Implementatieverslag Bachelorproject IN3405
Journal voor CHAINels

Opdrachtgever (CHAINels): Dr.Ir. S.I. Suddle
Begeleider (TU Delft): Dr. T. B. Klos
Coördinator (TU Delft): Phd. H.-G. Gross
Faculteit Elektrotechniek, Wiskunde en Informatica

Jeroen Mandersloot (4051874)

Benny Onrust (4019512) Laurent Verweijen (4030281)

8 juli 2012

Voorwoord

Op de TU Delft dient een bachelor afgesloten te worden door middel van een bachelorproject. Het doel van zo'n project is om te laten zien dat je met de opgedane kennis een compleet softwareproject uit kunt voeren. Wij hebben met zijn drieën dit project ingevuld door een project uit te voeren voor een startend bedrijf genaamd *CHAINels*.

In dit verslag bespreken wij wat we tijdens de uitvoering van het project precies gedaan hebben. Wij willen in ieder geval onze opdrachtgevers, Erwin Buckers en Vincent Koeman, bedanken voor hun steun tijdens dit project. Ook willen we Tomas Klos, onze begeleider namens de TU Delft, en Shahid Suddle, onze begeleider namens het bedrijf, bedanken.

Inhoudsopgave

1. Samenvatting

Wij hebben ons project uitgevoerd bij CHAINels, een professioneel bedrijf met als doel andere bedrijven in contact te brengen met elkaar. Via onze medestudenten Erwin en Vincent zijn wij met dit bedrijf in aanraking gekomen. CHAINels is een sociaal netwerk waar gebruikers als bedrijven aangemeld zijn en op verschillende manieren met andere bedrijven kunnen communiceren. De basis voor de website van CHAINels was er al voordat wij aan ons bachelorproject begonnen. Onze opdracht was het uitbreiden van deze website met een Journal, een pagina waar de meest relevante berichten uit het netwerk van een bedrijf bij elkaar worden weergegeven.

Het was onze taak om een algoritme te schrijven dat de meest relevante berichten voor een bedrijf kon bepalen uit de verzameling van alle berichten binnen het netwerk van het bedrijf. Ook moesten we deze berichten overzichtelijk weergeven binnen de website van CHAINels. We hebben in de oriëntatiefase van dit project al veel onderzoek gedaan naar bestaande algoritmes op dit gebied. Ook hebben we gezocht naar online kranten om een idee te krijgen van wat wel en niet werkt bij het ontwerpen van een online krant, want dat was in essentie wat de Journal moest gaan worden.

In de ontwerpfase van ons project hebben we ons algoritme helemaal uitgedacht en hebben we alle benodigde functionaliteiten van de Journal op een rij gezet. In overleg met de grafisch ontwerper binnen het bestaande team van CHAINels, Willem Buijs, zijn we tot de uiteindelijke lay-out voor de Journalpagina gekomen die naar ons idee de functionaliteiten van de Journal zo goed mogelijk naar voren brengt. Het idee was om zoveel mogelijk het gevoel van een gedrukte krant te benadrukken en dit is goed gelukt. Ook hadden we ons algoritme in detail uitgedacht en onderverdeeld in drie subalgoritmen: een content-based algoritme, een collaborative filtering algoritme en een timefilter. Al deze drie subalgoritmen zouden individueel een score toekennen aan een bericht, waarna deze score gecombineerd zou worden tot een uiteindelijke score die gebruikt kon worden om de berichten te sorteren op relevantie. Het content-based subalgoritme beoordeelde in hoeverre een bericht aansloot op de voorkeuren van een gebruiker op basis van het klikgedrag van de gebruiker in het verleden, het collaborative filtering subalgoritme zocht binnen CHAINels naar andere gebruikers van wie de voorkeuren overeenkomen met die van de ingelogde gebruiker en keek vervolgens in welke berichten zij interesse toonden (en kende aan deze berichten een hoge score toe) en het timefilter subalgoritme

beoordeelde hoe relevant een bericht was op basis van de leeftijd van het bericht.

Zo hadden we een goede basis voor de implementatie. We hebben de ideeën die we in de ontwerpfase hebben opgesteld vervolgens uitgewerkt tijdens de implementatiefase om het ook in praktijk werkend te krijgen. We hebben de Journal compleet geïntegreerd in de bestaande website van CHAINels zodat het gelijk in gebruik kon worden genomen. In de Journal krijgt men nu in de hoofdpagina een verzameling van berichten te zien die, verspreid over twee kolommen, in volgorde van relevantie aan de gebruiker getoond worden. De Journal wordt standaard geopend op de zogenaamde hoofdpagina; de pagina waar alle soorten berichten door elkaar staan. De Journal maakt hiervoor onderscheid tussen vier soorten berichten: nieuws-, vraag-, aanbod- en evenementberichten. Voor elk van deze soorten berichten (behalve vraag- en aanbodberichten, die vallen samen) is er ook een aparte pagina binnen de Journal waarin alleen berichten van de geselecteerde soort worden geladen.

Uiteraard kwamen bij het ontwikkelen van het algoritme voor de Journal problemen naar boven zoals de snelheid van het algoritme. Omdat de berekeningen die gedaan moeten worden behoorlijk zwaar zijn, heeft het algoritme toch wel een aantal seconden nodig om compleet uitgevoerd te worden. Om te voorkomen dat dit elke keer opnieuw gedaan moest worden, hebben we gebruik gemaakt van cookies. We slaan het resultaat van het algoritme op in een cookie, zodat we de volgende keer dat de Journal geladen moet worden alle benodigde informatie direct uit de cookie kunnen lezen. Op deze manier besparen we ontzettend veel tijd en wordt het een stuk fijner om de Journal te gebruiken.

Uiteindelijk hebben we zowel de back-end (het algoritme) als de front-end (de pagina voor de Journal) tot onze tevredenheid weten te implementeren. We vinden dat de Journal zoals hij nu is goed werkt en doet wat hij hoort te doen. We zijn trots op ons eindresultaat.

2. Inleiding

CHAINels is een netwerk dat zich richt op business-to-business-communicatie. Dit wil zeggen dat een gebruiker handelt in de naam van het bedrijf; alle communicatie naar buiten wordt gezien als afkomstig van een bedrijf en niet van een individu. Er kunnen wel meerdere accounts gekoppeld zijn aan één bedrijf, maar als een ingelogde gebruiker bijvoorbeeld een bericht plaatst, gebeurt dit in naam van het bedrijf waar hij of zij werkzaam is.

Zo kunnen bedrijven elkaar op de hoogte houden van nieuws en evenementen. In tegenstelling tot Facebook en LinkedIn is CHAINels geen netwerk tussen personen, maar tussen bedrijven. CHAINels is veel meer op bedrijven gericht en heeft hierdoor een zakelijker karakter dan persoon-tot-persoon netwerken. Het doel van CHAINels is om bedrijven op een professionele manier in staat te stellen om zakelijke relaties met elkaar te onderhouden. De naam "CHAINels" is een woordspeling op de woorden "channels" en "chain", waar "channels" het karakter van CHAINels als communicatiekanaal benadrukt. Het woord "chain" wordt binnen dit bedrijf gebruikt om een connectie tussen twee bedrijven aan te duiden, waarbij ze van elkaars activiteiten op CHAINels op de hoogte worden gehouden.

Wij kwamen met dit project in aanraking door onze medestudenten Erwin Buckers en Vincent Koeman die zelf verantwoordelijk zijn voor de ICT van dit bedrijf. Zij hebben hun bachelorproject ook bij CHAINels gedaan en zijn daarna bij het bedrijf gebleven.

CHAINels is een uitgebreid netwerk met een hoop functionaliteiten. Ons bachelorproject was specifiek gericht op de Journal van CHAINels . De Journal moest kort gezegd een pagina worden waar de meest interessante berichten uit het netwerk van een bedrijf bij elkaar geplaatst werden, zodat een bedrijf gemakkelijk het laatste nieuws kon lezen. Hiervoor hebben we zowel aan de algoritmes moeten werken die de belangrijke berichten onderscheiden van de minder belangrijke als aan de vormgeving en structuur van deze pagina binnen de rest van de website.

3. Probleemstelling en analyse

In CHAINels heeft elk bedrijf een persoonlijke pagina met berichten. Wanneer een gebruiker op de pagina van een ander bedrijf klikt, krijgt de gebruiker de berichten te zien die door dat bedrijf geplaatst zijn. Omdat het tijdrovend is om de pagina's van al je chains af te gaan, moet er een aparte pagina komen waarop de belangrijkste recente updates van de mensen in je netwerk overzichtelijk worden weergegeven. Het doel van ons project is dan ook het programmeren van deze pagina, oftewel de Journal.

Het algoritme dat gaat bepalen welke berichten relevant voor een bedrijf zijn, moet uiteraard accuraat zijn, maar het moet ook snel blijven werken als er een grote hoeveelheid bedrijven gebruik gaat maken van CHAINels . Het is daarom ook van groot belang dat ons algoritme efficiënt is, zodat alles soepel blijft gaan als CHAINels eenmaal uitgegroeid is tot een groot bedrijf.

We hebben te maken met bedrijven uit verschillende industrieën. Er zijn op dit moment ongeveer twintig voorgedefinieerde industrieën waar een bedrijf toe kan behoren. Voorbeelden hiervan zijn ICT, zorg en bouw. Een bedrijf kan zijn industrie ook nader specificeren, maar behoort altijd tot precies ëën hoofdindustrie.

In de Journal krijgen we te maken met verschillende soorten berichten. Zo hebben we nieuws-, evenement-, vraag-, aanbod- en informatieberichten. Het is niet de bedoeling dat informatieberichten mee worden genomen in het algoritme. Deze verschillende soorten berichten kunnen onderling op sommige punten verschillen. Nieuwsberichten zijn berichten die bedrijven kunnen plaatsen om informatie naar buiten te brengen, evenementberichten zijn bedoeld om andere bedrijven op de hoogte te stellen van een aankomend evenement, vraag- en aanbodberichten zijn een soort advertenties die bedrijven kunnen plaatsen als zij ergens naar op zoek zijn of juist iets aan te bieden hebben en informatieberichten zijn automatisch gegenereerde berichten over de activiteit van bedrijven op het netwerk (bijvoorbeeld als twee bedrijven een chain met elkaar aangaan).

Voor de gebruikers moet de Journal overzichtelijk zijn. De gebruikers moeten verschillende berichttypes duidelijk van elkaar kunnen onderscheiden en moeten gemakkelijk met de Journal kunnen werken. Naast het vinden van de relevante berichten moeten we dus ook nadenken over de weergave hiervan.

4. Ontwerp

In dit hoofdstuk zullen we uitleggen hoe het ontwerp van de Journal eruit ziet. Het ontwerp bestaat uit twee delen: de front-end en de back-end. Eerst zullen we de front-end bespreken, waar we aandacht besteden aan de lay-out van de Journal en de verschillende functionaliteiten die de Journal de gebruiker moet gaan bieden. Vervolgens sluiten we af met het ontwerp van de de back-end, waar we het voornamelijk zullen hebben over het aanbevelingsalgoritme en de verschillende subalgoritmes waaruit het is opgebouwd.

4.1 Front-end

Alhoewel ons project voornamelijk gericht was op het ontwikkelen van algoritmes die gepaste berichten konden vinden om in de Journal te plaatsen, was het natuurlijk ook zaak dat die berichten uiteindelijk overzichtelijk werden weergegeven. Daarnaast moest ook nagedacht worden over hoe de gebruiker om diende te gaan met de Journal - op welke manier zou de gebruiker interacteren met de Journal? Er moest dus een ontwerp voor de lay-out komen waarbij rekening werd gehouden met de functionaliteiten van de Journal voor de gebruiker. Binnen het CHAINels-team is Willem Buijs verantwoordelijk voor de grafische aspecten van de website en in overleg met ons zou hij een ontwerp voor de lay-out maken. De functionaliteit van de Journal werd volledig aan ons overgelaten.

In deze sectie zullen we bespreken hoe we tot het ontwerp van de uiteindelijke Journal - zowel grafisch als functioneel - zijn gekomen, welke afwegingen we daarbij hebben gemaakt en waarom we gekozen hebben voor de oplossingen die nu in de voltooide Journal terug te vinden zijn.

4.1.1 De basis

Vanaf het begin van dit project zijn we bezig geweest met het ontwerp van de front-end van de Journal, met name de lay-out hiervan. Uit gesprekken met Vincent en Erwin bleek al dat er qua lay-out nog een hoop onzekerheid was, waardoor onze eigen inbreng op dit gebied ook werd gewaardeerd. Er waren wel wat ideeën binnen het CHAINels-team, maar de precieze uitwerking hiervan lag nog open.

In de oriëntatiefase hadden we al snel een gesprek met Erwin waarin we ook kennis maakten met Willem. Dit gesprek was bedoeld om voor iedereen een wat duidelijker beeld te schetsen van wat de Journal precies in moest gaan houden. Na afloop was uiteraard nog niet alles helemaal uitgedacht, maar hadden we in ieder geval een beter idee welke kant we op moesten gaan. Het belangrijkste wat in dit gesprek naar voren kwam, was dat de Journal zo veel mogelijk het uiterlijk en het gevoel van een gedrukte krant moest nabootsen, zodat mensen de Journal ook meer als zodanig zouden beschouwen. De reden dat we graag wilden dat de Journal zo gezien zou worden, was dat een gedrukte krant toch iets zakelijks heeft en dat is waar het bij CHAINels het meest om draait. Met dit in ons achterhoofd zijn we op zoek gegaan naar websites van kranten die volgens ons aan dit criterium voldeden om enkele ideeën op te doen die toepasselijk zouden kunnen zijn voor onze eigen Journal.

Een ander punt dat aan de hand van dit gesprek naar voren kwam, was dat de Journal geopend moest worden door te klikken op een verticale balk of een knop aan de linkerkant van het scherm. De Journal zou dan vanaf links uit moeten schuiven en een nog nader te bepalen deel van het scherm vullen. Hiermee zou de Journal dus de inhoud van de pagina die op dat moment geopend was gedeeltelijk overlappen, wat goed aansloot op ons idee om de Journal zo veel mogelijk als een gedrukte krant te zien: het bootst een beetje het effect van het openslaan van een krant na.

Ook werd vastgesteld dat de Journal altijd op een vaste positie op het scherm moest blijven. Dat wil zeggen dat de Journal niet mee mocht bewegen wanneer een gebruiker naar boven of naar beneden zou scrollen op de pagina die achter de Journal verscholen zat. Op deze manier zou de Journal altijd op elke pagina te openen zijn, waar men zich ook zou bevinden op die pagina.

4.1.2 De details

In de twee weken na de oriëntatiefase was het van belang om het basisidee wat we nu van de Journal hadden gedetailleerder uit te werken zodat we genoeg hadden om het te implementeren. Een aantal zaken hadden we al aardig duidelijk in ons hoofd zitten tegen deze tijd, maar problemen zoals hoe we de verschillende berichten in de Journal weer zouden gaan geven en hoe we om zouden gaan met grote hoeveelheden berichten hadden we nog niet opgelost.

Hierom leek het ons een goed idee om nog een keer met Willem af te spreken en samen met hem te brainstormen over goede oplossingen. Het uiteindelijke gesprek leverde zelf niet gelijk heel veel op, maar we hebben wel naar elkaar duidelijk kunnen maken wat we van de ander nodig hadden om vooruitgang te boeken. Willem wilde graag van ons horen welke functionaliteiten de Journal moest gaan bieden, zodat hij een beter idee kon krijgen van wat de Journal precies moest kunnen. Dit zou hem ontzettend helpen met het ontwerpen van een lay-out, omdat de vormgeving van de Journal wel aan moet sluiten bij de functionaliteiten ervan.

De paar dagen daarna hebben we binnen onze groep en ook eenmaal met Erwin erbij druk overlegd wat we precies wilden en of we zelf een voorkeur hadden

voor een bepaalde manier waarop dit in de lay-out verwerkt zou kunnen worden. Uiteindelijk kwamen we op de volgende beschrijvingen en functionaliteiten uit:

- Links van het scherm staat een verticale balk waar de gebruiker op kan klikken. Zodra er op deze balk geklikt wordt, schuift de Journal uit naar rechts en wordt de inhoud geladen. De Journal zal ongeveer de breedte van het gehele scherm in beslag nemen; het lijkt ons wel mooi als er een klein beetje ruimte over wordt gelaten aan de rechterkant om de onderliggende pagina nog te kunnen zien.

- Als de Journal eenmaal is uitgeschoven en de inhoud is geladen, ziet de gebruiker een verzameling van berichten die volgens ons algoritme relevant zijn. De verschillende soorten berichten die we onderscheiden zijn:

  - Nieuwsberichten
  - Vraag- en aanbodberichten
  - Eventberichten
  - Infoberichten

- Er zijn verschillende manieren waarop we deze berichten kunnen sorteren om een aantrekkelijke Journal te maken. Ons idee is om een kolommenstructuur te gebruiken en daar de berichten in te plaatsen. We willen drie kolommen aanhouden, waarvan de middelste het breedst is (zeg ongeveer even breed als de linker- en rechterkolom samen). In deze kolom willen we op de hoofdpagina van de Journal een hoofdartikel plaatsen om onmiddellijk de aandacht van de gebruiker te trekken. Slechts een beperkt aantal berichten komt in aanmerking voor deze positie; er hoort bijvoorbeeld sowieso een afbeelding bij zo'n hoofdartikel die ook breed genoeg moet zijn om de hele kolom in de breedte te vullen. Deze invulling lijkt ons het best omdat de meeste mensen dit zullen herkennen uit gedrukte kranten en dus intuïtief weten wat ongeveer waar zal staan.

- De rechterkolom zouden we willen gebruiken voor externe berichten, berichten van buiten je eigen netwerk. Dit zijn bijvoorbeeld gesponsorde berichten of vraag- en aanbodberichten afkomstig van bedrijven buiten je eigen netwerk. We willen een gesponsord bericht in de rechterbovenhoek plaatsen en de overige berichten in die kolom daaronder. Helemaal rechtsonder kunnen we kort een aantal infoberichten plaatsen. De linkerkolom en de rest van de middelste kolom vullen we op met nieuws- en eventberichten.

- We raden aan berichten onderling te scheiden door middel van een horizontale streep in plaats van een blokkenstructuur waarin elk bericht een apart, zwevend blok is, omdat een blokkenstructuur totaal niet overeenkomt met hoe men normaal berichten in een krant leest. Daar is er een achtergrondkleur die achter alle berichten te zien is zonder dat de berichten van elkaar gescheiden worden door randen om de berichten, terwijl bij

een blokkenstructuur alle berichten juist heel duidelijk op die manier van elkaar zijn gescheiden.

- Als een bericht een afbeelding bevat, lijkt het ons mooi om die afbeelding (eventueel verkleind) in de Journal weer te geven boven het bericht zelf, omdat afbeeldingen toch altijd de aandacht van gebruikers trekken. Het leek ons het best om berichten met en zonder afbeeldingen af te wisselen, zodat het geheel er dynamisch uitziet.

- Om het gevoel van een gedrukte krant na te bootsen, willen we ook gebruik maken van verschillende pagina's binnen de Journal om het idee van een pagina omslaan erin te verwerken. Binnen een pagina kan een gebruiker wel iets omlaag scrollen, mocht zijn of haar beeldscherm niet groot genoeg zijn om de hele pagina er in een keer op weer te geven, maar het moet niet zo zijn dat je net als in de News Feed van Facebook oneindig lang door kan scrollen en dat er steeds nieuwe berichten blijven verschijnen.

- Het hoofdartikel verschijnt alleen op de eerste pagina. Vanaf de tweede pagina wordt die ruimte opgevuld met nieuws- en eventberichten, net als de rest van de middelste kolom.

- We willen aan de bovenkant van elke de naam van het bedrijf van de gebruiker tonen. Ook komen hier drie filterknoppen, namelijk voor nieuws-, event-, vraag- en aanbodberichten. Door op een van deze knoppen te drukken, wordt er een nieuwe Journal gegenereerd die alleen bestaat uit berichten uit de categorie die de gebruiker heeft aangeklikt. Elk van deze gefilterde Journals moet zijn eigen ontwerp krijgen, vergelijkbaar met hoe je in een gedrukte krant bijvoorbeeld een aparte sectie voor Sport hebt. De opzet van deze gefilterde Journals is hetzelfde als die van de gecombineerde Journal in de zin dat we nog steeds drie kolommen willen gebruiken met een hoofdartikel in de middelste kolom, etc. Het grote verschil is dat we nu in de banner (en wellicht ook in de rest van de pagina's) duidelijk willen aangeven dat het een gefilterde Journal is. Dit kan bijvoorbeeld door de banner een andere achtergrondkleur te geven of in ieder geval door het woord "Nieuws"groot in de banner te zetten in het geval van een gefilterde Journal voor nieuwsberichten.

De bovengenoemde punten hebben we naar Willem gestuurd zodat hij op basis hiervan een ontwerp voor de lay-out kon maken en een aantal dagen later kregen we dit ontwerp aangeleverd.

4.1.3 Het uiteindelijke ontwerp

Voor het grootste gedeelte kwam dit ontwerp (Figuur 4.1.3) overeen met de punten die wij naar Willem hadden gestuurd. Er was een duidelijke kolommenstructuur met drie kolommen, er was rekening gehouden met meerdere pagina's binnen de Journal en de bovenkant van de Journal was precies zoals wij het ons

Figuur 4.1: Willems ontwerp van de hoofdpagina van de Journal.

hadden voorgesteld. Er waren echter ook wat verschillen. Zo nam het hoofdartikel de gehele breedte van de Journal in beslag, werd er nog wel enigszins gebruik gemaakt van een blokkenstructuur en waren de linker- en rechterkolom niet even lang (de rechterkolom was iets korter gemaakt; de middelste kolom was nu even lang als de linkerkolom).

