685416363182
"Pen"	1.1034638683777303
"Hairbrush"	1.10346386837773
"Bottled Water"	0.48212128339800975
"Instant Chocolate Milk"	0.15000000000000002
"Roller Deodorant"	0.15000000000000002
"iPhone X"	0.15000000000000002

Started streaming 7 records in less than 1 ms and completed after 110 ms.

Figuur 3.4: Resultaat Personalized PageRank algoritme

3.2.6 Community Detection

Door het analyseren van de graaf met een community detection-algoritme kunnen groepen van gelijkaardige producten ontdekt worden. Op deze manier kan ontdekt worden welke gebruikers geïnteresseerd zijn in bepaalde groepen van producten, en kunnen er op basis van deze clusters aanbevelingen voorgesteld worden.

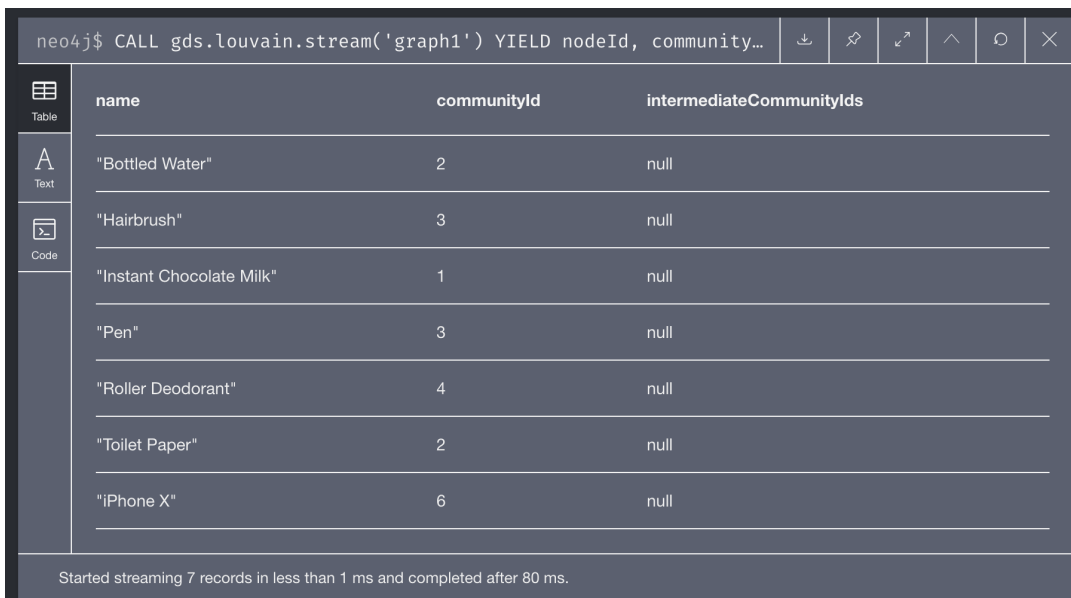
Een nadeel aan deze techniek is dat deze last heeft van het cold-start probleem. Met andere woorden dat voor nieuwe gebruikers of producten, waarvoor dus nog niet bekend is tot welke cluster ze behoren of in welke clusters zij interesse hebben, nog geen correcte aanbevelingen kunnen bepaald worden.

Een voorbeeld van een community detection algoritme is het Louvain algoritme. De genoemde graaf van de uitwerking van PageRank zal opnieuw worden gebruikt voor dit algoritme.

```
1 CALL gds.louvain.stream('graph1')
2 YIELD nodeId, communityId, intermediateCommunityIds
3 RETURN gds.util.asNode(nodeId).name AS name, communityId,
   intermediateCommunityIds
4 ORDER BY name ASC
```

Listing 3.16: Personalized PageRank algoritme

Uit de resultaten worden twee clusters ontdekt, met id 2 en 3, deze producten worden door het algoritme aanzien als gelijkaardig te zijn, omdat deze door een bepaalde groep gebruikers gekocht of leuk gevonden wordt.



name	communityId	intermediateCommunityIds
"Bottled Water"	2	null
"Hairbrush"	3	null
"Instant Chocolate Milk"	1	null
"Pen"	3	null
"Roller Deodorant"	4	null
"Toilet Paper"	2	null
"iPhone X"	6	null

Started streaming 7 records in less than 1 ms and completed after 80 ms.

Figuur 3.5: Resultaat Louvain algoritme

4. Conclusie

4.1 GDPR

Rekeninghoudend met de informatie die verkregen werd bij de literatuurstudie, kan er een antwoord geleverd worden op de onderzoeksvraag.

Indien de regels van de GDPR nageleefd worden inzake het verwerken van informatie van personen, kan er een systeem opgezet worden dat volgbaar is aan de wetgeving, en toch in staat is gepersonaliseerde zoekopdrachten uit te voeren.