Ook de verdeling van de berichten was enigszins veranderd. Zo bevatte de rechterkolom geen berichten van buiten je eigen netwerk, maar werd het in de hoofdpagina volledig opgevuld met informatieberichten. In de afzonderlijke pagina's voor berichten uit een bepaalde categorie werd deze rechterkolom opgevuld met berichten waarvan de gebruiker heeft aangegeven ze te willen volgen. Op deze manier zou de Journal nog beter geschikt zijn om de gebruiker op de hoogte te houden van interessante ontwikkelingen. Dit leek ons naderhand ook een beter idee, omdat het ook voor interactiemogelijkheden zorgde tussen de gebruiker en de Journal. Hierdoor kwamen we namelijk op het idee om gebruikers berichten uit de Journal te laten slepen naar de rechterkolom om hen zo makkelijk een bericht wat hen interessant leek te kunnen laten volgen. Dit zou de Journal van CHAINels ook wat extra's geven in vergelijking met soortgelijke functionaliteiten op andere websites.

Iets anders dat opviel, was dat er in dit design geen ruimte meer was voor externe of gesponsorde berichten. De externe berichten hebben we uiteindelijk achterwege gelaten, omdat die een te grote last op het algoritme zouden plaatsen. De gesponsorde berichten zijn weg gelaten omdat het systeem daarachter nog niet in het systeem geïmplementeerd was.

Ondanks het kleine aantal verschillen tussen dit uiteindelijke ontwerp en onze eigen visie van de Journal konden we ons toch goed vinden in dit uiteindelijke ontwerp. Het belangrijkste vonden wij dat de Journal nog wel voelde als een gedrukte krant en met dit ontwerp was dat gevoel zeker behouden. Daarnaast was het ontwerp erg duidelijk en makkelijk te begrijpen voor gebruikers: de icoontjes linksboven in elk bericht geven aan wat voor soort bericht het was (een nieuws-, vraag-, aanbod- of eventbericht) en aan het bedrijfslogo rechtsboven kan men zien door welk bedrijf het bericht geplaatst was. Onderaan elk bericht staan ook twee iconen - een envelop en een ster, respectievelijk bedoeld om op een bericht te reageren of een bericht te volgen. Deze iconen staan namelijk ook bij de berichten zoals ze worden weergegeven op de profielpagina's van bedrijven, dus op deze manier blijft deze weergave consistent binnen het systeem.

We besloten echter in nader overleg met Erwin om de Journal toch nog op een punt te wijzigen: het leek ons uiteindelijk toch geen goed idee om met pagina's te werken. In plaats daarvan wilden we van de Journal een lange pagina maken waar de gebruiker naar beneden kon scrollen om steeds meer berichten te zien. Onze reden hiervoor was dat, ondanks het feit dat een navigatie met behulp van pagina's een betere benadering zou zijn van een gedrukte krant, we vermoedden dat mensen op het internet toch liever zouden scrollen omdat daarbij niet telkens een nieuwe pagina geladen hoeft te worden.

4.2 Aanbevelingsalgoritme

Het aanbevelingsalgoritme bestaat uit verschillende subalgoritmes. Om een duidelijk overzicht te geven zullen we deze subalgoritmen één voor één bespreken. Eerst zullen we echter beginnen met uit te leggen hoe een gebruiker een bericht kan 'raten' en geven we een overzicht van de notaties die we in de rest van deze sectie gebruiken. Daarna wordt het eerste subalgoritme het 'content-based' algoritme uitgelegd. Dit is een algoritme dat kijkt naar de content van het bericht waar de gebruiker een review op heeft geplaatst. Daarna behandelen we het collaborative filtering algoritme, een algoritme dat kijkt naar gelijksoortige bedrijven en berichten. Het laatste subalgoritme is het tijdsfilter, wat rekening houdt met wanneer een bericht is geplaatst en op basis daarvan een score aan het bericht toekent. Tenslotte leggen we uit hoe we deze subalgoritmes combineren om uiteindelijk tot een definitieve waardering van een bericht te komen.

4.2.1 Ratings

Om aanbevelingen te maken voor een bepaalde gebruiker moet bekend zijn wat hij of zij vindt van bepaalde berichten. In veel aanbevelingssystemen is het mogelijk dat een gebruiker voor bepaalde items een score kan geven tussen 1 en 5 [1], waarbij 1 een zeer negatieve beoordeling is en 5 een zeer positieve. In het systeem van CHAINels is het echter niet mogelijk om direct zulke scores aan een bericht toe te kennen. Het is echter wel mogelijk om op een bericht een reactie te plaatsen, een bericht te volgen of op een bericht te klikken. Deze informatie kunnen we ook worden vertalen naar een rating voor een bericht [2]. Voor de duidelijkheid volgt nu een overzicht van deze acties en wat voor weging we aan elke actie toekennen:

- Een gebruiker klikt op een bericht in de Journal. Er verschijnt dan een pop-up van het bericht waar de reacties te zien op het bericht. Bovendien is het dan mogelijk om zelf een reactie plaatsen. De waarde 1 wordt toegekend voor een klik op het bericht.
- Een gebruiker volgt een bericht in de Journal. Dit doet de gebruiker door op het sterretje rechtsonder van het bericht te klikken. Voor deze actie geven wij de waarde 2 aan het bericht.
- Een gebruiker plaatst een reactie op een bericht in de Journal. We kennen de waarde 3 toe aan het bericht.

De reden dat de ene actie een hogere waarde oplevert dan de andere heeft te maken met het feit dat een gebruiker voor een reactie meer moeite moet doen dan voor een klik op een bericht. Ditzelfde geldt voor het volgen van een bericht. Wij gaan uit van de gedachte dat hoe meer tijd je in een bericht steekt, hoe interessanter je het bericht vindt. Daarom krijgt de ene actie dus een hogere score dan de andere. Belangrijk om te vermelden is dat we alleen de hoogste actie per bericht per gebruiker laten meetellen. Dus wanneer je klikt en reageert

op een bericht, dan telt in dit geval alleen de reactie op het bericht mee. Ook als je meerdere reacties achter laat op een bericht, dan telt dit maar als één reactie mee. Verder is het niet mogelijk om een blijk van afkeuring aan een bericht te geven. Dus bovenstaande manieren zien wij allemaal als positieve feedback op een bericht, waarvan de één zwaarder meetelt dan de andere. Wanneer wij het in de rest van dit verslag hebben over een gebruiker die een rating aan of een review van een bericht geeft, bedoelen wij dus dat hij op het bericht klikt, het bericht volgt of een reactie op het bericht achterlaat. De notatie hiervoor is r_{u_b,i_a} waarbij r de rating is van gebruiker u_b op bericht i_a.

Voordat we beginnen met het daadwerkelijk ontwerp van de subalgoritmes, moeten we ook eerst vermelden dat CHAINels natuurlijk wilt dat het algoritme efficiënt werkt, maar het wilt ook zo min mogelijk data in de database opslaan voor het algoritme. Dit heeft als gevolg dat we eigenlijk geen 'pre-processing' kunnen uitvoeren voor ons subalgoritmen. Dit betekent dat bijna alle benodigde informatie voor de subalgoritmes berekend moet worden op het moment dat het aanbevelingsalgoritme wordt aangeroepen. Dus zullen de subalgoritme in ieder geval langzamer werken. Wanneer er wel 'pre-processing' wordt gebruikt, dan worden er allerlei tijdrovende processen van het aanbevelingsalgoritme berekend en opgeslagen in de database voordat het aanbevelingsalgoritme een aanroep heeft gehad. Dit zorgt uiteraard ervoor dat het algoritme sneller werkt.

4.2.2 Content-based

In deze sectie zullen we het content-based subalgoritme bespreken. Het doel van het content-based subalgoritme is om de interesses van de gebruiker in de verschillende bedrijfsindustriën te berekenen. Aan de hand van deze interesses kan dan makkelijk worden nagegaan of een bericht wel of niet interessant is voor de gebruiker. Voordat we gaan uitleggen hoe we precies de gebruikersinteresse bepalen en hoe we weten in welke bedrijfsindustrie een bepaald bericht hoort, gaan we eerst bekijken welke bekende problemen er zijn met content-based methodes. Tijdens en na het uitleggen van het ontwerp zullen we hier nog op terug komen en bespreken we in hoeverre ons algoritme van deze problemen last heeft. Tenslotte zullen we het hebben over het ontwerpproces en vertellen hoe we uiteindelijk bij ons huidige ontwerp zijn gekomen.

Bekende problemen

Content-based methodes hebben meestal last van de volgende problemen [11] [2] [1]:

- Gelimiteerde content analyse: content-based methodes zijn gelimiteerd door de eigenschappen die verbonden zijn aan de objecten die het systeem aanbeveelt. Om een goede set eigenschappen te krijgen van een bericht moeten die ofwel per bericht aanwezig zijn of de gebruiker moet zelf goed zijn profiel hebben ingevuld of bepaalde eigenschappen bij een bericht hebben geplaatst.

- Overspecialisatie: dit probleem treedt op wanneer systemen alleen items aanbevelen als die hoog scoren bij het gebruikersprofiel. Dus stel dat een gebruiker alleen berichten leerst uit de economiesector, dan zal het systeem deze gebruiker geen enkel sportbericht aanbevelen, zelfs niet wanneer er bijvoorbeeld iets heel bijzonders gebeurt in de sport.

- Nieuwe gebruiker: nieuwe gebruikers hebben meestal weinig tot geen reviews gedaan, dus kan er niet accuraat bepaald worden wat interessant is voor de gebruiker. De aanbevelingen zullen dan vanzelfsprekend van mindere kwaliteit zijn.

We zullen nu eerst uitleggen hoe ons content-based algoritme precies werkt voordat we gaan bespreken in hoeverre ons algoritme last heeft van de bovengenoemde problemen.

Bedrijfsindustrie van een bericht

Een belangrijke eigenschap van een bericht is de bedrijfsindustrie waartoe het bericht behoort. Aangezien we de gebruikersinteresse berekenen per bedrijfsindustrie (in de volgende subsectie meer hierover) kan wanneer deze eigenschap bekend is heel makkelijk bepaald worden hoe interessant een bericht voor de gebruiker is.

Er zijn in ons geval twee mogelijkheden om de bedrijfsindustrie van een bericht te bepalen: we stellen de bedrijfsindustrie van een bericht gelijk aan de industrie van het bedrijf dat het geplaatst heeft of we passen tekstanalyse[10] [9] [7] [3] toe om automatisch te herkennen bij welke bedrijfsindustrie het bericht hoort.

Wij hebben gekozen voor de eerste methode vanwege een aantal redenen. Ten eerste is de door ons gekozen methode een stuk efficiënter en hoeft er veel minder data opgeslagen te worden, aangezien je bij tekstanalyse alle steekwoorden moet identificeren en bepalen in hoeverre ieder steekwoord bij een bepaalde bedrijfsindustrie hoort. Onze methode kan in constante tijd de bedrijfsindustrie achterhalen. Wel moet gezegd worden dat onze methode minder accuraat is, maar we moesten een afweging maken tussen accuratie aan de ene kant en snelheid en opslag aan de andere. Gezien het feit dat CHAINels zelf ook zo min mogelijk data wil opslaan, kozen we dus voor snelheid en minder dataopslag ten koste van accuratie.

De tweede reden heeft te maken dat er niet altijd tekst aanwezig hoeft te zijn in het bericht. Het is mogelijk om alleen een afbeelding in het bericht te zetten met als gevolg dat er geen of nauwelijks tekstanalyse kan worden uitgevoerd. Het is echter altijd bekend tot welke industrie een bedrijf behoort, dus heb je ook geen last van het gelimiteerde content analyse probleem.

Gebruikersinteresse

Nu we weten hoe we de bedrijfsindustire van een bericht kunnen bepalen, kunnen we de gebruikersinteresses berekenen voor elke bedrijfsindustrie. De gebruikers-

interesse berekenen we door gebruik te maken van een naïef bayesiaans netwerk [8][5]. Dit bayesiaans netwerk is gebasserd op de papers van J. Liu [8], die een content-based methode heeft ontwikkeld voor de Google News site, en J. He [5], die een speciaal social network-based recommender system (SNRS) heeft ontwikkeld.

Voordat de gebruikersinteresse berekend kan worden, moet er eerst achterhaald worden wat de reviewverdeling is van de gebruiker over de verschillende bedrijfsindustriën. Dat doen we op de volgende manier:

$$D(u_a) = (r_1/r_{total}, R_2/R_{total}, ..., R_j/R_{total}) \tag{4.1}$$

Hier is u_a de gebruiker, r_j de totale rating geplaatst op berichten van bedrijfsindustrie j en R_{total} het totaal aantal reviews geplaatst op berichten over alle bedrijfsindustriën. Hierbij telt een klik als 1, volgen als 2 en een reactie als 3. Dus stel je hebt alleen een reactie en een klik geplaatst in bedrijfsindustrie 'Sport' en voor de rest niet in andere bedrijfsindustrën, dan is de reviewverdeling voor de bedrijfsindustrie 'Sport': $(3 + 1)/(1 + 1)$.

De reviewverdeling is op zichzelf geen goede graadmeter van de gebruikersinteresse. De reden hiervoor is dat in ons geval binnen het netwerk van een gebruiker heel veel berichten van dezelfde bedrijfsindustrie geplaatst kunnen worden en heel weinig van een andere. Hierdoor zal je waarschijnlijk meer reviews maken in de bedrijfsindustriën waar veel berichten van worden geplaatst dan in de bedrijfsindustriën waar bijna niets van wordt geplaatst, terwijl je misschien die berichten die weinig worden geplaatst juist interessant vindt. Deze zouden dan geen hoge score krijgen van het algoritme, omdat er bijna geen reviews geplaatst zijn. Daarom berekenen we de gebruikersinteresse op de volgende manier:

$$interesse(industrie) = \frac{reviewverdeling(industrie) + 1}{berichtverdeling(industrie) + n} \tag{4.2}$$

We delen dus de reviewverdeling voor een bepaalde industrie door het aantal gepubliceerde berichten van je netwerk voor die bepaalde industrie. Dit heeft als gevolg dat wanneer in een industrie weinig berichten worden geplaatst, maar wel consequent reviews krijgen, ze toch een hoge score krijgen. Verder is er ook nog een 'smoothing' factor aanwezig, zoals de $+1$ en de n. Deze n staat voor het totaal aantal bedijrfsindustriën en deze factor zorgt er voor dat erbij weinig ratings toch nog een resultaat eruit komt. Dit wordt ook gebruikt in het systeem van J. He [5].

Het content-based algoritme is nu bijna af, maar het mist nog één ding. In de paper van J.Liu [8] werd aangegeven dat de gebruikersinteresse met de tijd kan veranderen en ook daarmee wordt rekening gehouden in het algoritme. Daarom ziet ons uiteindelijke algoritme er als volgt uit:

$$interesse_{tot}(ind) = (1 - labda) \times interesse_{rec}(ind) + labda \times interesse_{alg}(ind) \tag{4.3}$$

Hierbij is ind een bedrijfsindustrie, $interesse_{rec}$ de interesse voor een bepaalde industrie over de afgelopen maand en $interesse_{alg}$ is de interesse van

sinds dat die zich voor CHAINels heeft aangemeld. De *lambda* is een gewicht om de twee interesses samen te voegen.

Evaluatie

Nu zullen kort even de bekende problemen voor content-based methodes langs gaan en kijken in hoeverre ons algoritme daar last van heeft.

- Gelimiteerde content analyse: Zoals al eerder aangeven is dit probleem verholpen doordat we de bedrijfsindustrie van een bericht bepalen door de bedrijfsindustrie te pakken van het bedrijf dat het bericht geplaatst heeft. De industrie van een bedrijf is altijd bekend, omdat die verplicht moet worden ingevuld wanneer je een account gaat aanmaken.
- Overspecialisatie: Van dit probleem is ook bijna geen sprake, want door een 'smoothing' factor te gebruiken en rekening te houden met het aantal berichten per industrie die zijn gepubliceerd, kan het zo zijn dat wanneer je in een bepaalde industrie weinig reviews hebt gedaan die industrie dan toch als relevant wordt gezien.
- Nieuwe gebruiker: Dit probleem wordt niet echt verholpen door het algoritme zelf, maar meestal heeft een nieuwe gebruiker nog bijna geen chains en zal hij of zij dus sowieso weinig berichten hebben in de Journal. Wanneer er nog weinig berichten in de Journal te zien zijn, speelt het aanbevelingsalgoritme nog niet z'n belangrijke rol.

Ontwerpproces

Het ontwerpproces ging redelijk goed. Nadat we hadden besloten om geen tekstanalyse te doen, kwamen we al snel op het naïve bayesiaans netwerk. In het begin hebben we vooral het algoritme gevolgd dat werd beschreven in de paper van J. Liu [8], maar dat algoritme was niet helemaal geschikt voor ons systeem, omdat wij te maken hebben met een sociaal netwerk en hun algoritme is bedoeld voor een nieuwssite. Wij delen nu bijvoorbeeld de reviewverdeling door het totaal aantal gepubliceerde berichten, terwijl hun algoritme de reviewverdeling deelde door de reviewvedeling van de gebruikers uit hetzelfde land. Dit werkte niet voor ons, omdat je alleen berichten kan zien van binnen je netwerk en elke gebruiker heeft dus een eigen verzameling berichten die hij kan zien, terwijl bij de nieuwssite iedereen dezelfde berichten kan zien. Hierom vonden wij het niet relevant om de reviewverdeling van de andere gebruikers te gebruiken, dus hebben wij vervolgens aanpassingen doorgevoerd. Wel hebben we de belangrijkste ideeën van hun algoritme gebruikt, zoals de weging van recente berichten. Ook hebben we uiteindelijk de 'smoothing' factor gebruikt van J. He [5].

4.2.3 Collaborative Filter

Het tweede subalgoritme dat gebruikt wordt is collaborative filtering. Het doel van dit collaborative filtering algoritme is het vinden van gelijksoortige gebrui-

kers en gelijksoortige berichten om daarmee een score te bepalen voor een bericht. In deze sectie zullen we eerst de bekende problemen van collaborative filter bespreken. Vervolgens leggen we uit welke vergelijkingsformule we gebruiken om de gelijkheid te berekenen tussen twee gebruikers of berichten. Daarna kijken we hoe we gelijksoortige gebruikers kunnen vinden en hoe we op basis van die gebruikers een score kunnen geven aan een bericht. Daarna bespreken we dit voor gelijksoortige berichten. Vervolgens kijken we hoe we de twee scores op basis van gelijksoortige gebruikers en gelijksoortige berichten kunnen combineren en introduceren we ook een 'smoothing' factor. Daarna vervolgt nog een korte evaluatie over het algoritme en kijken we in hoeverre de bekende problemen wel of niet worden aangepakt. Ten slotte zullen we nog een blik werpen op het ontwerpproces van het collaborative filter subalgoritme.

Bekende problemen

Nu zullen we kort de problemen bespreken die vaak voorkomen bij collaborative filtering algoritmes [11] [2] [1]:

- Koude start: Een bericht kan pas in aanmerking voor het algoritme wanneer een gebruiker een rating heeft gegeven.
- Data sparsiteit: Dit is het geval wanneer een gebruiker nog maar weinig ratings heeft gedaan waardoor het moeilijk is om gelijksoortige gebruikers en gelijksoortige berichten te vinden.
- Schaalbaarheid: Het algoritme moet efficiënt blijven werken bij grote hoeveelheden data.

Vergelijkingsformule

Eén van de belangrijkste formules, die het collaborative filter algoritme nodig heeft, is de vergelijkingsformule [4]. De vergelijkingsformule berekent tussen twee vectoren de gelijkheid. Er bestaan veel verschillende soorten formules om dit te berekenen, maar we zullen nu de drie bekendste noemen: Cosine Similiarity, Pearson Correlation en Adjusted Cosine Similiairty. Wij hebben gekozen voor de Cosine Similiarity als vergelijkingsformule en we zullen zo dadelijk uitleggen waarom, maar eerst geven we de formule van Cosine Similarity en leggen we uit hoe het werkt:

$$sim(u_a, u_b) = \frac{\sum_{i \in I} r_{u_a,i} \times r_{u_b,i}}{\sqrt{\sum r_{u_a,i}^2} \times \sqrt{\sum_{i \in I} r_{u_b,i}^2}} \tag{4.4}$$

Hierbij zijn u_a en u_b de gebruikers die vergeleken met elkaar worden en $r_{u_a,i}$, zoals eerder gezegd de rating van gebruiker u_a op bericht i. Verder bestaat de verzameling I uit alle berichten die beide gebruikers een rating hebben gegeven. De keuze voor de cosine similarity vergelijkingsformule komt door de manier hoe wij hebben bepaald hoe wij reviews verzamelen van gebruikers. Zoals eerder verteld in dit hoofdstuk kunnen gebruikers alleen positieve reviews

achterlaten. Wij kunnen niet achterhalen of een gebruiker een bericht niet leuk vindt. Waarschijnlijk leest die het dan niet. Om terug te komen op de keuze voor onze vergelijkingsformule; de andere formules maken gebruik van de gemiddelde reviewscore van een gebruiker of in een geval van een bericht de gemiddelde score die een bericht krijgt. Deze gemiddelde score wordt in die formules vervolgens afgetrokken van de gegeven rating. Dit heeft als reden dat de ene gebruiker optimistischer reviewt dan een andere gebruiker. Echter, wanneer wij dit zouden toepassen dan heeft dit als gevolg dat wanneer alleen een klik wordt gemaakt, dat het dan altijd als iets negatiefs wordt beschouwd. Dit komt doordat, een klik de minimale waarde heeft en dus altijd gelijk of onder het gemiddelde ligt. Daarom hebben we gekozen voor cosine similarity, want die maakt niet gebruik van dat gemiddelde en ziet een klik altijd als iets positiefs. Ten slotte veranderen we nog één ding aan de vergelijkingsformule. Dit komt doordat deze formule geen rekening houdt met het feit dat bijvoorbeeld dat twee bedrijven slechts één gemeenschappelijk bericht hebben, terwijl een ander paar bedrijven tien berichten gemeenschappelijk hebben. Daarom delen we het aantal gemeenschappelijke berichten door de totaal aantal reviews gemaakt door beide bedrijven. Deze waarde doen we maal de waarde die uit de cosine similarty komt.

Gelijksoortige gebruikers

Dit subalgoritme zoekt naar gelijksoortige gebruikers en wordt er een score voor een bericht bepaald aan de hand van wat gelijksoortige gebruikers het desbetreffende bericht hebben gegeven [17] [19][16]. De formule die dit berekent, ziet er als volgt uit:

$$sur_{k,m} = \frac{\sum_{u_a \in S(u_k)} s_u(u_k, u_a)(r_{a,m})}{\sum_{u_a \in S(u_k)} |u_a|} \tag{4.5}$$

sur_k, m is de uiteindelijk score die bericht m krijgt door gebruiker k. Verder worden alle gebruikers die bericht m een rating hebben gegeven,ook wel de verzameling $S(u_k)$ vergeleken met gebruiker k en dat gebeurt dus d.m.v. de cosine similarity. De gelijkheid tussen de gebruikers$k$ en a wordt maal de rating $r_{a,m}$ van gebruiker a op bericht m gedaan. Dus de rating van gebruikers die meer op k lijken tellen zwaarder mee voor het eindresultaat. De som van deze ratings wordt uiteindelijk gedeeld door het totaal aantal gelijksoortige gebruikers om het relatief te houden.

Gelijksoortige berichten

In de vorige subsectie hebben we gekeken naar gelijksoortige gebruikers om aan de hand die gebruikers een score voor een bericht te bepalen. In dit gedeelte bespreken we het item-based subalgoritme. Dit algoritme geeft een score aan een bericht door te kijken naar gelijksoortige berichten die de gebruiker een

rating heeft gegeven [17] [19][16]. De formule om die score te bepalen, heeft de volgende structuur:

$$sir_{k,m} = \frac{\sum_{i_b \in S(i_m)} s_i(i_m, i_b)(r_{k,b})}{\sum_{i_b \in S(i_m)} |i_b|} \tag{4.6}$$

Deze formule lijkt heel erg op de formule die gebruikt wordt voor gelijksoortige gebruikers. Echter, nu wordt het bericht m vergeleken met alle berichten b en dat zijn de berichten waar de gebruiker m een rating op heeft gedaan. In dit geval geldt weer dat daardoor de berichten die meer op k lijken dat de rating op die berichten zwaarder meetelt. Ook hier wordt gedeeld door het totaal aantal gelijksoortige berichten dat gebruikt is en weer om het relatief te maken.

Combineren scores

In de vorige twee subsectie hebben we gezien hoe aan de hand van de user- en itembased collaborative filtering scores worden bepaald voor een bericht. Deze twee scores moeten echter ook worden gecombineerd. Dit doen door een gewicht te introduceren en dit gewicht maal de twee scores te doen [17] [19][16]. Vervolgens tellen we deze twee uitkomsten bij elkaar op. Onderstaande formule geeft dit weer:

$$x_{k,m} = \lambda \times sur_{k,m} + (1 - \lambda) \times sir_{k,m} \tag{4.7}$$

Waarbij dus $sur_{k,m}$ en $sir_{k,m}$ respectievelijk de userbased en itembased score zijn en λ het gewicht. Door op deze manier de scores bij elkaar op te tellen, blijven ze op dezelfde schaal en uit studies blijkt ook dat deze manier het meest accuraat is [16].

Om de uiteindelijk score bepalen voegen we nog een 'smoothing' factor toe. Deze factor wordt berekend door de score van gelijksoortige berichten reviews gemaakt door gelijksoortige gebruikers. We voegen die 'smoothing' toe, omdat het extra informatie geeft over de gebruiker [17] [19][16] en helpt bij het data sparsity probleem. De gelijkheid tussen deze gelijksoortige gebruikers en hun gelijksoortige berichten wordt berekend door de volgende formule:

$$sim(i_a, i_b)(u_c, u_d) = \frac{1}{\sqrt{(1/s_u(u_a, u_c))^2 + (1/s_i(i_b, i_d))^2}} \tag{4.8}$$

Waarbij s_u de gelijkheid berekend tussen de gebruiker en de gelijksoortige gebruiker en waarbij s_i de gelijkheid berekend tussen het bericht en het gelijksoortige bericht. Deze gelijkheidsfunctie is zo opgesteld, zodat de waarde die eruit komt altijd lager is dan de waarde van user- of itembased. Aan de hand van deze gelijkheidfunctie wordt vervolgende de score bepaald voor het bericht aan de hand van gelijksoortige berichten reviews gemaakt door gelijksoortige gebruikers [17][16]:

$$suir_{k,m} = \frac{\sum_{i_a \in S(i_m)} \sum_{u_b \in S(i_k)} sim(i_m, i_a)(u_k, u_b) \times r_{u_b, i_a}}{\sum_{i_a \in S(i_m)} \sum_{u_b \in S(i_k)} |u_b|} \tag{4.9}$$

Deze formule komt eigenlijk op hetzelfde neer als de formules voor user- en itembased. Alleen in dit geval pak je een andere gelijkheidsfunctie waarin je het gelijksoortige bericht gooit en de gelijksoortige gebruiker.

Een belangrijke vraag is nu hoe deze 'smoothing' factor wordt gebruikt in onze uiteindelijke formule. Dit zullen we doen door nog een gewicht in de formule toe te voegen [17] [16]. Dus onze uiteindelijk formule ziet er nu zo uit:

$$x_{k,m} = (1-\delta)\lambda \times sur_{k,m} + (1-\delta)(1-\lambda) \times sir_{k,m} + \delta \times suir_{k,m} \qquad (4.10)$$

Het enige dat nu nog moet gebeuren is het bepalen van de gewichten. In studie [16] hebben ze een empirisch onderzoek gedaan voor welke waardes het beste zijn voor beide gewichten. Daaruit bleek dat de beste resultaten werden gehaald $\lambda = 0.7$ en $\delta = 0.7$ en deze waardes hebben wij overgenomen in ons algoritme. Dit algoritme voor collaborative filtering is volledig gebaseerd op de papers [17] [19][16].

Evaluatie

Een korte evaluatie in hoeverre ons algoritme de bekende problemen bij collaborative filtering oplost:

- Koude start: Dit is bij ons nog steeds het geval, echter door gebruik van het contentbased subalgoritme krijgen berichten zonder rating wel een score.
- Data sparsiteit: Ons algoritme heeft hier veel minder last van, want door het gebruik van user- en itembased en de 'smoothing' factor werkt het algoritme stukken bij weinig data in vergelijking met andere algoritmes.
- Schaalbaarheid: Ons algoritme is redelijk schaalbaar, maar hierover zullen we het nog uitgebreid hebben in het hoofdstuk Testen.

Ontwerpproces

Het ontwerp van het collaborativefilter is vaak aangepast gedurende de implementatie en het heeft veel tijd gekost qua doorzoeken van wetenschappelijke papers en het begrijpen van de deze papers. Eerst hadden we een userbased subalgoritme ontworpen waar wij de gebruikersprofielen die in contentbased worden berekend met elkaar worden vergeleken. Hiervoor hadden we ook een aparte vergelijkingsformule voor gemaakt, maar eenmaal geïmplementeerd vonden we dit algoritme niet optimaal werken, omdat onder andere de waardes die eruit kwamen niet tot tevredenheid stemde. Ook was het niet erg efficiënt, omdat voor iedere gebruiker een gebruikersprofiel moet worden gemaakt, aangezien we dit gebruikersprofiel pas wordt gemaakt tijdens het produceren van aanbevelingen, gaat dat erg veel tijd kosten. Pre-processing zou dit wel een stuk sneller kunnen maken [18], maar zoals gezegd wil het bedrijf zo min mogelijk data opslaan.

Vervolgens hebben we gekeken naar andere algoritmes. Toen hebben we eerst hebben gekeken naar algoritmes die ontwikkeld zijn voor de Netflixprice. De Netflixprice hield in dat je een miljoen kon winnen wanneer je het huidige collaborative filter algoritme van het Netflix systeem met 10 procent verbeterd. Echter, het bleek dat de meeste algoritme die daarvoor zijn ontwikkeld niet toepasbaar zijn op andere problemen, omdat ze volledig waren aangepast waren aan het Netflix-probleem [14]. Eén methode de Matrix factorization bleek wel veelbelovend voor gebruik [13] [15] [20] [14] [6] [12] en hier hebben we ook flink wat tijd in gestoken om het te begrijpen en te kijken hoe we dit algoritme zouden kunnen toepassen. Echter, er was veel pre-processing voor nodig en snapten we sommige stappen in het algoritme niet en dus lieten we deze methode varen.

Uiteindelijk kwamen we bij het algoritme dat we hierboven hebben beschreven. De keuze voor dit algoritme was onder andere dat het redelijk makkelijk te begrijpen waardoor ook later de mensen van CHAINels weinig moeiten zullen hebben om eventuele aanpassingen, indien nodig, aan het algoritme toe te passen. Ook kwam het uit deze studies naar voren dat ze redelijk accuraat zijn. Bovendien, waren de papers die dit algoritme beschreven, veel geciteerd. Tijdens de implementatie van dit algoritme zijn we niet gestopt in het doorzoeken van de literatuur, maar hebben we gekeken naar mogelijke verbeteringen in de literatuur voor dit algoritme. We hadden geen tijd meer om deze verbeteringen toe te passen. De mogelijkheden ter verbetering van dit algoritme zullen we behandelen in het hoofdstuk Aanbevelingen.

4.2.4 Tijdsfilter

Het timebased algoritme berekent een score op basis van hoe oud een bericht is. Als tijd wordt de plaatsingstijd van het bericht genomen. Bij events wordt ook rekening gehouden met de begin- en einddatum van het event.

We gebruiken de formules die we besproken hebben in 4.2.4. We laten het bericht ongeveer gedurende een dag langzaam lineair aflopen. We verwachten dat mensen ongeveer een keer per dag zullen inloggen, dus willen we dat de berichten van de afgelopen 24 uur relevantie krijgen. Als de dag om is, laten we het bericht exponentieel aflopen, maar wel met zo'n kleine krimpfactor dat de relevantie van het bericht pas na ongeveer twee maanden afgenomen is tot een honderdste van de oorspronkelijke waarde.

Omdat onze content-based en collaborative algoritmes door het reviewgedrag een score teruggeven tussen de 0 en 3 hebben we om de verschillende algoritmes makkelijker met elkaar te kunnen vergelijken de waardes van het timebased algoritme ook geschaald tot een waarde tussen de 0 en 3. Dit maakt voor de resultaten van het algoritme echter niks uit.

De invloed van tijd op de relevantie van een bericht verschilt per type bericht. Zo heeft tijd een andere invloed op een bericht over een evenement dan op een nieuwsbericht. Hieronder geven we weer hoe we van plan zijn om de leeftijd van een bericht ook te betrekken in ons algoritme.

Nieuws-, vraag- en aanbodberichten Bij nieuwsberichten, infoberichten en vraag- en aanbodberichten is de relevantie als het bericht net geplaatst is hoog, maar naarmate de tijd verstrijkt worden deze berichten steeds minder interessant. Op een gegeven moment worden de nieuwsberichten zo oud dat ze alleen interessant worden voor de archieven. Dan hoeven deze berichten niet langer getoond te worden. Als een bericht net geplaatst is, moet het bericht hoog komen te staan en mag het niet te snel aflopen. Als een bericht een tijdje in de Journals van verschillende gebruikers gestaan heeft, wordt het oud nieuws en mag het bericht sneller gaan aflopen. Met een lineaire formule kunnen we berichten geleidelijk laten aflopen en met een gebroken exponentiële formule kunnen we de berichten als ze oud nieuws zijn snel in relevantie laten aflopen. We kunnen daarom gebruikmaken van een combinatie van een lineaire en exponentiële formule. In het begin zal de relevantie afnemen volgens de lineaire formule, maar na een bepaalde tijd zal de relevantie afnemen volgens de exponentiële formule.

Onze formule heeft dus de volgende vorm:

$$S(t) = \begin{cases} b - at_v & \text{als } t_v < c \\ bg^{t_v - c} & \text{als } t_v \geq c \end{cases}$$

waarin $t_v = t - t_b$ de verlopen tijd representeert en a, b, c en g constantes zijn.

Bepaling constantes

a is een kleine waarde die ervoor zorgt dat een bericht van 5 minuten geleden toch net iets belangrijker is dan een bericht van 16 uur geleden.

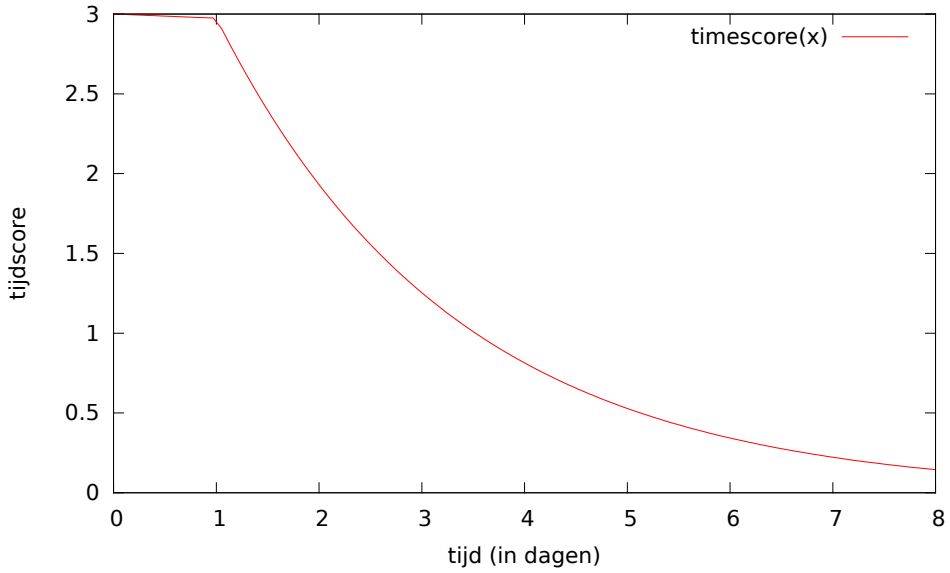
b is de maximale waarde die de score kan aannemen. De waarde hiervan maakt voor de werking van het tijdalgoritme weinig uit omdat de waarde altijd met deze factor geschaald wordt. Omdat de eerder genoemde algoritmes beide een maximale score van 3 kunnen aannemen, hebben we deze waarde gelijkgesteld aan 3.

c is de tijd waarin het bericht heel erg relevant is. Na brainstormen hebben we onder de aanname dat bedrijven CHAINels om de dag bekijken deze waarde gelijkgesteld aan een periode van 24 uur.

g is de exponentiële afname. Deze constante is zodanig gekozen dat de score van het bericht na een week ongeveer is gedaald tot een 10e van de maximale score. Hierdoor heeft het bericht gedurende een week nog een redelijke relevantie.

Bij deze formule hoort de grafiek in figuur 4.2.4.

Events Voor events lijkt het ons logisch om de relevantie van het bericht vlak na het plaatsen ervan hoog te houden. Het event is dan net aangekondigd en als organisator wil je dan dat bedrijven ervan op de hoogte worden gesteld en



Figuur 4.2: Tijdscoregrafiek voor nieuws en vraag- en aanbodberichten

erover kunnen praten met hun connecties. Zodra er wat tijd verstrijkt, neemt de relevantie van het bericht enigszins af. We kunnen aannemen dat bedrijven die geïnteresseerd zouden kunnen zijn het bericht tegen die tijd toch wel gelezen hebben en dat bedrijven die er op dat moment nog niet van gehoord hebben er ook niet meer van gaan horen.

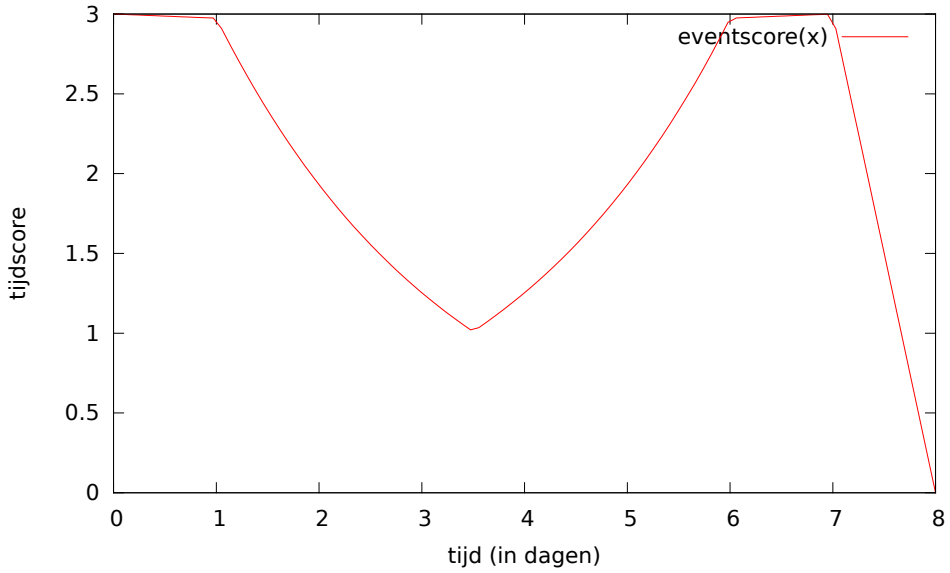
Als het event bijna gaat plaatsvinden is de relevantie van het event weer hoog, want dan is het event bijna in zicht. Het opnieuw toekennen van een hoge relevantie op dit moment zorgt ervoor dat de interesse in het event kort van tevoren weer stijgt en functioneert als een soort laatste herinnering voor bedrijven die nog niet hebben besloten of ze wel of niet deel willen nemen aan het event.

Omdat we events wel willen vergelijken met de andere berichttypes (een event kan immers ook een hoofdartikel zijn) hebben we de formule die we voor de andere berichttypes gebruikten als basis genomen.

Onze formule heeft dus de volgende vorm:

$$S(t) = \begin{cases} \max S(t - t_b), S(t_b - t) & \text{als } t < t_b \\ S(t - t_b) & \text{als } t \geq t_b \text{ en } t_e < t \\ 0 & \text{als } t_e \geq t \end{cases}$$

waarin $S(t)$ de formule is die we voor de andere berichtcategorieën gebruiken, t de huidige tijd en t_b en t_e respectievelijk de begin- en eindtijd van het event.



Figuur 4.3: Tijdscore voor een evenement dat op dag 0 aangekondigd is, op dag 7 begint en van dag 7 tot dag 8 duurt.

Een grafiek voor de score van een evenement kan eruit zien zoals in figuur 4.2.4.

4.2.5 Combineren

Een belangrijke stap in het aanbevelingsalgoritme is het combineren van de drie subalgoritmes. In eerste instantie deden we het combineren van de drie subalgoritmes op de volgende manier:

$$rec(bericht) = cb(bericht) \times cf(bericht) \times tb(bericht) \tag{4.11}$$

Hierbij is $cb(bericht)$ het content-based subalgoritme, $cf(bericht)$ het collaborative filter subalgoritme en $tf(bericht)$ het timefilter subalgoritme. Om de snelheid van het algoritme te verhogen en wegens beperkingen bij het opslaan van de data (hierover meer in het volgende hoofdstuk) hebben we besloten dat we eerst het timefilter over de gehele set van berichten laten lopen. Uit het resultaat pakken we dan de bovenste X berichten en die gebruiken we als input voor de content-based en collaborative filter subalgoritmes. De uiteindelijke formule ziet er dan als volgt uit:

$$
\begin{aligned}
V &= \text{berichtenset} \\
W &= \{tb(bericht) | bericht \in V\} \\
X &= \{b_1, b_2, \ldots, b_{200} \in V | \\
&\quad tb(b_1), tb(b_2), \ldots, tb(b_{200}) \text{ zijn de 200 hoogste getallen uit W } \} \\
scores &= \{rec(bericht) = cb(bericht) \times cf(bericht) \times tb(bericht) | bericht \in X\}
\end{aligned}
$$

De berichten die de hoogste score krijgen uit deze formule zijn dan de meest relevante berichten voor de gebruiker.

5. Implementatie

We zullen het nu gaan hebben over de implementatie van ons systeem. Eerst zullen we het hebben over de algemene structuur van ons systeem en hoe dit functioneert binnen het systeem van CHAINels. Vervolgens bespreken we de implementatie van de back-end van het systeem en welke componenten daar aanwezig zijn. Ten slotte sluiten we af met het uitleggen van de implementatie van de front-end.

5.1 Algemene structuur

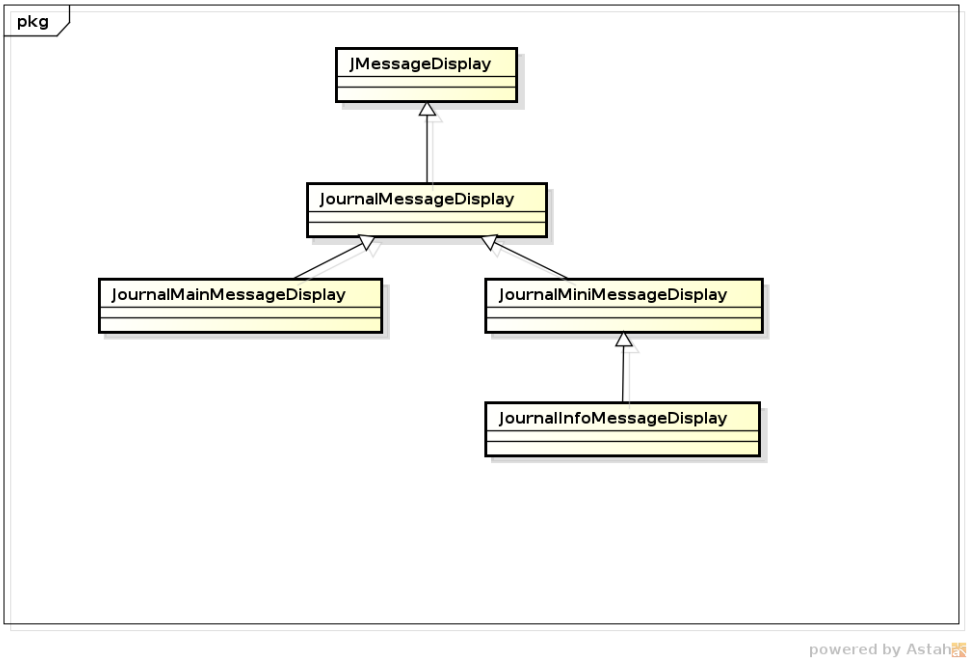
Onze code bestaat uit de frontend klasses en de algoritme klasses. De frontend klassen zijn te vinden in de mappen *html* en *javascript*. De klassen in *html* zijn verantwoordelijk voor de weergave en de klassen in *javascript* voor het gedrag van de website. De backend klassen zijn te vinden in de map *php*.