Er zal een eenduidig en ondubbelzinnig beleid moeten worden opgesteld dat de gebruiker al dan niet kan accepteren. Indien de gebruiker niet akkoord gaat met het beleid, of zijn informatie slechts deels ter beschikking wenst te stellen, zal hierbij rekening gehouden moeten worden dat de resultaten van zoekopdrachten in sommige gevallen minder of zelfs niet gepersonaliseerd kunnen worden.

De keuze moet hierbij volledig in handen van de gebruiker liggen, en deze moet via het beleid op de hoogte gesteld worden van de doeleinden waarvoor zijn informatie gebruikt zal worden, om zo een beslissing te maken. In de voorgestelde toepassing zal dus duidelijk vermeld moeten staan dat zijn persoonlijke informatie enkel en alleen gebruikt wordt voor het aanbieden van gepersonaliseerde zoekresultaten.

Bij het aanmaken van een account zal nogmaals een beleid moeten opgesteld worden, waarin duidelijk vermeldt staat waarvoor de data die ingevuld wordt bij de creatie van het account gebruikt zal worden. In deze toepassing zal het dus gaan over de gegevens over de woonplaats van een gebruiker, leeftijdscategorie, naam, etc.

Bijvoorbeeld voor het gebruik van de data over de woonplaats zal dus duidelijk verwoord moeten worden dat deze data gebruikt wordt voor het linken van personen binnen één gezin.

Ook zal er geen data mogen bijgehouden worden die niet relevant is voor de doelstellingen, gegevens zoals bijvoorbeeld een telefoonnummer zijn niet van belang voor onze doelstellingen, en mogen dus niet verwerkt worden. De gebruiker moet steeds in staat zijn om zijn gegevens op te vragen of deze te laten wissen, en dit moet steeds zonder onredelijke vertraging gebeuren.

Hiermee is de onderzoeksvraag beantwoordt; ja, het is mogelijk om een systeem op te stellen met een gepersonaliseerde zoekfunctie die als legaal wordt aanzien onder de wetgeving omtrent GDPR.

4.2 ElasticSearch

Elasticsearch is zeer snel en eenvoudig in het gebruik als het gaat over het vinden van producten die relateren aan de gebruiker zelf. Uit het onderzoek bleek ook dat specifieke zoektermen en filters relatief eenvoudig te implementeren zijn.

Elasticsearch biedt ook de mogelijkheden om een graaf met de relaties tussen de verschillende documenten op te slaan, maar deze is niet zomaar verkrijgbaar, hier is een Platinum of Enterprise licentie voor vereist. Het is mogelijk om op deze manier dan toch rekening te kunnen houden met familiale relaties, alsook filters toe te passen op de zoekopdrachten.

4.3 Neo4j

Neo4j, als graph databank, staat toe om gebruik te maken van alle relaties tussen de verschillende producten (nodes) om aanbevelingen te genereren. Ook zijn er enkele algoritmen reeds ingebouwd in een Graph Data Analysis plugin, die zorgen voor de meest performante werkwijze.

Qua eenvoud van implementatie is Neo4j de betere keuze, mede met dank aan het feit dat alles visueel kan toegelicht worden door simpelweg naar de graaf te kijken.

Zowel het gebruik van de graaf, als het zoeken op basis van een tekstveld zijn mogelijk in Neo4j zonder dat hier extra kosten aan vast hangen voor licenties.

4.4 Resultaat

Er kan dus geconcludeerd worden dat elk van deze technieken uitblinkt in zijn eigen branche, de literatuurstudie wees ook uit dat een Knowledge Graph gebruikt kan wor-

den om aanvullende informatie te verschaffen bij het genereren van gepersonaliseerde zoekresultaten.

Een optie is dus om beide platformen te combineren, en elkaar te laten aanvullen. Een voorbeeld hier van is om de resultaten van een zoekalgoritme in Neo4 te combineren met de scores die gevonden werden in Elasticsearch.

Deze ontdekking zet de weg open tot verder onderzoek in hoe deze twee platformen efficiënt samen gebruikt kunnen worden, en hoe de modellen die in dit onderzoek werden opgebouwd, verder gespecialiseerd kunnen worden om deze combinatie mogelijk te maken.

A. Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1 Introductie

De meeste e-commerce online platformen hebben reeds een zoekfunctie ingebouwd, maar deze is vaak beperkt tot enkel de productcatalogus. Rekening houdend met de algemene trend rond personalisering van de customer experience zou het ook aangewezen zijn om ook de search op de e-commerce site te personaliseren. Dit zou resulteren in een betere experience voor de klant en een hogere conversie voor het bedrijf. In deze bachelorproef wordt onderzocht in hoeverre de persoonlijke data van een specifieke gebruiker en contextuele data kan ingeschakeld en gecombineerd worden met de data uit de productcatalogus, om zo persoonlijke en relevante zoekresultaten te genereren. Met deze persoonlijke data wordt bedoeld de historische aankoopdata van de persoon zelf, maar ook bepaalde demografische data zoals geslacht, leeftijd, gezinssamenstelling, etc. De onderzoeksvraag kan dus als volgt geformuleerd worden: Welk databankmodel presteert het best om gepersonaliseerde zoekresultaten te bekomen?

A.2 State-of-the-art

Personalized Search (Pitkow e.a., 2002) verwijst naar het zoeken op het web waarbij de resultaten afhankelijk zijn van de interesses en voorkeuren van de gebruiker die verder gaan dan de query zelf.

Personalisatie in e-commerce toepassingen biedt een groot voordeel aan bedrijven. De loyaliteit van klanten wordt veel sterker als deze gebruik maken van gepersonaliseerde features. (Telang & Mukhopadhyay, 2005) Een zoekfunctie is een voorbeeld van zo een feature. Recommender Systems (Resnick & Varian, 1997) zijn aanbevelingen vanuit het systeem die rekening houden met de beschikbare informatie van gebruikers en hun voorkeuren om zo een filter te plaatsen op de informatie die weergegeven wordt.

De studie van Diehl (2003) onderzocht het effect van gepersonaliseerde zoekresultaten op de kwaliteit van keuzes die klanten maken, en vond een positieve correlatie. De studie ontdekte dat het verlagen van search cost (Smith, Venkatraman & Dholakia, 1999) leidde tot minder kwaliteitsvolle keuzes. De reden daarvoor is dat klanten slechtere beslissingen maken als de search cost lager ligt omdat zij minder ideale opties aangeboden krijgen. Personalized Search en Recommender System zorgen voor een enorme verbetering in de kwaliteit van de keuzes die de klant maakt, en verminderen het aantal producten die deze klant bekijkt alvorens hij/zij gevonden heeft wat hij/zij nodig heeft.

Een gevolg van gepersonaliseerde zoekopdrachten is dat we een Filter Bubble (Pariser, 2011) creëren. Het verlaagt de kans dat nieuwe informatie gevonden wordt doordat de resultaten van een zoekopdracht partijdig zijn en eerder wijzen naar dingen die de gebruiker reeds gezien heeft. Dit concept wordt een Filter Bubble genoemd omdat gebruikers eigenlijk geïsoleerd worden in hun eigen wereldje, waar ze enkel de informatie te zien krijgen die ze willen zien. Als we deze gebruikers met hun bubbels in groepen opdelen, verkrijgen we wel het probleem dat deze een vertekend beeld op de realiteit krijgen, zij krijgen bijvoorbeeld in het nieuws slechts het deel te zien dat voor hun interessant is. Als we deze lijn doortrekken naar de klanten van e-commerce websites, zullen zij ook slechts de merken te zien krijgen waar hun voorkeur naar uit gaat. Hierdoor verminder je de kans dat ze een nieuw merk ontdekken of een ander product uitproberen. Als we terug refereren naar de studie van Diehl (2003), dan kunnen we afleiden dat dit een positief effect zal hebben op de klanttevredenheid.

A.3 Methodologie

Om de onderzoeksvraag te beantwoorden wordt er een simpele webapplicatie opgezet waar een zoekterm kan ingevoerd worden. Deze webapplicatie zal louter gebruikt worden om met behulp van een tekstveld een query op een databank uit te voeren. Ook worden er twee databankmodellen ontworpen, een model dat gebruik maakt van Neo4j (Graph databank platform), en een model dat gebruik maakt van Elasticsearch (OLAP databank platform) en Kibana om de resultaten te visualiseren. Graph is een databankstructuur van het type NoSQL, dit wil algemeen zeggen dat ze geen gebruik maken van SQL. Bij Graph databanken worden gegevens voorgesteld door een geheel van entiteiten en verbindingen, alsook vrije relaties tussen deze entiteiten. Kortom is dit, zoals de naam al laat vermoeden, een graaf. OLAP is de afkorting voor Online Analytical Processing, dit is een technologie die geoptimaliseerd is voor het uitvoeren van query's en rapporten in plaats van transacties. Deze zullen elk met hun eigen API communiceren, en in beide modellen wordt dezelfde gebruiker- en productdata ingevoerd. Bij het opstellen van de modellen wordt mogelijk

al duidelijk of één van de modellen niet in staat zal zijn om dezelfde functionaliteiten te hebben als het andere, en dan zal moeten afgewogen worden of de voordelen van het ene model opwegen tegenover het andere model. Als beide modellen dezelfde functionaliteit kunnen bereiken, wordt er bekeken welke het meest performante is. Mogelijks zou het ook haalbaar zijn om beide manieren te combineren op voorwaarde dat de responstijd binnen de acceptabele norm valt.