5.2 Front-End

Na het uitwerken van het ontwerp dat ten grondslag lag aan het ontwikkelen van de Journal zullen we nu duidelijk maken met welke middelen de Journal daadwerkelijk in elkaar is gezet. Allereerst zullen we een algemeen beeld geven van de structuur van de code achter de Journal en vervolgens zullen we aan de hand van de verschillende functionaliteiten die de Journal biedt uitleggen hoe deze gerealiseerd zijn, om zo uiteindelijk een volledig beeld te geven van de onderliggende technieken die de Journal tot stand hebben gebracht.

5.2.1 De basis

De code achter de Journal bestaat uit een combinatie van HTML, CSS, JavaScript en PHP. We maken gebruik van een aantal PHP-klassen waarvan de meeste bedoeld zijn om berichten binnen de Journal correct weer te geven. De basis van deze zogenaamde weergaveklassen is vastgelegd in de klasse *JMessageDisplay*. Deze (statische) klasse bevat methodes die nodig zijn voor het weergeven van alle soorten berichten, vandaar dat de overige weergaveklassen ook erven van *JMessageDisplay*.



Figuur 5.1: Een klassediagram waarin de relaties tussen de verschillende weergaveklassen wordt beschreven

In het ontwerp van de Journal hebben we al aangegeven dat er naast reguliere berichten ook een hoofdbericht, een overzicht van berichten die de gebruiker volgt en een aantal kleine informatieberichten worden getoond. Voor elk van deze classificaties van berichten hebben we een aparte klasse die de weergave ervan verzorgt. Voor reguliere berichten is dit de klasse *JournalMessageDisplay*, voor hoofdberichten wordt de klasse *JournalMainMessageDisplay* gebruikt, voor de gevolgde berichten is er de klasse *JournalMiniMessageDisplay* en de informatieberichten worden afgehandeld door de klasse *JournalInfoMessageDisplay*.

Een duidelijk, schematisch overzicht van hoe deze klassen onderling met elkaar samenhangen is gegeven in figuur 5.1. We hebben gekozen voor deze structuur omdat veel soorten berichten overeenkomsten met elkaar vertonen. Hier kunnen we efficiënt gebruik van maken in de code door steeds specifiekere klassen te laten erven van de wat algemenere klassen. Zo bevat *JournalMessageDisplay* methodes om bepaalde elementen van berichten zoals de tijden en titels weer te geven die ook terug te vinden zijn in de andere soorten berichten.

Het zogenaamde hart van de Journal ligt in de *JournalHTML*-klasse. Deze klasse erft een deel van haar methodes van *Page*, wat binnen dit systeem gebruikelijk is voor PHP-klassen die verantwoordelijk zijn voor het samenstellen

van de HTML-code voor een deel van een pagina (binnen CHAINels ook wel een *Pagelet* genoemd). Zo kan de Journal op dezelfde wijze als elke andere pagina geladen worden, namelijk via het BigPipe-systeem. Dit systeem zorgt ervoor dat aan elke Pagelet een prioriteit toegekend kan worden waar tijdens het laden van een pagina rekening mee gehouden wordt. Zo worden de delen van de pagina met de hoogste prioriteit ook als eerst geladen. Door onze *JournalHTML*-klasse ook in dit systeem te verwerken, kunnen we dus ook een prioriteit aan de Journal toekennen om deze zo een plek te geven in de rangorde van de verschillende paginadelen.

De *JournalHTML*-klasse bevat methodes om alle delen van de Journal in nette HTML-code op te vragen. Via deze klasse wordt de Journal dus daadwerkelijk naar het browserscherm van de gebruiker gestuurd. Ze maakt daarbij gebruik van de bovengenoemde weergaveklassen om ook de berichten netjes te formatteren. Alle HTML-code wordt opgebouwd aan de hand van een aantal al bestaande PHP-klassen binnen het systeem die hier speciaal voor zijn ontworpen. De voornaamste van deze klassen is *Basics*, met onder andere methodes als *div()* en *link()* om respectievelijk <*div*>- en <*a*>-elementen weer te geven.

Omdat onze Journal dynamisch is, hebben we ook gebruik gemaakt van JavaScript zodat de gebruiker kan interacteren met de Journal. Als basis hiervoor hebben we de populaire jQuery-library gebruikt (http://jquery.com), wat een fijne syntax biedt om regelmatig voorkomende problemen snel en makkelijk op te lossen. Bovendien werken de methodes die jQuery aanbiedt op alle moderne browsers, waardoor we ons daar zelf minder zorgen over hoeven te maken.

Onze eigen JavaScript-code is volledig in het bestand *journal.js* terug te vinden. Dit bestand bevat een verzameling van functies die de gebruiker laten interacteren met de Journal, maar is voornamelijk bedoeld om nieuwe data van de server op te halen zonder dat de pagina vernieuwd hoeft te worden. Dit gebeurt via Ajax, een techniek binnen JavaScript die bedoeld is om data naar de server te sturen en van de server te ontvangen zonder dat daarbij de weergave en het gedrag van de openstaande pagina verstoord worden. Dit wordt gedaan door data naar een PHP-script op de server te sturen waar dan vervolgens bepaald wordt wat er met die data gedaan moet worden en welke data teruggestuurd moet worden. Binnen het systeem van CHAINels worden al deze taken afgehandeld door het script *Ajax.php*, wat op basis van een string die mee wordt gestuurd met het Ajax-verzoek bepaalt om welke data gevraagd wordt of waar de ontvangen data opgeslagen dient te worden.

De vormgeving van de Journal is vrijwel compleet bepaald in het CSS-bestand *journal.css*. Alle regels die bepalen hoe de Journal eruitziet, zijn hierin terug te vinden. De enige uitzondering hierop is het plaatsen van een verticale scrollbar voor de inhoud van de Journal. Zoals we in het ontwerp van onze Journal hebben aangegeven, hebben we uiteindelijk gekozen om van de Journal een lange pagina te maken waarin de gebruiker naar beneden moet scrollen om meer berichten te kunnen lezen. Vaak kan een scrollbar prima geplaatst worden met CSS alleen, maar door de positionering van de Journal werd dat hier lastig.

De Journal moest namelijk met een vaste positie op de pagina geplaatst worden zodat deze niet mee zou bewegen als de gebruiker buiten de Journal zou scrollen (in CSS heet dit principe *fixed positioning*). Hiermee wordt de Journal ook tot helemaal onderaan het scherm van de gebruiker getrokken, zodat er geen ruimte tussen de onderkant van het scherm en de Journal is. De hoogte van de Journal is dus afhankelijk van het beeldscherm van de gebruiker en om de scrollbar de juiste hoogte te geven, moet deze hoogte wel bekend zijn. Omdat CSS geen informatie van de browser kan gebruiken, wordt de hoogte van de Journal nu bij het laden ervan bepaald via JavaScript en vervolgens aan CSS gevoerd zodat de scrollbar op ieder scherm helemaal doorloopt tot aan de onderkant van de Journal.

5.2.2 De functionaliteiten

Nu we een beeld geschetst hebben van de structuur van onze code en welk deel van de code verantwoordelijk is voor welke taken zullen we de functionaliteiten van de Journal een voor een bespreken. Per functionaliteit zullen we aangeven hoe we het gerealiseerd hebben met onze code. We onderscheiden de volgende zeven meest belangrijke functionaliteiten die we zullen gebruiken om onze code uit te leggen:

- het uitschuiven van de Journal,
- het laden van de berichten,
- het weergeven van de berichten,
- het updaten van de berichten,
- het laden van een pagina voor een specifieke categorie,
- het volgen van een bericht,
- het handmatig vernieuwen van de Journal.

Het eerste wat een gebruiker moet doen wanneer hij of zij de Journal wil bekijken, is de Journal uitschuiven zodat de inhoud zichtbaar wordt. Voordat dit gebeurt, is de inhoud van de Journal verborgen en is alleen de uitschuifbalk zichtbaar. Bij het laden van de pagina wordt er een event listener toegevoegd aan deze uitschuifbalk die registreert wanneer een gebruiker erop klikt. Zodra een dergelijk event geregistreerd wordt, zorgt een aanroep naar de jQuery-functie *animate()* ervoor dat de inhoud van de Journal geleidelijk aan getoond wordt op een vooraf gedefinieerde snelheid.

Ook wordt de achtergrondafbeelding van de uitschuifbalk aangepast. Voordat de Journal zichtbaar is, staat hier een pijl naar rechts om aan te geven dat de Journal naar rechts uit zal schuiven wanneer er geklikt wordt. Zodra de Journal eenmaal is uitgeschoven, wordt deze pijl omgedraaid voor het geval de gebruiker de Journal hierna weer in wil klappen. Het inklappen gebeurt voor de

rest op praktisch dezelfde manier als het uitklappen, maar dan net andersom.

De tweede genoemde functionaliteit is het laden van de berichten. Eigenlijk gebeurt dit chronologisch gezien nog voordat de gebruiker de Journal uitschuift. Bij het laden van de Journal wordt de inhoud alvast geplaatst, ook al is deze op dat moment nog onzichtbaar. Het ophalen van de berichten gebeurt in de *getJournalMessages()*-methode van de PHP-klasse *JournalHTML*. Deze methode gebruikt een aantal andere methodes uit de *JournalHTML*-klasse waarvan de namen al aangeven welke deelfunctionaliteit deze methodes precies leveren. Eerst wordt *getMainMessage()* aangeroepen om het hoofdartikel te isoleren, vervolgens worden door *getRegularColumns()* de kolommen met reguliere berichten geladen en tenslotte wordt de laatste kolom aangemaakt door ofwel *getInfoColumn()* als deze kolom gevuld moet worden met informatieberichten (zoals op de hoofdpagina van de Journal), ofwel *getStayUpToDateColumn()* in het geval dat deze kolom gevolgde berichten bevat (zoals op een categoriale pagina).

De reguliere berichten en het hoofdbericht worden aangeleverd door de PHP-klassen die ons algoritme verzorgen. Deze zijn dan al voorgesorteerd en hoeven dan alleen nog maar netjes in HTML-code geschreven te worden. Dit brengt ons bij onze derde functionaliteit, namelijk het weergeven van de berichten. Zoals al eerder is gezegd, worden hier de vijf weergaveklassen voor gebruikt. De hoogste, meest abstracte klasse hiervan, *JMessageDisplay*, bevat methodes als *getTitle()* en *getContent()* waarvan de functionaliteit voor vrijwel alle onderliggende klassen zowel benodigd als identiek is. Deze functies zijn bedoeld voor het weergeven van bepaalde delen van de berichten en zijn ook vrij eenvoudig.

In de klasse *JournalMessageDisplay*, die erft van *JMessageDisplay*, wordt de methode *displayMessage()* gedefinieerd. Deze methode combineert allerlei deelfuncties om de HTML-code voor een heel bericht weer te geven. Deze methode neemt als parameter een instantie van de *BaseMessage*-klasse aan, die al vooraf in het systeem stond. Deze klasse zelf is een abstracte klasse, dus in de praktijk worden instanties van de *DeskMessage*- of *InfoMessage*-klasse gebruikt, afhankelijk van wat voor soort bericht moet worden weergegeven. Deze instanties bevatten alle benodigde informatie voor het weergeven, zoals de inhoud van het bericht, het bedrijf dat het bericht heeft geplaatst en de tijd waarop het bericht is geplaatst.

Omdat alle overige weergaveklassen op hun beurt weer erven van *JournalMessageDisplay* heeft elke klasse toegang tot deze *displayMessage()*-methode, alhoewel de functionaliteit hiervan per klasse wel verschilt. Deze methode wordt dus in elke klasse weer overschreven, maar het feit dat de naam van de methode voor elke klasse hetzelfde is, zorgt wel voor duidelijkheid en gemak bij het programmeren.

Met deze methode worden alle kolommen en het hoofdbericht van hun HTML-code voorzien. Het hoofdartikel wordt - als een geschikt artikel gevonden is - als eerst weergegeven. Vervolgens worden de kolommen met de reguliere berichten gevuld. Hierbij worden de berichten om en om toegevoegd aan de kolommen, zodat deze kolommen uiteindelijk evenveel berichten bevatten en dus ongeveer

even lang zijn. De derde kolom wordt met informatieberichten of met gevolgde berichten gevuld, afhankelijk van de pagina van de Journal die geopend is.

We hebben in de *JournalHTML*-klasse een maximaal aantal berichten ingesteld dat in eerste instantie wordt geladen. Vaak is een gebruiker die de Journal opent niet geïnteresseerd in alle berichten uit de Journal, maar slechts in de meest relevante. Het zou dus zonde zijn om elke keer dat de Journal wordt geopend alle berichten te laden als dat niet eens nodig is. Om toch alle berichten te kunnen blijven zien hebben we het zo gemaakt dat er pas nieuwe berichten geladen worden wanneer een gebruiker naar beneden scrolt tot ongeveer driekwart van de pagina. Dit is gelijk ook onze vierde functionaliteit. Elk type bericht heeft zijn eigen constante binnen de klasse die aangeeft hoeveel berichten er in het begin worden geladen en hoeveel er per update bij mogen komen. Zo is de constante INITIALMESSAGECOUNT, die aangeeft hoeveel reguliere berichten er bij het openen van de Journal maximaal geladen zijn, nu ingesteld op tien.

Het laden van extra berichten is typisch iets waar Ajax uitermate goed geschikt voor is. Met JavaScript wordt bijgehouden hoe ver de gebruiker door de Journal gescrold is. Zodra gedetecteerd wordt dat deze voor driekwart naar beneden is gescrold, wordt een Ajax-verzoek naar de server gestuurd om het juiste aantal berichten op te halen. Hiervoor wordt een *offset* meegestuurd, die vertelt hoeveel berichten er overgeslagen moeten worden (namelijk alle berichten die al in de Journal staan), en een *amount*, waarmee wordt aangegeven hoeveel berichten er opgehaald moeten worden. Het eerder genoemde *Ajax.php*-bestand ontvangt deze data en voert hiermee de juiste bewerkingen uit op basis van een string die als derde parameter wordt meegestuurd. Deze string vertelt *Ajax.php* welke code moet worden uitgevoerd met de meegestuurde parameters als invoer. In *Ajax.php* staat namelijk een *switch()*-statement met voor elke toegestane string een aparte case.

In het Ajax-script zijn drie aparte cases voor het ophalen van extra berichten: een voor het ophalen van reguliere berichten, een voor informatieberichten en een voor gevolgde berichten. Deze cases maken respectievelijk gebruik van de methodes *getRegularColumns()*, *getInfoColumn()* en *getStayUpToDateMessages()* uit de *JournalHTML*-klasse om de juiste data - correct geformatteerd in HTML - op te halen en terug te sturen als antwoord op het Ajax-verzoek. De methode *getStayUpToDateMessages()* is een deelfunctie van *getStayUpToDateColumn()* en retourneert alleen de HTML van de berichten in de kolom, terwijl deze laatste functie daarbij ook de HTML van de kolom zelf meestuurt. Dat hebben we in dit geval niet nodig.

Nadat het Ajax-script de juiste data heeft teruggestuurd, ontvangt JavaScript als antwoord correcte HTML code. Deze code kan dan onmiddellijk toegevoegd worden aan de juiste kolom in de Journal zonder dat de gebruiker hier last van heeft. Zo kan de gebruiker ongestoord door blijven scrollen terwijl op de achtergrond via Ajax steeds nieuwe berichten geladen worden.

De vijfde functionaliteit is het laden van een pagina voor een bepaalde categorie van berichten, zoals bijvoorbeeld de pagina die enkel nieuwsberichten bevat.

Bovenin de Journal staat, zoals in het ontwerp ook besproken is, voor elke categorie een icoon. Als de gebruiker op een van deze iconen klikt, wordt de bijbehorende pagina via Ajax geladen. Ook worden de iconen opnieuw geladen, omdat het actieve icoon dat bij de geladen pagina hoort iets donkerder moet worden.

Het laden van zo'n categoriepagina gebeurt eigenlijk op dezelfde manier als het laden van de hoofdpagina. Het enige verschil is eigenlijk dat het nu via Ajax gebeurt en dat er geen informatieberichten in de rechterkolom worden geplaatst, maar in plaats daarvan gevolgde berichten uit de categorie van de pagina. Via JavaScript worden bij het laden van de Journal event listeners toegekend aan de iconen die, wanneer op een icoon geklikt wordt, een Ajax-verzoek naar de server sturen. Dit wordt op dezelfde wijze afgehandeld als bij het laden van extra berichten, alleen wordt er nu natuurlijk anders omgegaan met de data die naar de server wordt gestuurd. Het principe erachter is echter hetzelfde.

Met dit Ajax-verzoek wordt de categorie van de pagina die geladen moet worden meegestuurd. Het Ajax-script roept vervolgens de *getJournalMessages()*-methode uit *JournalHTML* aan met deze categorie als parameter. In dat geval geeft *getJournalMessages()* net als bij het laden van de hoofdpagina de complete HTML-code terug voor alle berichten die geplaatst moeten worden, maar nu alleen met berichten uit de meegegeven categorie. Alle overige berichten worden eruit gefilterd door simpelweg alle berichten af te gaan en alleen een bericht te behouden als de categorie ervan overeenkomt met de gewenste categorie.

Daarnaast wordt er ook een tweede Ajax-verzoek naar de server gestuurd om de nieuwe HTML-code voor de iconen op te halen. Niet alleen de iconen worden hierbij vernieuwd, maar er wordt ook in het midden van de horizontale balk waarin de iconen zich bevinden een kop geplaatst waar men aan kan zien welke pagina nu geopend is. Als de nieuwspagina geladen is, komt hier bijvoorbeeld de tekst "Nieuws"te staan.

Zoals gezegd wordt op deze categoriepagina's een overzicht van de gevolgde berichten weergegeven in de rechterkolom. Hier is het mogelijk om, door een bericht uit de Journal naar deze kolom te slepen met de muis, heel eenvoudig een nieuw bericht te volgen. Het slepen van HTML-elementen is niet een standaard functionaliteit binnen browsers, dus om dit toch mogelijk te maken hebben we jQuery UI, een handige uitbreiding op de jQuery-library, gebruikt (http://jqueryui.com). Deze uitbreiding is speciaal gemaakt om redelijk complexe interacties met HTML-elementen makkelijk te maken.

In jQuery UI is er een functie, *draggable()*, waarmee het mogelijk is om een selectie van HTML-elementen *draggable* te maken. Dit voegt verder geen functionaliteit toe, het maakt het alleen mogelijk om de elementen over de pagina te verslapen. In combinatie met een andere functie, *droppable()*, wordt het echter wel interessant. Met deze functie kan aan een aantal HTML-elementen een event listener toegekend worden die wacht tot er draggable element bovenop wordt losgelaten. Op dat moment wordt een zelf te specificeren JavaScriptfunctie uitgevoerd.

Elke keer als berichten worden geladen in onze Journal worden deze *draggable()*- en *droppable()*-functies uitgevoerd op de juiste berichten (dus de reguliere berichten worden draggable gemaakt en de gevolgde berichten droppable). Als een van de reguliere berichten gesleept wordt naar de gevolgde berichten wordt er een Ajax-verzoek naar de server gestuurd met het identificatienummer van het bericht (het MID van het bericht) als parameter. Het Ajax-script roept dan de *recommendMessage()*-methode van de *DeskMessage*-klasse aan (deze was al aanwezig in het systeem). Deze functie slaat in de database op dat een gegeven bedrijf een bericht met het MID dat werd meegestuurd met het Ajax-verzoek volgt.

Als dit is gebeurd, wordt er een tweede Ajax-verzoek naar de server gestuurd om de verzameling met gevolgde berichten opnieuw te laden, zodat de gebruiker onmiddellijk ziet dat het gewerkt heeft. In dit geval doet het Ajax-script niks anders dan opnieuw de *getStayUpToDateMessages()*-methode van *JournalHTML* aanroepen en de resulterende HTML-code terugsturen zodat deze weergegeven kan worden.

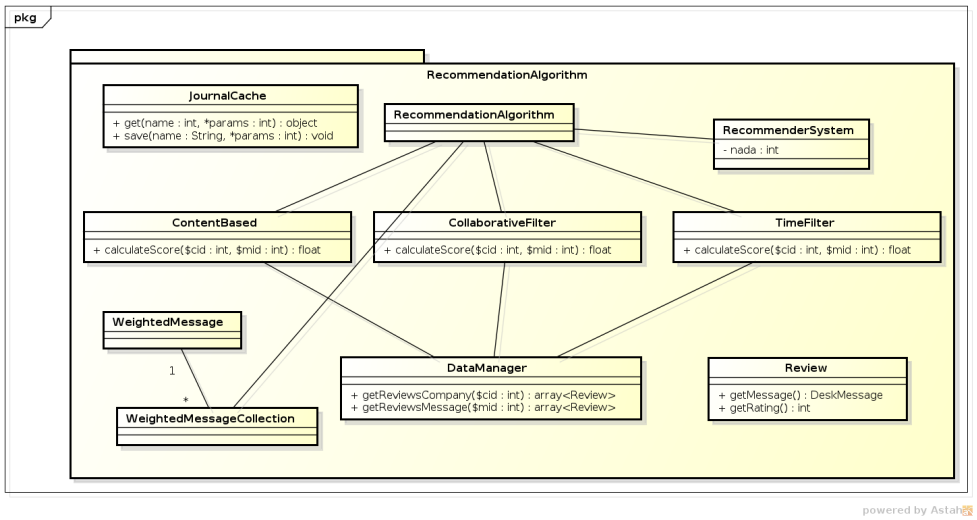
Tenslotte is er nog een laatste actie die de gebruiker kan ondernemen, namelijk het handmatig vernieuwen van de Journal. Het is nu zo dat de inhoud van de Journal opnieuw geladen wordt als een gebruiker inlogt en vanaf dan hetzelfde blijft totdat de gebruiker uitlogt en weer opnieuw inlogt. Als de gebruiker in de tussentijd echter wil kijken of er al nieuwe berichten zijn geplaatst in zijn of haar Journal, kan de gebruiker op de knop "Vernieuwen"klikken. In dat geval wordt, wederom via een Ajax-verzoek, het gehele algoritme dat de Journal samenstelt opnieuw uitgevoerd en wordt het resultaat in een cookie opgeslagen, waarbij de oude cookie wordt verwijderd. De resulterende berichten worden dan op de inmiddels bekende manier teruggestuurd als antwoord op het Ajax-verzoek en vervolgens weergegeven in de browser.

Het is ook nog mogelijk om binnen de Journal op een bericht te reageren, maar deze functionaliteit zit al in het systeem, aangezien dit op exact dezelfde manier op de profielpagina's van bedrijven gebeurt.

5.3 Back-End

De klassen in de map *php* zijn opgedeeld in kleinere packages afhankelijk van hun functie. Het algoritme bevat de implementatie van ons aanbevelingsalgoritme. De klassen werken samen om de relevantie van een bericht te berekenen. Deze waardes worden gebruikt om de berichten die in de Journal moeten verschijnen te sorteren zodat de belangrijkste berichten bovenaan komen te staan. Beide groepen roepen functies aan uit de back-end van de website, die door de mensen die bij CHAINels werken geschreven is.

De map *lib* bevat hulpklassen die door meerdere andere klassen gebruikt worden. Zo zorgt *DataManager* ervoor dat informatie uit de database opgevraagd kan worden. Verder zitten hier ook de objecten die worden gebruikt als



Figuur 5.2: Klassediagram

input en output voor de algoritmen.

De klassen in de map *cache* zorgen ervoor dat in de algoritmeklassen geen waardes dubbel berekend worden of dat er vaker dan eens een dure aanroep naar de database gedaan wordt. *JournalCache* onthoudt waardes die tijdens het uitvoeren van het algoritme nodig zijn. *JournalCookie* zorgt ervoor dat het resultaat van het complete algoritme bewaard blijft, zodat de gebruiker door de Journal kan bladeren zonder dat de inhoud van de Journal telkens opnieuw berekend hoeft te worden.

De map *test* bevat tenslotte alle tests. In het hoofdstuk 6 zal uitgebreider ingegaan worden op hoe het systeem getest wordt.

We zullen nu de werking van de back-end bespreken.

5.3.1 Program flow

Aan *RecommenderSystem* wordt door de front-end om aanbevelingen gevraagd. RecommenderSystem kijkt dan eerst of er een cookie bestaat met al eerder opgeslagen aanbevelingen erin. In het geval deze niet bestaat, zorgt deze klasse ervoor dat de berichten die in aanmerking komen voor het aanbevelingsalgoritme verzameld en worden doorgestuurd naar het algoritme. Deze verzamelde berichten komen dan in de klasse RecommendationAlgorithm terecht. Vervolgens worden de 150 meest recente berichten bepaald aan de hand van het TimeFilter algoritme. We kiezen er 150 omdat er niet heel veel meer berichten in een cookie kunnen worden opgeslagen. Daarover zullen we het in meer detail hebben in subsectie 5.3.3. Daarnaast zorgt deze beperking er ook voor dat het systeem bij grote hoeveelheden data nog redelijk efficiënt blijft. Wanneer de meest recente

berichten zijn bepaald, worden deze één voor één doorgegeven aan de content-based en collaborative subalgoritmes. Dat zijn dus respectievelijk de klassen *ContentBased* en *CollaborativeFilter*.

Algorithm 1 RecommenderSystem

```
function getRecommendations(user)
    if a cookie exist then
        recommendations ← load recommendations from cookie
    else
        messages ← load all messages
        recommendations ← calculateRecommendations(user, messages, 150)
    end if
end function
```

Algorithm 2 RecommendationAlgorithm

```
function calculateRecommendations(user, messages, max)
    messages ← get max most recent messages
    weightedMessages ← empty
    for all messages as message do
        contentbased ← Contentbased.calculateScore + 1
        collaborative ← Collaborative.calculateScore + 1
        timebased ← TimeFilter.calculateScore + 1
        score ← contentbased ∗ collaborative ∗ timebased
        weightedMessage ← new WeightedMessage(message, score)
        weightedMessages ← weightedMessages ∩ weightedMessage
    end for
    return weightedMessages
end function
```

5.3.2 Hulpklassen

DataManager In de klasse *DataManager* wordt alle informatie over het gebruikersgedrag verzameld en teruggegeven in de vorm van Review-objecten. De reviews kunnen opgevraagd worden per gebruiker of per bericht. Ook kunnen alle berichten in een netwerk of vanaf een zekere tijd opgevraagd worden. Omdat de hulpfuncties door de verschillende subalgoritmes aangeroepen worden, worden de resultaten gecached.

Review In een Review-object worden een gebruiker en een bericht opgeslagen. Ook wordt er in opgeslagen hoe de gebruiker op het bericht reageerde. Een 1 wordt opgeslagen als de gebruiker klikte, een 2 als de gebruiker het bericht volgde, en een 3 als de gebruiker een reactie achterliet. Dit gedrag is besproken in hoofdstuk 4.

5.3.3 Cache klassen

Omdat in onze functies veel functies vaker dan eens gebruikt worden, hebben we de resultaten van alle functies die vaker dan eens aangeroepen worden gecached. Ook hadden we in het begin het inladen van Companies en DeskMessages gecached door middel van een wrapperfunctie. Deze cachefuncties hadden we later niet meer nodig omdat de werkgevers besloten hadden deze functies op lager niveau te cachen, waardoor ook de performance van alle andere delen van de website verbeterd werd.

6. Testen

Tijdens dit project hebben we ook veel aandacht besteed aan het goed testen van alle functionaliteit. We hebben de journal op verschillende manieren getest. In dit hoofdstuk zullen we het eerst hebben over de unit tests die we gebruiken voor de back-end van ons systeem. Vervolgens zullen we het hebben over de gebruikerstests. In deze gebruikerstests werd vooral gekeken naar de functionaliteiten in de GUI van de Journal en de lay-out van de Journal. Daarna zullen we iets zeggen over de snelheid van het systeem en hoe het na verloop van tijd verbeterd is. Ook zullen we dan een grafiek laten wat de snelheid afhankelijk van de hoeveelheid data. Ten slotte zullen we terugblikken naar de requirements zoals we die opgesteld hebben in bijlage A en het resultaat aan de hand van de daarin besproken criteria evalueren.

6.1 Unit tests

Voor het unit testen hebben we gebruik gemaakt van het simpletest framework. We hebben voor dit framework gekozen omdat het bedrijf dit framework zodanig geconfigureerd heeft dat de tests plaatsvinden in een testdatabase zodat andere groepjes die tegelijkertijd bezig zijn in de database geen last van de tests ondervinden.

Normaal gesproken wil je bij unit tests bepaalde klassen mocken, zodat je elke klasse onafhankelijk kan testen. In het systeem van CHAINels kunnen veel klassen niet gemocked worden, omdat deze als final zijn gedeclareerd. Daarom moesten wij zelf data in de testdatabase genereren. De testdatabase is een aparte database speciaal opgezet voor de unit tests. Voor het genereren van deze data hebben wij speciale setup klasses gemaakt. Deze setup klassen maken het mogelijk om heel makkelijk nieuwe data toe te voegen. Dit komt doordat wij methodes hebben geschreven die automatisch velden invullen die nodig zijn wanneer er bijvoorbeeld een company aangemaakt wordt. Hierdoor hoeven er slechts twee of drie parameters mee gegeven te worden met hetgeen dat er getest moet worden. De volgende setupklasses hebben wij aangemaakt: SetupCompany, SetupAccount, SetupMessages en SetupReviews. In elk van deze klassen zijn de methodes setUp() en tearDown te vinden; setUp() wordt aan het begin van elke test aangeroepen zodat de data wordt gegeneeerd voor die unit test en tearDown() wordt aan het einde van elke test aangeroepen, zodat alle data ook

weer uit de database wordt gegooid. Hieronder is een klasse diagram te zien van deze setup klasses.

Elke unittest heeft een object SetupTestData en via dat object wordt de setup gestart, echter eerst wordt in de setup methode van het SetupTestData object nog de testdatabase schoongemaakt en deze schoonmaak gooit alleen de data weg die wij erin hebben gestopt. Dit is nodig, omdat er soms fatale fouten in de code zat die werd getest, waardoor de tests werden afgebroken en de tearDown() niet meer werd aangeroepen. Dit betekent dat er dan nog data achterblijft in de database en moet die eerst verwijderd uit de database. Voor de rest wordt bijna elke klasse in de back-end uitgebreid getest door een unit test.

6.2 Usertest

User tests hebben we eigenlijk vooral zelf gedaan en uit laten voeren door andere mensen die werkzaam zijn bij CHAINels. Omdat alle werknemers onze Journal kunnen zien met hun eigen testaccount, heeft iedereen bij CHAINels ons algoritme kunnen testen en feedback kunnen geven. We kregen vooral feedback over wat ze van de layout vonden. Het algoritme leek redelijk goed te werken aangezien we berichten waar veel op geklikt was en die recent zijn erg hoog zagen staan, terwijl we de andere berichten lager zagen staan.

6.2.1 Data genereren

Iedereen in ons groepje kon een testaccount aanmaken en ook door CHAINels zelf waren er een aantal testbedrijven in de database gegooid. Maar omdat de frontendfuncties in het begin nog niet allemaal geïmplementeerd waren en we meer data nodig hadden, hebben we een scriptje geschreven dat automatisch een enorme hoeveelheid data genereert. Zo werden er automatisch een aantal bedrijven aangemaakt, werden er chains tussen de bedrijven gelegd, werden en berichten geplaatst. Ten slotte werd er ook de berichten geklikt, werden er berichten gevolgd en werd er op berichten gereageerd. Met deze testset konden we ons algoritme in werking zien en konden we zelf ons eigen algoritme testen en aanpassen om de kwaliteit van de resultaten te verbeteren.

6.3 Testen van requirements

Verder hebben we de functionaliteit getest aan de hand van de door ons opgestelde requirements. Onze Journal bleek in zowel in firefox als chrome, als internet explorer goed te werken. In internet explorer is het effect van de schuifbalk echter iets lelijker dan in de andere browsers, maar dat is voor de gebruiker niet heel storend.

6.3.1 Functionele requirements

Van de requirements besproken in bijlage A, zullen we hier hier een overzicht weergeven welke van deze requirements we wel en niet gehaald hebben. De nummers komen overeen met de nummers van de requirements zoals ze bijlage A staan.

Requirement	Resultaat	Commentaar
1	Gehaald	
2	Gehaald	
3	Niet gehaald	Alleen mogelijk om berichten te zien als op het envelopje rechtsonder wordt geklikt omdat dit zo in het design zat.
4	Niet gehaald	Deze eis is bijgesteld. Infoberichten komen nu op de hoofdpagina.
5	Niet gehaald	Deze eis is vervallen omdat gevonden werd dat deze functionaliteit niks toevoegde en de Journal alleen onoverzichtelijker zou maken
6	Gehaald	
7	Gehaald	
8	Niet gehaald	Gesponsorde berichten zitten nog niet in het systeem en vraag- en aanbod kunnen niet efficiënt uit het systeem opgehaald worden
9	zie 8	
10	Niet gehaald	Afbeeldingen zitten nog niet in het systeem.
11	Niet gehaald	Afbeeldingen zitten nog niet in het systeem.
12	Gehaald	
13	Gehaald	
14	Gehaald	
15	Niet gehaald	Wel gehaald bij een kleine hoeveelheid berichten, omdat preprocessing niet mocht plaatsvinden

16	Niet gehaald	Internet Explorer wordt niet ondersteund omdat CHAINels dit zelf niet ondersteunt. Ook mobiele browsers werken niet, omdat deze een aparte versie van de website nodig hebben.

6.3.2 Niet-Functionele requirements

Requirement	Resultaat	Commentaar
1	Gehaald	
2	Gehaald	
3	Gehaald	
4	Gehaald	

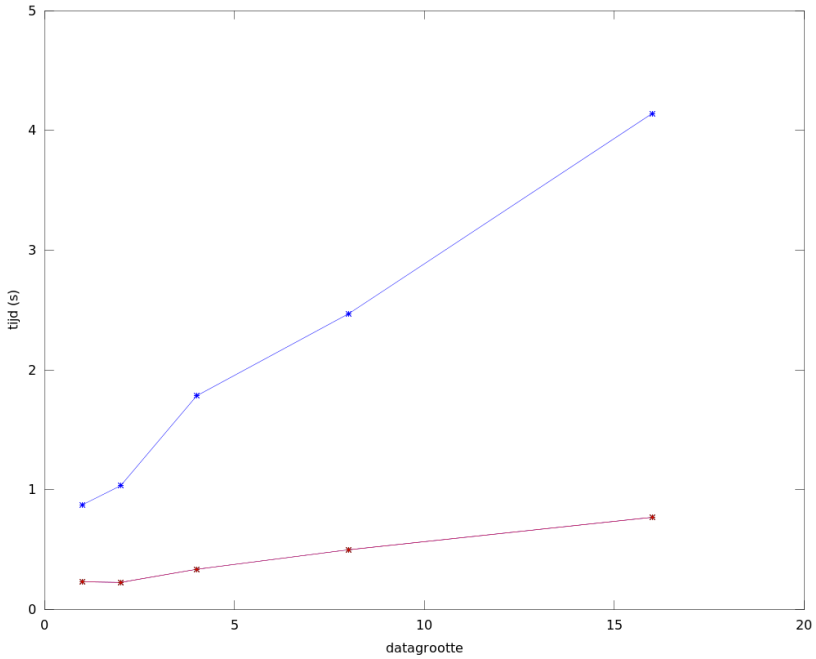
6.4 Profiling en benchmarking

Het testen van grote hoeveelheden data, kon gedaan worden door hetzelfde scriptje voor het genereren aan te roepen met andere (grotere) parameters. Om de performance te analyseren hebben we gebruik gemaakt van de profiler xdebug. Hier hebben we veel aan gehad. Zo hebben we een fout in de cache ontdekt waardoor er veel extra calls werden gemaakt naar methodes waardoor de snelheid van het algoritme afnam. Verder hebben we ook een benchmarkklasse gemaakt die de precieze tijdsduur meet van ons algoritme. In figuur 6.1 is een grafiek te zien van de snelheid van ons algoritme bij verschillende grootte data.

De verticale as van de grafiek is de tijd en geeft aan hoe lang het aanbevelingsalgoritme doet over het maken van aanbevelingen. De horizontale as geeft de grootte van de dataset aan.

Dataset	Aantal bedrijven	Aantal berichten	Aantal reviews	Looptijd zonder preprocessing	Looptijd met preprocessing
1	25	100	500	0.8751764297	0.2317234039
2	50	200	1000	1.0340203285	0.2264570236
3	100	400	2000	1.7841028214	0.3359915733
4	200	800	4000	2.4689182281	0.4986436367
5	400	1600	8000	4.1413439751	0.7699187756

De groottes van de data die we hebben gebruikt zijn te vinden in bovenstaande tabel. In De keuze voor deze verhouding van de dataset komt door de informatie die we van Erwin hebben gehad over de frequentie van het plaatsen



Figuur 6.1: De performance van het algoritme. Blauw is de looptijd van het algoritme wanneer er geen pre-processing plaatsvindt en rood als er wel pre-processing plaats zou vinden

van berichten in het oude systeem door bedrijven. Om de snelheid te testen pakken we iedere keer hetzelfde bedrijf. Dit is een speciaal bedrijf, want dit bedrijf is gechained met alle bedrijven in de dataset en dus ziten alle berichten in zijn netwerk. In de grafiek zelf zijn twee lijnen te zien; de blauwe lijn geeft aan hoe lang het algoritme duurt wanneer die voor het eerst wordt aangeroepen. De rode lijn geeft aan hoe lang het algoritme duurt direct nadat het algoritme al een keer is uitgevoerd. Nu duurt het algoritme minder lang, omdat al veel informatie in de cache aanwezig is. De vergelijking tussen de blauwe en rode lijn wilden we maken om te laten zien wat er nog mogelijk is wanneer er pre-processing wordt toegepast. Natuurlijk zal de tijdsduur met behulp van pre-processing niet gelijk worden met de rode lijn, maar het zal zeker dicht in de buurt kunnen komen.

7. Proces

We hebben dit project met zijn drieën gedaan. Behalve met elkaar hebben we ook samengewerkt met de werkgevers bij CHAINels en met de industriële vormgever. Bij dit project moesten we daarom met meer mensen samenwerken dan we bij voorgaande projecten die we moesten doen. De organisatie voelde toch redelijk kleinschalig aan en de communicatie verliep ook erg goed.

De eerste twee weken hebben we (zoals verplicht)gewerkt aan het oriëntatieverslag en plan van aanpak. Ook hadden we vergaderingen met het CHAINels-team om elkaar voor te stellen. Omdat we toen nog niet zo goed bekend waren met het concept van CHAINels en een aantal beslissingen nog niet vast stonden, waren er enkele onduidelijkheden waar we mee om moesten gaan tijdens het maken van de verslagen. Deze onduidelijkheden waren bijvoorbeeld hoe het uiteindelijke design eruit moest zien van de Journal en welke soort berichten in de Journal terecht moesten komen.

De derde en vierde week hebben we gewerkt aan de requirements specificaties en design specificaties. Ook zijn we begonnen aan het schrijven van code voor het uitschuiven van de Journal. De requirement specificaties hebben we met zijn allen gewerkt. De design specificaties zijn vooral door Benny en Laurent gemaakt, terwijl Jeroen aan het uitschuiven van de Journal aan het werk was. De keuzes voor de algoritmes de we in het verlag beschrijven hebben we met z'n allen gemaakt.

In de vijfde en zesde week zijn we begonnen met de implementatie van ons algoritme en het afmaken van het design. De algoritmes zoals we die toen hadden waren voor een groot deel geïmplementeerd, maar niet getest. Ook is er nagedacht over hoe de gebruiker met de Journal zal gaan werken en welke interface en knoppen en functies hij of zijn tot zijn of haar beschikking zou moeten hebben. Omdat het voor Willem toen niet helemaal duidelijk hoe het design eruit moest gaan zien en welke functionaliteiten de Journal heeft, hebben we een document geschreven met daarin de verschillende functionaliteiten en scenario's die vertellen hoe de gebruiker om kan gaan met de Journal. Dit document hebben we in samenwerking met Erwin gemaakt.

In de vijfde week hebben we de algoritmes zoals we ze toen bedacht hadden afgekregen en er tests voor geschreven. We hebben ons algoritme ook naar Sig gestuurd en ontvingen 4 sterren voor het resultaat. Het enige dat niet goed was aan de code, was dat we soms te lange methodes gebruiken en dat er geen duidelijk componentenstructuur aanwezig was.

In de zesde week besloten we het een en ander aan onze collaborative algoritme te veranderen. Alhoewel de algoritmes stabiel waren, vielen de resultaten van ons collaborative ons toch tegen. In plaats van alleen userbased willen we ons algoritme ook itembased maken, want uit papers bleek dat deze betrouwbaarder en efficiënter werkt dan userbased algoritmes.

In de laatste twee weken hebben we ons algoritme helemaal afgemaakt, getest en geoptimaliseerd.

7.1 Persoonlijke reflecties

Nu volgen korte reflecties van onszelf op het proces en hoe dit project hebben ervaren:

Benny

> De laatste maanden heb ik met veel plezier aan het project bij CHAINels gewerkt. De werksfeer was goed. Toen ik aan het project begon, had ik zelf helemaal geen ervaring met php, maar als snel bleek dit geen probleem te zijn, dit kwam onder andere door de ervaring in andere programeertalen. Ook was het een voordeel dat er al een framework klaar lag om de Journal in te bouwen, hierdoor was het makkelijk opstarten. Echter, het verminderde ook voor een gedeelte de vrijheid om bepaalde dingen te implementeren of bepaalde data op te vragen uit de database. Dat was soms wel jammer, omdat daardoor sommige optimalisaties niet konden worden doorgevoerd in het algoritme. Mijn tijd tijdens het project heb ik vooral aan het aanbevelingsalgoritme besteed, dat had als gevolg dat ik ook heel veel literatuuronderzoek moest doen. Zelf heb ik af en toe het idee dat ik bijna de helft van de tijd met mijn neus in de papers zat om een goed efficiënt algoitme te ontwikkelen. Na een tijdje vond ik dit toch wat minder leuk worden. Zeker, omdat de meeste papers lastig te begrijpen zijn, zoals bijvoorbeeld de papers over matrix factorization. Daarnaast had je ook papers die je wel begreep, maar dan bleek weer dat het niet helemaal toepasbaar was voor ons systeem. Ten slotte wil ik zeggen dat ik heel erg fijn heb samengewerkt met Jeroen en Laurent. Zelf heb ik niet het idee gehad dat er problemen waren in de samenwerking en verliep de communicatie meestal goed.

Laurent

> Ik vond dit project erg leuk om te doen en heb veel geleerd. Ik was gemotiveerd om dit project te doen omdat internetapplicaties en sociale netwerken een steeds groter wordende markt zijn. Tijdens mijn studie heb ik veel geleerd over software, maar heb ik toch weinig ervaring met webapplicaties opgedaan, terwijl dit toch een grote markt is. Toch kwam ik er tijdens het project achter dat de meeste

principes die we geleerd hebben bij deze studie ook toepasbaar zijn op het web.

Ik heb tijdens dit project vooral samen met Benny aan het aanbevelingsalgoritme gewerkt. We hebben daarvoor algoritmes bestudeerd uit papers en deze algoritmes zo goed mogelijk proberen aan te passen aan onze situatie. Ook hebben we moeten werken met profilers om het algoritme snel te krijgen.