A.4 Verwachte resultaten

Er wordt verwacht dat er ofwel een duidelijk verschil merkbaar is in performantie tussen de twee verschillende modellen. Mogelijk is dat één van de twee modellen totaal niet haalbaar is om efficiënt een link mee te leggen tussen bijvoorbeeld familieleden, in dit geval bekijken we of het mogelijk is deze twee modellen samen uit te voeren, als dit een resultaat biedt dat binnen de norm valt qua performantie, dan is dit ook een mogelijke oplossing. Het kan zich ook voordoen dat beide modellen vrij performant en efficiënt de query's kunnen verwerken, in dit geval worden de voor- en nadelen alsook de moeilijkheidsgraad van implementatie afgewogen

A.5 Verwachte conclusies

Er wordt verwacht dat het Graph model makkelijker te implementeren zal zijn en beter zal presteren als we rekening willen houden met het aankoopgedrag van vrienden en familie. Indien dit ook mogelijk is bij een OLAP-model, verwachten we dat Graph nog steeds beter zal presteren. Indien we hier geen rekening mee houden zal het OLAP-model beter presteren, aangezien dit model aangepast is aan grote hoeveelheden data waar complexe zoekopdrachten op kunnen uitgevoerd worden.

Bibliografie

- Adomavicius, G. & Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6), 734–749.
- Çano, E. (2017). Hybrid Recommender Systems: A Systematic Literature Review. *Intelligent Data Analysis*, 21, 1487–1524. doi:10.3233/IDA-163209
- Dehkordi, G. J., Rezvani, S., Rahman, M. S., Nahid, F. F. N. & Jouya, S. F. (2012). A conceptual study on E-marketing and its operation on firm's promotion and understanding customer's response. *International Journal of Business and Management*, 7(19), 114.
- Diehl, K. (2003). Personalization and decision support tools: Effects on search and consumer decision making. *ADVANCES IN CONSUMER RESEARCH*, VOL 30, 30, 166–167.
- Linden, G., Smith, B. & York, J. (2003). Amazon.Com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, 7(1), 76–80. doi:10.1109/MIC.2003.1167344
- Lops, P., de Gemmis, M. & Semeraro, G. (2011). Content-based Recommender Systems: State of the Art and Trends. In *Recommender Systems Handbook* (pp. 73–105). doi:10.1007/978-0-387-85820-3_3
- NeoTechnology. (2014). Case Study: Walmart uses Neo4j to give customers best web experience through relevant and personal recommendations. Verkregen van http://dev.assets.neo4j.com.s3.amazonaws.com/wp-content/uploads/Neo4j_CS_Walmart.pdf?_ga=1.189104616.2108618949.1430736703
- Netflix. (2009). The Netflix Prize. Verkregen 29 maart 2020, van <https://www.netflixprize.com>
- Pariser, E. (2011). *The Filter Bubble: What the Internet Is Hiding from You*. Penguin Group, The.

- Pitkow, J., Schütze, H., Cass, T., Cooley, R., Turnbull, D., Edmonds, A., ... Breuel, T. (2002). Personalized search. *Communications of the ACM*, 45(9). doi:10.1145/567498.567526
- Resnick, P. & Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, 40(3), 56–58. doi:10.1145/245108.245121
- Sarwar, B., Karypis, G., Konstan, J. & Riedl, J. (2001). Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web* (pp. 285–295). WWW '01. doi:10.1145/371920.372071
- Schafer, J. B., Frankowski, D., Herlocker, J. & Sen, S. (g.d.). Collaborative Filtering Recommender Systems. In *The Adaptive Web* (pp. 291–324). doi:10.1007/978-3-540-72079-9_9
- Smith, G. E., Venkatraman, M. P. & Dholakia, R. R. (1999). Diagnosing the search cost effect: Waiting time and the moderating impact of prior category knowledge. *Journal of Economic Psychology*, 20(3), 285–314. doi:10.1016/s0167-4870(99)00010-0
- Telang, R. & Mukhopadhyay, T. (2005). Drivers of Web portal use. *Electronic Commerce Research and Applications*, 4(1), 49–65. doi:10.1016/j.elerap.2004.10.004
- Vavliakis, K. N., Katsikopoulos, G. & Symeonidis, A. L. (2019). E-commerce Personalization with Elasticsearch. *Procedia Computer Science*, 151, 1128–1133. The 10th International Conference on Ambient Systems, Networks and Technologies (ANT 2019) / The 2nd International Conference on Emerging Data and Industry 4.0 (EDI40 2019) / Affiliated Workshops. doi:https://doi.org/10.1016/j.procs.2019.04.160