Ik denk dat de samenwerking tussen iedereen, zowel binnen ons team, als tussen ons team en het bedrijf CHAINels erg vlot is verlopen. Wat daarbij waarschijnlijk een rol speelde is dat veel van de mensen die bij CHAINels werken zelf ook nog aan TU Delft studeren.

Jeroen

Ik was vanaf het moment dat Erwin en Vincent ons vertelden over dit project erg enthousiast en gemotiveerd. Ik werk zelf al een aantal jaar als webdeveloper en dus trok dit project onmiddellijk mijn aandacht, omdat het een aantal interessante technieken gebruikt ter optimalisatie van het systeem (zoals BigPipe). Mij werd verteld dat dit op het gebied van webprogrammatuur zo'n beetje het hoogste van het hoogste was en dat sprak mij heel erg aan. Ik heb uiteindelijk niet bijzonder veel nieuws geleerd op het gebied van programmeren zelf, maar wat wel leerzaam was, was het werken aan een systeem dat zo groots was opgezet. CHAINels heeft vanaf begin af aan rekening gehouden met een gigantische groei, zodat hun website ook om zou kunnen gaan met een veel grotere groep gebruikers wanneer het bedrijf eenmaal op gang was gekomen.

Tijdens dit project heb ik me voor het grootste gedeelte met de front-end bezig gehouden, omdat ik al ruime ervaring had met het maken van websites en de talen die daarbij komen kijken, zoals HTML, CSS, JavaScript en PHP. Ik heb de lay-out en de functionaliteiten zoals we die hebben bedacht geïmplementeerd en heb de JavaScript geschreven waardoor de gebruiker kan interacteren met de Journal. Ook heb ik geholpen aan de koppeling tussen de back-end en de front-end zodat de berichten die door het algoritme gevonden waren, efficiënt konden worden weergegeven.

De samenwerking met Benny en Laurent vond ik erg goed gaan; we kunnen het erg goed met elkaar vinden dus we hadden tijdens het project ook ontzettend veel plezier met elkaar. We konden ook efficiënt met elkaar samenwerken doordat we al een beetje een gevoel voor de werkwijze van elkaar hadden ontwikkeld. Verder vond ik het contact en de communicatie tussen met leden van het CHAINelsteam (voornamelijk met Erwin en Vincent) ook uitstekend verlopen. Het was altijd mogelijk om vragen te stellen of opmerkingen te maken en vaak werd daar dan ook snel op gereageerd.

8. Conclusie

Het doel van ons project is om een Journal te maken waarin op een overzichtelijke en prettig manier de meest relevante berichten uit het netwerk van een bedrijf getoond worden. Naar ons idee is dit redelijk goed gelukt, want als we kijken naar de requirements dan hebben wij daarvan de meeste behaald. De requirements die we niet hebben gehaald, kwam meestal doordat CHAINels het zelf niet meer wilde of dat het design veranderde. De tijd van het algoritme is naar ons idee net wat te langzaam, maar dit heeft als voornaamste reden dat er geen pre-processing plaats vindt. Wel denken wij dat de accuratie van het algoritme naar behoren is, omdat je vaak echt goed kan zien hoe een bericht belangrijker wordt wanneer je het bijvoorbeeld gaat volgen. Helaas, hebben we niet kunnen testen met echte bedrijven en kunnen we dus niet zeggen of de bedrijven echt behoefte hebben aan dit aanbevelingsalgoritme. Gelukkig, zijn er nog wel vele mogelijkheden om het huidige algoritme aan te passen en te verbeteren.

Ook denken we dat we een goed te onderhouden product hebben afgeleverd, aangezien SIG vier sterren gaf voor onze code en bovendien hebben wij de structuur van ons systeem zoveel mogelijk afgestemd op die van CHAINels. Daarnaast wordt heel de back-end goed getest en zijn deze tests overzichtelijk opgebouwd. Waarbij we uiteraard het test framework gebruiken dat CHAINels ook gebruikt voor hun unit tests. Bovendien is hierdoor naar ons idee ook een heel erg stabiel systeem afgeleverd wat echt een meerwaarde is voor CHAINels.

Het groepsproces en de samenwerking binnen de groep is ook goed verlopen. Afspraken werden eigenlijk altijd nagekomen en als er problemen waren met betrekking tot de implementatie of iets anders dan werd dat altijd goed besproken.

Kortom wij denken dat dit aanbevelingsalgoritme echt iets toevoegt aan CHAINels en hopelijk denken de bedrijven die op CHAINels gaan, hier hetzelfde over.

9. Aanbevelingen

In dit hoofdstuk doen we aanbevelingen die ons systeem nog verder zouden kunnen verbeteren. We beginnen met aanbevelingen voor ons aanbevelingsalgoritme waar we kijken hoe de snelheid en accuraatheid van ons algoritme omhoog kan. Vervolgens sluiten we af met aanbevelingen voor de GUI van de Journal.

9.1 Aanbevelingsalgoritme

Het aanbevelingsalgoritme kan nog verder verbeterd worden qua snelheid en qua accuraatheid. Deze verbetering hebben vooral betrekking tot het collaborative filter algoritme. Dit algoritme neemt veruit de meeste tijd in beslag en daarnaast zijn er nog kleine verbeteringen mogelijk voor dit algoritme. Eerst zullen we de verbeteringen qua snelheid bespreken en vervolgens de verbeteringen voor de accuraatheid.

9.1.1 Snelheid

Op dit moment maakt het algoritme geen gebruik van pre-processing. Dit heeft tot gevolg dat wanneer het aanbevelingsalgoritme wordt aangeroepen alle gelijkheden tussen de verschillende gebruikers en berichten opnieuw berekend moet worden. Dit kost onnodig veel tijd waardoor het algoritme zeer wordt vertraagd en daarom raden we pre-processing zeker aan, ook al moet er wat meer data opgeslagen worden. We zullen nu enkele voorbeelden geven van hoe de pre-processing mogelijk zou kunnen worden gemaakt met betrekking tot ons algoritme:

- Als eerste zou het gebruikersprofiel dat bij het content-based algoritme wordt berekend veel sneller kunnen worden opgezet. Dit is mogelijk wanneer de reviewverdeling en de berichtenverdeling automatisch zouden worden bijgewerkt in de database. Dit is geen zware taak voor de database, omdat het slechts bij elke gemaakte review van de gebruiker twee getallen hoeft bij te stellen. Ditzelfde geldt voor de berichtenverdeling, want die hoeft slechts te worden vernieuwd als er een bericht in het netwerk van de gebruiker wordt toegevoegd.

  Bovendien kan aan de hand van dit gebruikersprofiel ook het collaborative filter worden versneld, want in plaats van dat je telkens tussen twee

gebruikers de gemeenschappelijke berichten moet zien te vinden, pak je hun gebruikersprofiel. Doordat het gebruikersprofiel in constante tijd kan worden opgevraagd, is dit veel sneller dan door een lijst van reviews heen te lopen. Daarnaast wordt ook het datasparsiteit probleem verholpen, doordat je nu twee gehele gebruikersprofielen vergelijkt.

- Ten tweede kan er gebruik worden gemaakt van clustering [18]. In dat geval wordt in de pre-proces fase elke gebruiker in een bepaalde cluster gezet. Hierdoor is automatisch al bepaald welke gelijksoortige gebruikers een gebruiker heeft en dat scheelt in het zoeken van gelijksoortige gebruikers.

- Niet alleen het collaborative filter kan sneller door pre-processing, maar ook het opvragen van berichten uit je netwerk. Op dit moment moet je voor elke chain apart de berichten opvragen die hij heeft gepost. Dit is onnodig inefficiënt. Veel sneller zou zijn om in één keer alle berichten uit je netwerk op te vragen. Dit zou kunnen worden bewerkstelligd door apart in de database voor elke gebruiker de berichten op te slaan uit zijn netwerk.

Om de snelheid te verbeteren zijn dit onze drie belangrijkste aanbevelingen. Deze aanbevelingen vereisen natuurlijk flink wat opslag in de database, maar het algoritme zal bij grotere hoeveelheden data veel sneller werken. Onder andere doordat de aanroepen naar de database tijds het draaien van het algoritme sterk worden verminderd. Dat is ook de grootste bottleneck op dit moment in ons algoritme.

9.1.2 Accuraatheid

De gewichten die gebruikt worden bij het collaborative filter om de subalgoritmes te combineren kunnen automatisch worden bepaald [19]. Op dit moment zijn deze constant, maar uit onderzoek blijkt dat wanneer deze gewichten zich kunnen aanpassen aan de situatie de accuraatheid stijgt [10][19]. Het berekenen van deze gewichten moet wel via pre-processing worden gedaan, omdat het tijdsintensieve methodes zijn die daarvoor gebruikt worden.

Om te kunnen bepalen of het huidige algoritme zoals opgezet in dit verslag daadwerkelijk iets toevoegt aan de Journal, moet er een vergelijking gemaakt worden met een situatie zonder algoritme. Hiervoor zou een simpel onderzoeksdesign opgezet kunnen worden met twee condities: ëën Journal met het huidige algoritme en ëën Journal zonder die fungeert als controleconditie (waarbij automatisch de meest recente berichten als eerste in de Journal verschijnen, in plaats van die zoals gemanipuleerd door het algoritme). Hierbij kan de populatie binnen een bedrijf random aan de twee condities toegewezen worden om de invloed van individuele eigenschappen op de eindresultaten te voorkomen. De variabele die gebruikt zal worden om de twee groepen vervolgens te vergelijken, is de klikactiviteit (een muisklik op een werkende link van de Journal; een bericht, in

andere woorden) van de proefpersonen. Dit is afgeleid uit een onderzoek van Google News [11], waarbij de klikactiviteit van de gebruikers valide genoeg bleek om de invloed van het algoritme te testen.

De onafhankelijke variabele in dit design is de aan- of afwezigheid van het algoritme (dichotoom) en de klikactiviteit fungeert als afhankelijke variabele (interval). Om de twee groepsgemiddelden te kunnen vergelijken en een mogelijk significant verschil vast te kunnen stellen, kan een variantie-analyse of ANOVA gebruikt worden. Hierbij wordt een $\alpha = 0.05$ gehanteerd als significantieniveau. Indien zo een onderzoek wordt uitgevoerd, wordt er verwacht dat de klikactiviteit significant hoger is in de algoritmeconditie dan in de niet-algoritmeconditie.

Vervolgens kan uit de resultaten van het hierboven beschreven onderzoek de conclusie getrokken worden of het algoritme significant toevoegt aan de Journal of niet. Aan de hand daarvan kan het algoritme eventueel verbeterd worden en accurater worden opgezet.

9.2 Extra functionaliteit

Om de gebruiker meer controle te geven zou er meer ruimte kunnen komen voor customizability. Zo zou een bedrijf de verschillende onderdelen van de Journal naar verschillende plaatsen kunnen slepen en nieuwe widgets toe kunnen voegen. Ook zou het bedrijf de Journal kunnen aanpassen naar zijn of haar huisstijl.

A. Requirements

In deze sectie zullen we de eisen (requirements) die we van tevoren aan ons eindproduct hebben gesteld bespreken, te beginnen met de functionele requirements. Dit zijn de requirements die aangeven wat een gebruiker allemaal moet kunnen met ons systeem. Vervolgens gaan we het hebben over de niet-functionele requirements. Dit zijn de eisen waaraan het systeem moet voldoen. Ten slotte maken we enkele use-cases van onze functionele requirements.

A.1 Functionele Requirements

1. De gebruiker moet de Journal open kunnen schuiven d.m.v. een kader links in het scherm.
2. De gebruiker moet previews van berichten kunnen lezen in de Journal.
3. De gebruiker moet op een bericht kunnen klikken om het gehele bericht te kunnen zien en lezen.
4. De gebruiker moet de keuze hebben om alleen bepaalde categorieën berichten (nieuws, events, info, etc...) te zien.
5. De gebruiker moet berichten kunnen zoeken in de Journal d.m.v. keywords.
6. De Journal moet, indien mogelijk, een mix van verschillende categorieën berichten tonen.
7. De Journal kan alleen nieuwsberichten, infoberichten en events van bevriende bedrijven tonen.
8. De Journal kan vraag- en aanbodberichten en gesponsorde berichten van alle bedrijven tonen.
9. De Journal moet gesponsorde berichten tonen aan een bepaald aantal gebruikers. Deze eis moet later nog gedetailleerd worden uitgewerkt, omdat de mensen binnen CHAINels op dit moment ook niet compleet duidelijk hebben hoe deze gesponsorde berichten in de Journal moeten worden afgehandeld.

10. De Journal moet, indien mogelijk, een mix van berichten met en zonder afbeeldingen tonen.

11. De Journal moet wanneer een artikel een afbeelding bevat deze tonen in de preview van het bericht. Als een artikel meerdere afbeeldingen bevat, moet alleen de eerste afbeelding worden getoond.

12. De Journal moet berichten met de hoogste relevantie voor de gebruiker vooraan in de Journal plaatsen.

13. De Journal moet berichten op een overzichtelijke manier aan de gebruiker weergeven, zodat de Journal het gevoel van een gedrukte krant nabootst.

14. De Journal moet de verschillende categorieën van elkaar kunnen onderscheiden, bijvoorbeeld d.m.v. kleuren.

15. De Journal moet in een zodanige tijd (¡ 1s) geladen worden, dat het niet nodig is om de gebruiker een wachtscherm te laten zien voor het laden van de Journal.

16. De Journal moet in ieder geval weergegeven kunnen worden in Internet Explorer 7 of hoger, Firefox 12 of hoger, Chrome 18 en Opera 12. Ook in Safari en mobiele browsers moet de Journal weergegeven kunnen worden.

A.2 Niet-functionele requirements

1. De gebruikersinterface moet aansluiten op het design van CHAINels.

2. Het systeem moet onderhoudbaar zijn, zodat de mensen bij CHAINels zonder problemen onze code kunnen aanpassen.

3. Het systeem moet testbaar zijn, zodat fouten in de code makkelijk gevonden kunnen worden.

4. Het systeem moet stabiel zijn en mag geen foutmeldingen geven.

A.3 Use-cases / Scenarios

Aan de hand van de eisen die we hierboven hebben genoemd, hebben we een aantal use cases opgesteld. Deze zullen we hieronder verder uitwerken.

UC-01	**Een gebruiker logt in**
Actor:	Gebruiker
Goals:	Inloggen

Preconditions:	
Steps	1. Klik op login 2. Het systeem roept het aanbevelingsalgoritme aan. 3. De Journal plaatst de aanbevolen berichten op de achtergrond. 4. Het systeem plaatst een balk links van het scherm om de Journal uit te schuiven.

UC-02	**Een gebruiker opent de krant**
Actor:	Gebruiker
Goals:	De krant bekijken
Preconditions:	UC-01
Steps	1. De gebruiker drukt op de balk aan de linkerkant van de website. 2. Het systeem schuift de Journal uit.

UC-03	**Een gebruiker sluit de krant**
Actor:	Gebruiker
Goals:	De krant laten verdwijnen
Preconditions:	UC-02

Steps:	1. De gebruiker drukt op de balk aan de rechterkant van de Journal. 2. De Journal klapt in en is niet langer zichtbaar. Wel blijft de balk links van het scherm zichtbaar.

UC-04	**Een user selecteert een categorie**
Actor:	Gebruiker
Goals:	Berichten van een categorie bekijken
Preconditions:	UC-02
Steps:	1. De gebruiker klikt op de categorie. 2. Het systeem laadt de berichten uit de gekozen categorie.

UC-05	**Een user volgt een bericht**
Actor:	Gebruiker
Goals:	Een bericht volgen
Preconditions:	UC-04
Steps:	1. De gebruiker sleept het bericht naar de kolom met gevolgde berichten. 2. Het systeem voegt het bericht toe aan de kolom met gevolgde berichten en markeert het bericht als gevolgd.

B. Opdrachtomschrijving

Via onze medestudenten Erwin Buckers en Vincent Koeman zijn wij in aanraking gekomen met CHAINels, een project waar met een klein groepje mensen aan wordt gewerkt. Erwin en Vincent hebben CHAINels als volgt beschreven toen zij in de zomer van 2011 begonnen aan hun bachelorproject bij dit bedrijf:

> "Uiterst kort is CHAINels te beschrijven als 'de Facebook voor bedrijven'. In meer detail is het een professionele netwerktool die bedrijven en organisaties met andere bedrijven en organisaties verbindt (chains). Chainels kan worden gebruikt om zakelijk contact te houden met relevante relaties. Het 'key-concept' is dat het puur op bedrijven gefocust is; het geheel moet dus een hoog professionaliteitsgehalte hebben. Dit is dan ook de onderscheiding tussen Chainels en al bestaande netwerken als Facebook, Twitter, LinkedIn, enzovoorts: het is bedoeld voor pure bussiness-to-business networking. Met dit oogpunt is de naam Chainels ook te verklaren: het belangrijkste voor een bedrijf is zijn 'chain' en zijn 'channels'. Bedrijven houden via Chainels professioneel contact met hun relaties en kunnen vraag en aanbod met betrekking tot diensten en producten onbeperkt 'uploaden' en meer leren over bedrijven waarmee ze zaken doen."

Het plan is dat CHAINels binnenkort online gaat. Het doel is om uiteindelijk miljoenen gebruikers te kunnen ondersteunen, dus bij het opzetten van de website moet al goed nagedacht worden over optimalisatietechnieken zodat alles zo snel mogelijk verloopt. Er zijn nog verschillende functionaliteiten die voor de launch geïmplementeerd moeten worden. De belangrijkste onderdelen die nog geïmplementeerd moeten worden, en die voor ons interessant zouden kunnen zijn, zijn onder andere de verschillende soort algoritmes: zoek-algoritme, chain suggesties, vraag-aanbod algoritmes en de verschillende algoritmes die nodig zijn voor een goed werkende Journal. Daarnaast zijn er ook nog onderdelen zoals het verkrijgen verwerken van verschillende statistieken van een klant. Het beter inzicht kregen van ieder z'n bedrijfsnetwerk door middel van graven.

Tijdens ons bachelorproject zullen wij ons richten op de Journal van CHAINels. Wat is de Journal? De Journal is een soort krant die iedere gebruiker via z'n homepage kan openen. In de Journal kunnen berichten worden gelezen

van (bevriende) bedrijven. Ook kunnen er gesponsorde resultaten in voorkomen. Aangezien er maar beperkte plaats voor verschillende berichten is (zie het plaatje dat is bijgevoegd), moeten alleen de meest relevante berichten worden getoond voor het bedrijf. Ook moet dit systeem efficiënt blijven werken voor miljoenen gebruikers. Dus wat er moet nu allemaal gedaan worden voor de Journal om het goed werkend te krijgen? Ten eerste moeten de relevantie en de prioriteit van de berichten worden bepaald. Aangezien in een miljoenen netwerk, de Journal met veel berichten te maken zal krijgen, is het belangrijk dat dit efficiënt en snel gebeurd. Ten tweede moet er ook rekening worden gehouden met verschillende soorten berichten zoals nieuwsberichten, events, demand, supply en info. Daarnaast moet er ook worden gekeken hoe de berichten in de Journal moeten worden geplaatst, dus in welke volgorde en welke plaats. Hierbij moet ook rekening worden gehouden met de grootte van de verschillende berichten.

Ten derde moet er in de Journal gefilterd kunnen worden op categorieën, bijvoorbeeld op soort bericht of per industrie. Daarnaast zou je ook moeten kunnen zoeken op keywords in de Journal zelf. Ook moet er gekeken in hoeverre tijdsgevoeligheid een rol moet spelen op de Journal en z'n inhoud. Wat moet er bijvoorbeeld met de berichten gebeuren wanneer je twee keer op dezelfde dag kijkt in de Journal? Moeten de berichten steeds veranderen of hetzelfde blijven of gedeeltelijk veranderen?

Ten slotte moeten er ook bepaalde functies toegevoegd worden die toepasbaar zijn op de berichten zelf. Bijvoorbeeld een *stay-up-to-date* functie, waarmee

je een bericht in de gaten kunt houden. Deze moet in de Journal ergens terugkomen zodat je makkelijk de reacties kan volgen en dus het bericht in de gaten kan blijven houden. Daarnaast moeten berichten ook kunnen worden geopend en bijvoorbeeld notificaties worden gegeven wanneer er op een bericht wordt gereageerd dat jij interessant vindt. De vraag is natuurlijk hoe je die notificatie laat zien aan de gebruiker.

Kortom ons plan is dus om ons uitsluitend te richten op de Journal met alle functionaliteiten en algoritmes die daarbij horen. Er moet worden geschreven in object-georiënteerd PHP, omdat het direct wordt gebruikt voor in de website en deze is ook in PHP geschreven. Er wordt gebruik ook gemaakt van Redis, een NoSQL databasesysteem. Dit is dus de omgeving waarin wij zullen programmeren.

C. Oriëntatieverslag

C.1 Inleiding

Ons bachelorproject is specifiek gericht op de Journal van CHAINels. De Journal is een aparte pagina binnen CHAINels met een overzicht van allerlei berichten die interessant zijn voor een bedrijf. In ons oriëntatieverslag zullen we ons allereerst oriënteren op de technische aspecten die vereist zijn om aan dit project te werken, zoals de programmeertaal waarin CHAINels is geschreven. Vervolgens zullen we naar bestaande technieken kijken die nuttig kunnen zijn bij het ontwerpen van de Journal. Omdat een van de doelen van onze opdracht het schrijven van algoritmes die uit alle beschikbare data de beste berichten kunnen filteren om weer te geven op de Journal is, zullen we ook kijken naar wat voor algoritmes er al bestaan voor soortgelijke doeleinden om te beoordelen in hoeverre deze nuttig zijn voor ons project. Uiteindelijk zullen we de ontwikkelingsmethode die wij aan zullen houden bespreken.

C.2 Programmeertaal

CHAINels is geprogrammeerd in PHP, een van de meest populaire server-side programmeertalen voor websites [16]. Omdat CHAINels zo uitgebreid is en om het allemaal overzichtelijk te houden, wordt in CHAINels uitsluitend op een object-georiënteerde manier geprogrammeerd. Er is al een grote basis aan PHP classes voor het systeem waar wij gebruik van kunnen maken tijdens het programmeren van de Journal. Natuurlijk zijn niet alle classes voor ons even relevant, vandaar dat we in de eerste week van het project ook gelijk een groepsmeeting hebben gehad waarin ons werd uitgelegd wat elke PHP class inhield, wat er mee gedaan kon worden en in hoeverre wij er tijdens ons project gebruik van konden maken.

Van de mensen uit onze projectgroep hadden Benny en Laurent nog geen eerdere ervaring met PHP, maar op dat gebied verwachten we geen problemen. PHP is een van de makkelijkere programmeertalen, onder andere omdat alle functies uitgebreid gedocumenteerd zijn met talloze voorbeelden, maar ook door het grote aantal ingebouwde functies die in de taal zijn inbegrepen. Bovendien zijn alle concepten die we de afgelopen drie jaar tijdens de verschillende colleges over object-georiënteerd programmeren hebben geleerd heel goed toepasbaar op

PHP en is het dus alleen nodig om de syntax van de taal zelf onder de knie te krijgen tot we er allemaal goed mee overweg kunnen. Dit zal geen probleem worden, wederom dankzij de online documentatie. Daarbij komt ook dat we waarschijnlijk voor het grootste gedeelte gebruik kunnen maken van functies die al in het CHAINels project zitten, waardoor het programmeergedeelte al iets vergemakkelijkt zal zijn.

C.3 Database

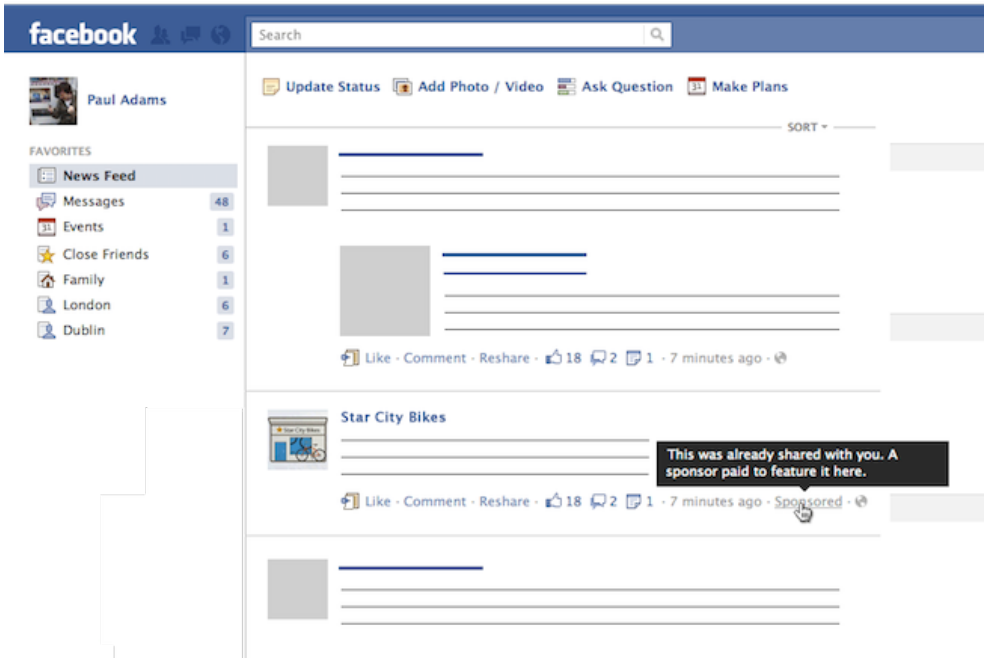
CHAINels gebruikt Redis, een NoSQL databasesysteem, om data op te slaan. Uiteindelijk wil men natuurlijk zo veel mogelijk bedrijven bij CHAINels betrekken en om met die ambitie rekening te houden, is nu al gekozen om een NoSQL database systeem te gebruiken in plaats van een relationeel database management systeem. Een NoSQL systeem is vaak een betere keuze als het om gigantische hoeveelheden data gaat; de populariteit van dit soort systemen is dan ook tegelijkertijd met de opkomst van grote internetbedrijven zoals Google en Facebook enorm gestegen.

Het voornaamste verschil tussen deze twee soorten systemen is dat NoSQL systemen specifiek ontworpen zijn om met grote hoeveelheden data om te gaan. Er wordt minder waarde gehecht aan run-time functionaliteit, waardoor operaties zoals JOINs en dergelijke ook niet mogelijk zijn. In plaats daarvan ligt de nadruk op de opslag van de data en biedt het systeem verder ook niet veel meer functionaliteit. Afhankelijk van het soort data dat opgeslagen moet worden, kan dit systeem dus veel voordelen hebben.

Wij zullen ons bij het programmeren van de Journal echter niet zo heel erg veel bezig moeten houden met het achterliggende databasesysteem. Zoals het bestaande deel van CHAINels nu is ontworpen, bestaan er voor het ophalen van data uit de database al allerlei PHP-functies die aangeroepen kunnen worden. In de code die wij zullen gaan schrijven, zullen we dus niet direct contact hoeven te maken met de database, maar kunnen we daarvoor al bestaande PHP-functies gebruiken. Mochten we data nodig hebben die nog niet opgehaald kan worden via een van de PHP-functies uit het huidige systeem, dan kunnen we dat aangeven bij een van de begeleiders met wiens samenwerking we die benodigde functionaliteit dan alsnog in zullen bouwen.

C.4 Vergelijkbare producten

n deze sectie gaan we kijken naar vergelijkbare producten die gebruiken maken van een news feed, dus hoe worden bijvoorbeeld status-updates getoond in sociale netwerken. Ook kijken we naar producten die op andere manieren informatie willen tonen aan de mensen, zoals nieuws berichten in kranten op het internet. Van al deze producten zullen we vooral kijken hoe verschillende informatie wordt getoond aan de gebruik. We zullen beginnen met Facebook als representatie voor sociale netwerken. Vervolgens bekijken we Flipboard, dit



Figuur C.1: News feed Facebook

is een applicatie voor tablets waar je nieuwsberichten op kan vinden en updates uit sociale netwerken. Ten slotte nemen we een kijkje in hoe online kranten eruit zien.

C.4.1 Facebook

Facebook is een sociale netwerk waar je verschillende soort informatie op kan zetten. Deze informatie kan je onder andere terug vinden via de news feed van het netwerk. In deze news feed kan je vooral nieuwe informatie vinden van bevriende gebruikers. De news feed is een rechte brede kolom en alle informatie staat onder elkaar. Wanneer je naar beneden scrolt, wordt de kolom automatisch groter, zodat je gewoon door kan blijven lezen. In Figuur 1 is een plaatje te zien van de newsfeed van Facebook.

C.4.2 Flipboard

Flipboard is een programma voor de IPhone die status updates van verschillende social media zoals Twitter, Facebook in een enkele applicatie toont. De informatie die het toont is afhankelijk van de feeds waar je jezelf op kunt abonneren. De gebruikersinterface probeert zo veel mogelijk om de ervaring van een krant na te bootsen. Zo kan er door de bladzijdes gebladerd worden met behulp

van gebaren op een touchscreen en wordt dit ook geanimeerd als het omslaan van een pagina.

De beginpagina van flipboard heeft de naam cover stories. Hier worden de meest interessante en populaire verhalen van al je accounts bij elkaar gezet. De keuze wat er getoond moet worden, kan aangepast worden door middel van een content guide. Hier kan de gebruiker kiezen in welke categorieën hij of zij geïnteresseerd is. Als de categorieën gekozen zijn, kan de gebruiker via zijn content-guide de categorie kiezen waarvan hij of zij de artikelen wil lezen. In figuur 2 zijn de cover stories en content guide te zien [13].

C.4.3 Online kranten

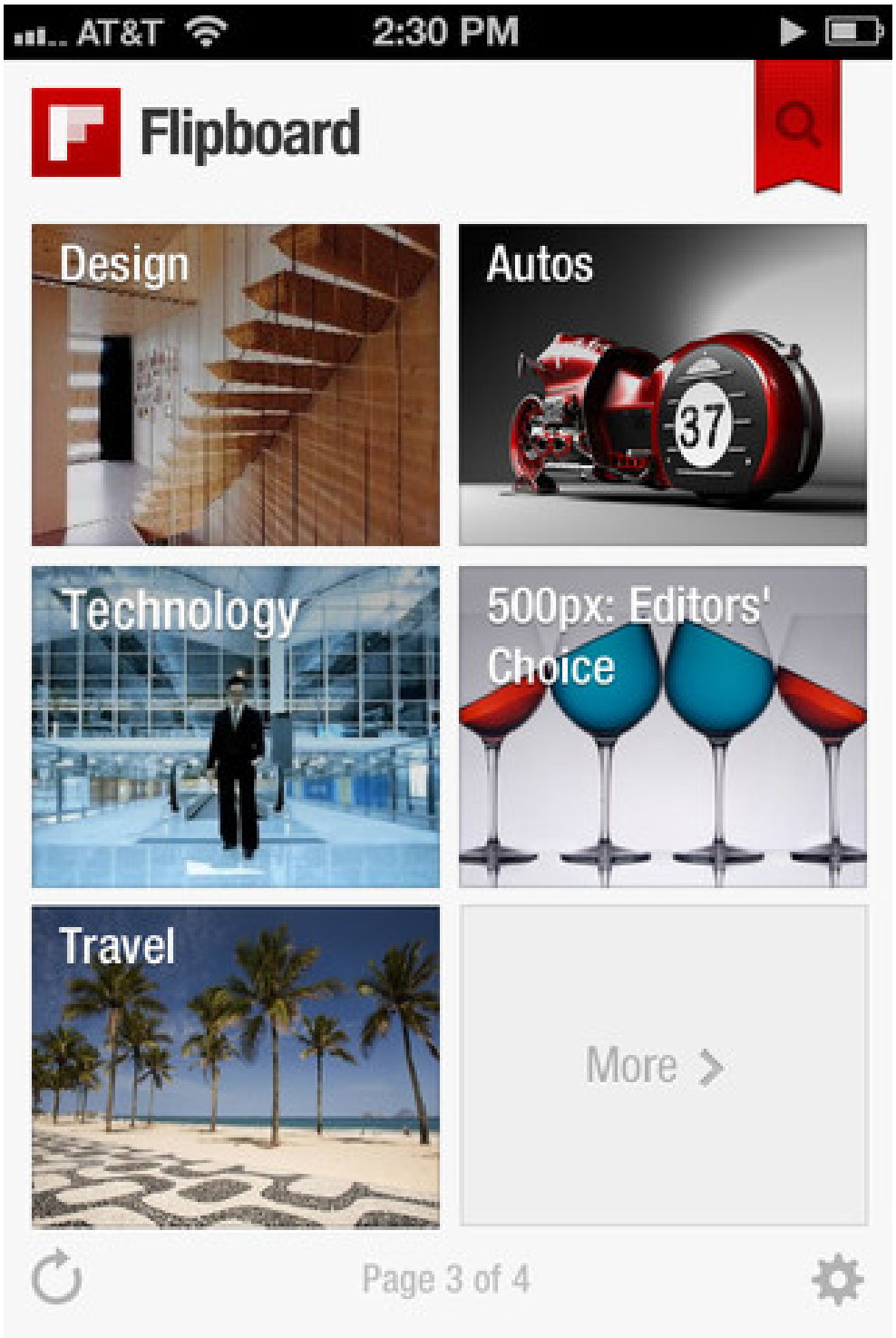
De uiteindelijke indeling van de Journal ligt nog heel open en wij zullen tijdens ons project ook nadenken over wat de beste verdeling van de verschillende soorten berichten over de Journal is. Een van de ideeën tot nu toe is wel om het zoveel mogelijk als een krant aan te laten voelen, zodat gebruikers het ook echt als een krant behandelen. Daarom hebben we onder andere naar de website van de New York Times gekeken [17]. Naar onze mening weet deze website heel goed het gevoel van een krant op een digitale manier over te brengen. Hieronder is een afbeelding te zien van een deel van de homepagina op 1 mei 2012:

Wat ons hier vooral aansprak, was het gebruik van een foto op de homepagina met daaromheen teksten van andere artikelen, iets wat ook vaak genoeg in gedrukte kranten te zien is. Zo'n soort indeling zou volgens ons ook goed bij de Journal van CHAINels passen. De vraag blijft dan nog hoe we bepalen van welk bericht precies een foto wordt geplaatst en meer van dat soort zaken, maar dat is iets wat we aan de hand van de algoritmes die we gaan schrijven op moeten lossen.

Naast de New York Times zijn er nog meer online kranten waaruit we inspiratie op kunnen doen, zoals bijvoorbeeld die van de Washington Post (http://www.washingtonpost.com). Aan de hand van dit soort websites kunnen we tijdens het verloop van het project alvast wat ideeën opdoen over het ontwerp van de Journal, die we vervolgens tijdens de groepsbesprekingen met de begeleiders aan kunnen voeren. We hebben wel gezamenlijk besloten om prioriteit te geven aan de algoritmes zelf en pas daarna tot een definitief ontwerp te komen, maar dat neemt niet weg dat tussentijdse ideeën altijd welkom zijn.

C.5 Algoritmes

In deze sectie gaan we kijken wat voor soort algoritmes we zouden kunnen gebruiken om ervoor te zorgen dat de berichten met de hoogste prioriteit ook vooraan in de Journal komen te staan. In de Journal komen verschillende soorten berichten voor; nieuwsberichten, events, infoberichten en vraag/aanbodberichten. Bij nieuwsberichten en events gaat het vooral om de inhoud van tekst en in mindere mate het bedrijf die het bericht plaatst. Bij infoberichten wordt de prioriteit compleet door het bedrijf of de bedrijven die in het infobericht voor-



Figuur C.2: Links cover stories en rechts content guide

komen bepaald. Bij vraag en aanbod gaat het echt om het verkopen en kopen van een dienst of product. Dus moet er berekend worden hoe relevant een product is voor een bedrijf. Ook is het belangrijk dat er rekening wordt gehouden met diensten of producten die het bedrijf zelf wilt hebben. Waarschijnlijk zullen we voor deze groepen verschillende soorten algoritmen moeten gebruiken.

C.5.1 Recommender sytems algoritmes

Wat voor soort algoritmes zouden dit kunnen bewerkstelligen? Al snel kwamen we uit op recommender systems algoritmen, dit zijn algoritmen die de prioriteit berekenen van elk item en dit teruggeven aan de gebruiker [3]. Echter, de recommender sytems bestaat uit een hele grote groep algoritmen, daarom zullen we eerst de verschillende categorieën bekijken waarin we recommender systems kunnen onderverdelen. Daarna zullen we verschillende algoritmen bekijken die we voor onze Journal zouden kunnen gebruiken en benoemen we de voor- en nadelen.

Eerst gaan we dus de verschillende categorieën bekijken waaruit recommender systemen bestaan [4]. We hebben de volgende categorieën: collaborative filtering waarin we ook demographic en community-based bespreken. Daarnaast bekijken we ook content- based met daarin ook knowlegde-based systemen. Ten slotte zeggen we iets over hybride recommender systemen.

C.5.2 Collaborative filtering

Bij collaborative filtering wordt er gekeken naar wat gelijksoortige gebruikers waarderen [4]. Gelijksoortige gebruikers worden gevonden door te kijken welke artikelen de gebruikers eerder waardeerden. Nieuwe artikelen die door gelijk-

soortige gebruikers gewaardeerd werden, zullen dan als aanbevelingen gedaan worden.

Voordelen van collaborative filtering zijn onder andere dat het artikelen kan adviseren uit verschillende categorieën [4,5,6]. Ook omdat de techniek compleet vertrouwd op gelijksoortige gebruikers, is er geen domeinkennis nodig over de artikelen. Ook wordt er niet alleen naar het onderwerp gekeken wordt, maar ook naar de waardering. Omdat het systeem de smaak van de gebruiker leert kennen, wordt de kwaliteit van de suggesties ook steeds beter en is het algoritme erg schaalbaar. Hoe langer het algoritme werkt, hoe beter het wordt. Zo kan er impliciet feedback over de recommandaties gedaan worden.

Het nadeel is dat nieuwe items vaak nog geen rating hebben (het cold start problem). Ook hebben nieuwe gebruikers nog niet veel artikelen beoordeeld. Daarom is dit systeem niet zo geschikt voor nieuwe gebruikers of nieuwe artikelen. Verder is het voor dit algoritme nodig dat er veel gebruikers zijn en dat deze gebruikers erg actief zijn. Een voorbeeld van een collaborative filter is het "k Nearest Neighbors" algoritme [7]. Dit algoritme identificeert paren van mensen die bijvoorbeeld ongeveer hetzelfde koop-historie hebben. Op deze manier zijn de dichtstbijzijnde buren bepaald (=k) en vervolgens wordt er gekeken voor items die je nog niet heb bekeken of een score hebt gegeven hoe relevant die voor jou zijn op basis van de ratings van je dichtstbijzijnde buren die wel een rating hebben gegeven.

C.5.3 Community-based

Community-based filtering is vergelijkbaar met collaborative filtering [4]. Alleen wordt er bij community-based filtering niet gezocht naar gelijksoortige gebruikers, maar wordt er gekeken naar de voorkeuren van de bedrijven waar je een chain mee hebt. Uit onderzoek blijkt dat mensen meer op aanbevelingen van bekenden vertrouwen dan op aanbevelingen van anonieme gelijksoortige mensen (zoals bij collaborative filtering). Omdat bedrijven zelf kunnen kiezen welke bedrijven ze uitnodigen voor chains, geeft deze methode bedrijven ook meer controle over de aanbevelingen die ze te zien zullen krijgen [15].

C.5.4 Demographic

Bij demographic filtering worden gelijksoortige gebruikers gevonden door naar geslacht, leeftijd, land van afkomst, etc. te kijken en in te delen in stereotypes [4,5,6]. Bij bedrijven zullen we vooral kijken naar de locaties van de vestigingen, de industrie en de grootte. Bij CHAINels kan deze informatie simpelweg uit het gebruikersprofiel afgelezen worden. Het voordeel van deze methode is dat het aanbevelingen kan doen, nog voordat de gebruiker eerdere artikelen heeft moeten beoordelen. Het algoritme is makkelijk op te zetten. Omdat bedrijven waarschijnlijk hun hele profiel zullen invullen is het niet heel moeilijk om aan de demografische informatie van bedrijven te komen. Verder heeft ook dit algoritme de mogelijkheid om te leren. De aanbevelingen van deze methode zijn echter niet heel erg op de persoon afgestemd in vergelijking met andere methoden en houdt

het bijna geen rekening met gebruikers met afwijkende voorkeuren. Verder heeft deze techniek ook last van het cold item probleem en is er veel data nodig.

C.5.5 Content-based

Content-based filtering systemen recommanderen items gebaseerd op beschrijvingen of content van de items [4]. Ook wordt er gebruik gemaakt van item-naar-item correlatie voor het geven van recommandatie. In deze systemen start het proces voor recommandaties door content data te verzamelen van de items. Bijvoorbeeld door van nieuwsberichten de titel, schrijver en andere informatie te verzamelen. Vervolgens wordt de gebruiker gevraagd om ratings te geven aan items. Ook kan er gebruik gemaakt worden van het gebruikersprofiel om daaruit de voorkeur voor berichten te bepalen. Ten slotte wordt er voor elke item een score bepaald door gebruik van het opgezette gebruikersprofiel en de beschrijving van het item dat je een score wilt geven. Uiteindelijk krijg je een lijst terug met scores voor elk item, waarbij de hoogste score de hoogste relevantie voor de gebruiker aangeeft.

Een voorbeeld van zo'n algoritme is het naïeve bayesiaane algoritme. Dit is een algoritme dat gebruikt kan worden om aan de hand van de woorden die in een tekst voorkomen deze tekst tot een van de categorieën in te delen. Zo kan er voor elke gebruiker de kans berekend worden dat het tot de interessante of oninteressante artikelen behoort. Voordelen van content-based algoritmen zijn onder andere dat er geen domein kennis nodig is, alhoewel dit niet altijd waar is, omdat je soms bepaalde concepten van te voren moet opstellen voor sommige algoritmen [4,5,6]. Daarnaast geldt ook bij content-based technieken dat hoe langer het algoritme werkt hoe beter het wordt en kan het impliciet feedback verwerken. Ook heeft het geen last van het cold item probleem, dus items die nog niet bekeken zijn kunnen meteen als suggestie worden voorgesteld aan de gebruikers.

Nadelen van content-based algoritmen zijn echter dat bij nieuwe gebruikers nog niet alle voorkeuren bekend zijn waardoor de recommandaties van mindere kwaliteit zijn. Ook zijn content-based algoritmen heel erg afhankelijk van hoe een item wordt beschreven, waardoor een item verkeerd herkend kan worden door het algoritme. Het algoritme kijkt verder alleen naar de inhoud van wat er beschreven wordt, maar niet naar het nut voor een gebruiker.

C.5.6 Knowledge-based

Bij knowledge-based systemen heeft het systeem kennis van de gebruiker en zoekt het systeem items die aansluiten bij de wensen van de gebruiker [4]. Zo kan, als de gebruiker aangeeft welke artikelen hij mag, het systeem deze als voorbeeld gebruiken om soortgelijke items te zoeken. Het systeem kan ook rekening houden met hoeveel geld de gebruiker wil besteden.

Voordelen van knowledge-based algoritmen zijn dat het geen problemen heeft met opstartproblemen, dus het maakt niet uit voor de prestatie van algoritme

wanneer er een nieuwe gebruiker is of een nieuw item. Daarnaast is deze techniek gevoelig voor verandering van gebruikers voorkeuren, dus deze techniek herkent gelijk wanneer een gebruiker een andere voorkeur en niet zoals bij bijvoorbeeld content-based dat heel het algoritme zich nog moet aanpassen. Ook kan het zoals net aangegeven gebruikers voorkeuren omzetten naar producten die worden aangeboden.

Nadelen van deze techniek is dat het niet kan leren uit het verleden, dus het zal altijd dezelfde fout maken en ook is domein kennis nodig om de techniek goed werkend te krijgen [15].

C.5.7 Hybrid recommender systems

Elk van de technieken hierboven heeft nadelen, vandaar dat een hybrid recommender systeem meerdere van de boven beschreven technieken combineert om zo de zwaktes van de ene techniek op te vangen door de andere. Het enige dat nog belangrijk is bij hybrid recommender systemen, is hoe je de uiteindelijke score bepaalt van het systeem. Dus hoe combineer je de verschillende technieken in je hybride systeem. Methodes die hiervoor bestaan noemen we hybridization methoden. In tabel 1 zullen we daarvan een overzicht geven [6,8].

Nu zullen we kort een uitleg geven bij elk Hybridization methode, zodat het duidelijk is wat er in de tabel bedoeld wordt. Bij de weighted methode krijgen de scores van de verschillende gebruikte technieken een zelfde gewicht. De scores worden bij elkaar genomen en de hoogste is dan de beste recommandatie.

Bij de switching methode maakt het hybride systeem gebruik van criteria waaruit het bepaalt welke recommandatie techniek het op een zeker moment gebruikt. Bijvoorbeeld wanneer een content-based systeem niet met zekerheid een goede recommandatie kan geven dan switcht het naar collaborative recommandatie om een goed resultaat te krijgen.

De mixed methode laat verschillende recommandatie technieken tegelijkertijd uitvoeren. De recommandaties die door beide technieken worden gemaakt, worden allebei getoond.

Bij feature combining leent het hybride systeem de recommandatie logica van een andere techniek in plaats dat het een aparte component wordt voor geïmplementeerd. Bijvoorbeeld een content-based recommandatie systeem werkt door het maken van geleerd model voor iedere gebruiker. Dit keer wordt echter de gebruikers rating data gecombineerd met de producteigenschappen. Dus dit systeem werkt op een content-based wijze, maar de content van het product is geassocieerd met collaborative recommandatie.

Het idee van cascade is erop gericht dat een zwakkere recommandatie systeem niet de recommandaties volledig kan overrulen gemaakt door een sterker systeem. Het zwakkere systeem kan alleen de resultaten verfijnen.

Feature augmenting lijkt erg veel op feauture combining, maar het verschil is dat in plaats van dat het ruwe eigenschappen uitbreidt aan het bestaande product, wordt er bij feature augmenting de resultaat van een berekening aan het product toegevoegd.

Hybridization methoden	beschrijving
Weighted	De scores van de verschillende recommandatie technieken worden met elkaar gecombineerd.
Switching	Het systeem switcht tussen de verschillende recommandatie technieken en welke die gebruikt hangt af van de situatie.
Mixed	De recommandaties van de verschillende technieken worden op hetzelfde moment getoond.
Feature Combination	Eigenschappen van verschillende recommandaties data bronnen worden gebruik als input voor één recommandatie algoritme.
Cascade	De ene techniek verfijnd de recommandaties voor de andere techniek.
Feature Augmentation	Output van de ene techniek wordt gebruikt als input eigenschap van de andere techniek.
Meta-level	Het model geleerd bij de ene techniek wordt gebruikt als input voor de andere techniek.

Tabel C.2: Hybridization methoden

Ten slotte bij meta-level wordt een model geleerd bij het ene recommandatie systeem volledig gebruikt als input voor het andere recommandatie systeem. Dus de origineel input wordt volledig vervangen en het gebruikte recommandatie systeem kan dan alleen zulke modellen als input krijgen en geen ruwe profiel data.

C.5.8 Samenvatting

In deze sectie zullen we een korte samenvatting geven van de verschillende recommandatie technieken. In tabel 2 is kort de werking van de verschillende recommandaties technieken samengevat. In de kolom achtergrond van deze tabel wordt verteld wat voor soort data er wordt gebruikt. In de kolom input welke data de technieken binnen krijgen. De kolom proces geeft aan hoe de data verwerkt wordt en bij voorbeelden staats soms een algoritme die voor deze techniek gebruikt wordt.

Techniek	Achtergrond	Input	Proces	Voorbeelden
Collaborative filtering	Waarderingen van alle gebruikers over de artikelen.	Waarderingen van een gebruiker u over de artikelen.	Zoek gebruikers die lijken op de gebruiker en gebruik hun voorkeuren om een schatting te geven van hoe interessant een artikel is voor gebruiker u.	k-Nearest Neighbors algoritme
Community-based	Waarderingen van alle gebruikers over de artikelen	De vrienden van de gebruiker waarvoor artikelen gevonden moeten worden.	Kijk naar de vrienden van de gebruiker en gebruik hun voorkeuren om een schatting te geven van hoe interessant een artikel is voor een gebruiker.	

Demografisch	Demografische informatie over alle gebruikers en hun waarderingen.	Demografische informatie een gebruiker u.	Vind gebruikers met dezelfde demografische informatie en gebruik hun voorkeuren om een schatting te geven van hoe interessant een artikel is voor gebruiker u	Zie collaborative filtering
Content-based	Eigenschappen van alle artikelen (bijvoorbeeld de woorden die erin voorkomen).	Waarderingen van een gebruiker u over de artikelen.	Maak een model van de waarderingen en roep het aan over de artikelen	Naïve bayesian
Knowledge-based	Eigenschappen van alle artikelen en hoe die zin hebben voor een gebruiker.	Een beschrijving van wat een gebruiker nodig heeft.	Zoek een match tussen de artikelen en wat de gebruiker nodig heeft.	Zie content-based

Tabel C.3: Overzicht verschillende recommendation algoritmen

- Beperkt zich niet tot een enkele categorie, maar kan naar meedere onderwerpen kijken.
- Geen domeinkennis nodig.
- Zelflerend. De recommendaties worden na verloop van tijd beter.
- Kijkt niet alleen naar de inhoud, maar ook naar of het artikel interessant gevonden wordt.
- Cold start voor nieuwe gebruikers en artikelen.
- Niet handig voor bedrijven met afwijkende voorkeuren.
- Veel data nodig.

Collaborative filtering	• Beperkt zich niet tot een enkele categorie, maar kan naar meedere onderwerpen kijken. • Geen domeinkennis nodig. • Zelflerend. De recommendaties worden na verloop van tijd beter. • Kijkt niet alleen naar de inhoud, maar ook naar of het artikel interessant gevonden wordt.	• Cold start voor nieuwe gebruikers en artikelen. • Niet handig voor bedrijven met afwijkende voorkeuren. • Veel data nodig. • X

Community-based	• Beperkt zich niet tot een enkele categorie. • Geen domeinkennis nodig. • Zelflerend. • Mensen vertrouwen hun vrienden meer dan onbekenden. • Kijkt niet alleen naar de inhoud, maar ook naar of het artikel interessant gevonden wordt. • Meer controle over waar de suggesties vandaan komen.	• Cold start voor nieuwe gebruikers en artikelen. • Niet handig voor gebruikers met afwijkende of unieke voorkeuren. • Veel data en gebruikers nodig.
Demografisch	• Beperkt zich niet tot een enkele categorie. • Geen domeinkennis nodig. • Zelflerend. • Nieuwe gebruikers die nog niets hebben beoordeeld kunnen ook geholpen worden. • Makkelijk te implementeren.	• Cold start voor nieuwe artikelen. • Niet handig voor bedrijven met afwijkende voorkeuren. • Veel data nodig. • Aanbevelingen zijn niet zo sterk op de persoon gericht dan bij andere technieken.

Content-based	• Geen domeinkennis nodig. • Zelflerend. • Geen cold start probleem voor nieuwe artikelen. • Kan een verklaring geven voor waarom een item aangeraden werd.	• Cold start voor nieuwe gebruikers. • Veel data nodig. • Kan alleen aanraden op basis van onderwerp, maar niet op basis van nut.
Knowledge-based	• Kan suggesties geven aan nieuwe gebruikers. • Erg gevoelig voor een verandering in voorkeuren. • Kijkt naar producteigenschappen.	• Kan niet leren. • Kennis over de artikelen moet verzameld worden.
Hybrid	• Compenseert nadelen van de andere algoritmes.	• Ingewikkelder systeem • Langere ontwikkelingstijd.

C.5.9 Voorbeelden Recommender Systems

News@Hand In dit project wordt een context-aware recommender systeem gebruikt voor nieuwswebsite News@Hand [9]. Dit systeem werkt door een query in te vullen en vervolgens gaat het algoritme zoeken naar een artikelen gebaseerd op jouw interesses. Ook wordt er rekening gehouden met scores die je aan andere artikelen hebt gegeven. Het vernieuwende aan dit algoritme is dat rekening kan houden met de context van de query. Dit komt doordat ze alle gebruikersvoorkeuren representeren in vectoren, deze geven een gewicht aan een

interesse van een gebruiker ten op zichten van een concept in de bij behorende domein ontologie. Deze ontologiën zijn van te voren opgesteld en elk ontologie bestaat uit een aantal concepten. Verder worden de huidige interesses ook uitgebreid door een verspreiding van een semantische voorkeur mechanisme. Deze kan concepten aandragen die een semantische relatie hebben met concepten in de ontologie die de gebruiker interessant vindt. Voor de duidelijkheid met semantisch wordt bedoeld dat een zin of een woord meerdere betekenissen kan hebben in een bepaalde context. Het voordeel van z'n systeem is dat het makkelijk is om relevant nieuws te vinden voor de gebruiker en kan de interesses van de gebruiker worden uitgebreid door toevoeging van semantische relaties waardoor het nog makkelijk wordt om extra relevant nieuws voor de gebruiker te vinden.

The Krakatoa Chronicle The Krakatoa Chronicle is een persoonlijke nieuwskrant op het Internet [10]. De interactie tussen de krant en de gebruiker is heel erg groot, want de gebruiker kan heel veel functies zelf, zoals de layout, helemaal aanpassen. Ook worden er op verschillende manieren informatie verkregen om het gebruikersprofiel op te maken. Aan de ene kant impliciet door te kijken naar welke artikelen die bekijkt of aanpast of juist doorscrolt. Aan de andere kant expliciet doordat de gebruiker de scorebalk bij het desbetreffende artikel verandert. Ook kan het doordat ze specifiek een aantal keywords bijhouden op hun profiel. Deze online krant is vooral een mooi voorbeeld over hoe impliciet informatie over de nieuwsartikelen wordt verkregen met interactie van de gebruiker.

ePaper Het ePaper project heeft als doel om een persoonlijke krant te maken voor op je mobiel [11]. Om de krant persoonlijke te maken maken ze gebruik van een hybrid filtering methode en deze combineert ontologie- content-based filtering met tijdsgevoelige collaborative filtering. Met het gebruik van ontologie wordt er voor gezorgd dat er gemeten kan worden hoeveel een nieuwsbericht verschilt met de interesses van de gebruiker. De content-filtering zorgt er ook voor dat nieuwe items zonder leeshistorie kunnen worden aangeboden. Dynamisch zal een collaborative filter het gewicht aanpassen naarmate een nieuwsitem meer "clicks" krijgt en hoe lang het wordt gelezen. Ook wordt er rekening gehouden met hoe lang het artikel al bestaat, dus na het verstrijken van een bepaald tijdstip, zal de score van het artikel steeds meer afnemen. De uiteindelijke score wordt bepaald door een combinatie van de content-based score en collaborative score te nemen.

Google News Voor Google News is er een hybrid model gemaakt om persoonlijke recommandaties te geven aan gebruikers [12]. De resultaten van dit nieuwe model zijn ook onderzocht. In dit model wordt ook eerst een vorm van content-based toegepast en vervolgens een collaborative filter. Voor de content-based maken ze gebruik van text analyse om te kijken of een artikel bij de voorkeuren van een gebruiker past. Voor het collaborative filter wordt er gebruik gemaakt

van het klikgedrag van de gebruikers. Ook wordt er nog gekeken naar allen de gebruikers uit hetzelfde land. Om te bepalen wat de huidige interesses zijn van de gebruiker hebben ze een Bayesiaans framework gebruikt.Het hele systeem hebben ze ook getest en vergeleken met hoe het nieuws op dat moment werd aangeboden. Uit deze test bleek dat de nieuwe methode meer bezoeker trekt en het de kwaliteit van het nieuws verbeterde.

C.6 Ontwikkelingsmethode

Omdat de ontwikkeling van de Journal erg afhankelijk is van de wensen van de gebruiker en de ontwikkeling van het product niet heel erg rechtlijnig is, wordt er van een agile development methode gebruikt gemaakt. Heel erg belangrijk bij de Journal is dat het systeem goed te onder houden is omdat de Journal eigenlijk altijd wel verbeterd kan worden en omdat je erg van de feedback van klanten afhankelijk bent. Voor dit project gaan we Scrum gebruiken. Deze ontwikkelingsmethode hebben we allemaal in een eerder project al gebruikt en dus hebben we hier al enige ervaring in. Bij scrum wordt er gewerkt met sprints die van een week tot een maand kunnen duren. Omdat we voor dit project niet heel veel de tijd hebben, zullen we proberen om elke week iets gedaan te krijgen. Ook zal er iets minder van pair programming gebruik gemaakt worden dan voor scrum gebruikelijk is, vanwege de korte ontwikkelingstijd. Test-driven development is bij scrum geen vereiste, maar we zullen hier ook niet veel gebruik van kunnen maken, omdat we bij het testen van de algoritmes die sorteren op relevantie, feedback van de gebruiker nodig hebben.

C.7 Conclusie

Bij het bekijken van de recommandatie algoritmen hebben we gezien dat elk van de methoden voor- en nadelen heeft. Daarom hebben we besloten dat we voor het recommanderen van items gebruik moeten maken van een hybride systeem. Ons recommandatie algoritme zal gebaseerd zijn op collaborative (of cummunity-based) en content-based filtering, omdat deze methodes beide kunnen leren en we daarmee het systeem schaalbaar houden. Ook hebben we deze methodes vaak in gebruik gezien op succesvolle nieuwssites. Bij content-based willen we rekening houden met klikgedrag, volggedrag en reacties. Ook compenseert deze methode voor het cold start probleem voor nieuwe artikelen. Het samenvoegen van deze waarderingen zal vervolgens gedaan worden door een weighting hybrid methode, omdat we beide algoritmes in hun volledige kracht willen gebruiken en de andere hybride methodes die we vonden een van de twee algoritmes een beetje afzwakt. We twijfelen nog enigszins over het gebruik van een collaborative methode in plaats van een community methode. Het hoofdzakelijke verschil tussen zo'n soort methode en een community methode is dat een collaborative methode recommendaties doet op basis van gelijksoortige geruikers, terwijl community methodes dit doen op basis van de bedrijven die via

chains direct verbonden zijn met de gebruiker zelf. We weten op dit moment nog niet zeker welke methode de beste resultaten op zal leveren, omdat het voor ons op dit moment nog niet compleet duidelijk is of bedrijven liever berichten zien die betrekking hebben op hun directe chains, of dat ze ook geïnteresseerd zijn in berichten die te maken hebben met anonieme bedrijven waarmee ze volgens ons algoritme een grote gelijkenis vertonen. We zullen tijdens ons project door middel van bijvoorbeeld enquêtes onderzoeken welke methode meer aan de voorkeuren van bedrijven voldoet.

D. Plan van aanpak

Voorwoord

In dit document staat ons plan van aanpak beschreven over ons Bacheloreindproject bij CHAINels. We zullen eerst onze opdracht uitleggen. Vervolgens bespreken we ons aanpak en planning voor onze opdracht. Daarna bekijken we aan welke eisen ons product uiteindelijk moet voldoen en vertellen we hoe we ons aan die eisen gaan houden.

D.1 Introductie

Via onze medestudenten Erwin Buckers en Vincent Koeman zijn wij in aanraking gekomen met CHAINels, een project waar met een klein groepje mensen aan wordt gewerkt. Via hen kregen we een lijst met mogelijke opdrachten die wij konden gaan uitvoeren. Uiteindelijk besloten wij na overleg om aan de Journal van CHAINels te gaan werken. De Journal moet uiteindelijk de meeste relevante berichten van je CHAINs kunnen weergeven. Deze opdracht is ook goedgekeurd door de bachelorcoördinator. In dit plan van aanpak bespreken we eerst onze opdracht met de bijbehorende eisen en onze planning en sluiten we af met hoe we ons aan de verschillende eisen gaan houden.

D.2 Projectopdracht

D.2.1 Projectomgeving

Op dit moment is CHAINels nog in ontwikkeling. Er is al een oude bèta versie online, maar binnenkort komt er een nieuwe open bèta online. Het doel van CHAINels is om uiteindelijk miljoenen gebruikers te kunnen ondersteunen,daarom is het van belang dat tijdens de ontwikkeling van de website goed over verschillende optimalisaties moet worden nagedacht. De Journal, waar wij ons op richten, is een van de dingen die nog gedaan moeten worden voor de launch en zal de belangrijkste berichten (onder andere vraag en aanbod, events en gesponsorde berichten) tonen.

D.2.2 Doelstelling project

De Journal zal een dynamisch onderdeel worden van de website waarvan altijd een verticale balk aan de linkerkant van het scherm zal worden getoond. Door op de balk te klikken, schuift de Journal open over de pagina en wordt het volledig zichtbaar. In deze Journal zullen zoals gezegd de belangrijkste berichten op een aparte pagina getoond worden zodat een bedrijf hier een snel overzicht van krijgt. Belangrijk voor ons is om deze in een overzichtelijk geheel weer te geven en de berichten te sorteren op relevantie..

D.3 Opdrachtformulering

Wij zijn verantwoordelijk voor zowel de front-end als backend van de Journal. De verantwoordelijkheid voor de front-end wordt gedeeld met Willem Buys. Wij zijn in dit opzicht verantwoordelijk voor de correcte werking van de frontpagina, terwijl hij voor de lay-out verantwoordelijk is. Voor de back-end hiervan is ons team volledig verantwoordelijk. Voor de kwaliteit van de frontpagina delen wij echter de verantwoordelijkheid.

D.3.1 Op te leveren producten en diensten

Het resultaat van het project zal een Journal zijn voor de website van CHAINels, die door de bedrijven van CHAINels gebruikt kan worden.

D.3.2 Eisen en beperkingen

Onze opdrachtgevers hebben zelf ook nog geen compleet beeld van hoe de Journal er helemaal uit moet komen te zien. Er wordt daarom ook van ons verwacht dat we tijdens ons project input leveren met onze eigen ideeën hierover. Een hoop details zullen dus tijdens de gesprekken met onze projectbegeleiders uitgewerkt moeten worden, waardoor we op dit moment op sommige punten nog geen harde eisen kunnen stellen. De volgende eisen worden aan onze Journal gesteld:

- Verschillende soorten berichten moeten getoond worden in de Journal. De volgende soorten berichten bestaan er: nieuwsberichten, vraag- en aanbodberichten, infoberichten, events en gesponsorde berichten. Er moet ook een duidelijk onderscheid zijn tussen deze berichten.
- Berichten met de hoogste relevantie voor het bedrijf moeten vooraan in de Journal geplaatst worden.
- Gesponsorde resultaten moeten ten minste aan X andere bedrijven getoond worden. Aangezien deze hele feature nog niet in het systeem zit, kan deze eis nog veranderen.
- Overzichtelijke weergave van berichten.

- De Journal moet in een zodanige tijd (¡ 1s) geladen worden, dat het niet nodig is om de gebruiker een wachtscherm te laten zien voor het laden van de Journal.
- Een zoekfunctie zodat je in de Journal berichten kan zoeken op keywords.
- De Journal moet in ieder geval weergegeven kunnen worden in Internet Explorer 7 of hoger, Firefox 12 of hoger, Chrome 18 en Opera 1

D.3.3 Voorwaarden

Voor ons werk aan de Journal is het nodig dat het systeem ons een manier geeft om toegang tot de informatie van bedrijven te krijgen. Het is aan onze opdrachtgevers om hiervoor te zorgen. Verder verwachten we van onze opdrachtgevers dat ze eens per week aanwezig zullen zijn om onze voortgang te bespreken.

D.4 Aanpak en tijdsplanning

D.4.1 Oriëntatiefase (Week 1-2)

We kijken naar vergelijkbare producten zoals sociale netwerken en online kranten en naar algoritmen die gebruikt kunnen worden in ons ontwerp. We hebben een aantal vergaderingen met verschillende mensen van het team achter CHAINels om iedereen een beetje te leren kennen en een algemeen idee te krijgen van wat er tot nu toe allemaal al aan CHAINels gedaan is. Op het moment van schrijven is dit plan van aanpak het laatste wat nog gedaan moet worden in onze oriëntatiefase; het oriëntatieverslag zelf is al helemaal af. Er hoeft voor deze fase dus geen lijst van activiteiten meer opgesteld te worden.

D.4.2 Ontwerpfase (Week 3-4)

Dit bestaat uit het uitdenken van ons algoritme en het maken van de indeling van de Journal in samenwerking met Willem. Belangrijke beslissingen met betrekking tot het ontwerp worden in deze fase op papier gezet. De eerste week van deze fase (week 3) zullen we besteden aan het uitwerken van onze algoritmes. Aangezien dit het belangrijkste onderdeel van ons project is, willen we hier goed de tijd voor nemen. Daarom zullen we deze hele week hiervoor vrij houden. Tijdens deze week is het nodig dat we een aantal keren met leden van het CHAINels team afspreken om te overleggen over de algoritmes. Zij kunnen aangeven of er nog bepaalde dingen zijn waar de algoritmes rekening mee moeten houden. De tweede week van deze fase (week 4) houden we vrij om de berichten die in de Journal moeten komen in te delen. Hiervoor zullen we met Willem over Skype moeten overleggen, omdat hij deze periode in Duitsland is. Voorbeelden van problemen die we nog moeten bespreken, zijn hoe we de verschillende soorten berichten onderling gaan ordenen en op welke manier we het gevoel van een gedrukte krant terug willen laten komen in de Journal. Hier hebben we ook een volle week voor uitgetrokken.

D.4.3 Implementatiefase (Week 5-8)

In deze fase zal het gekozen ontwerp zal worden geïmplementeerd. Naast de implementatie zelf, zullen er in deze fase ook testcases bij de implementatie worden gemaakt. Deze testcases zullen geschreven worden gedurende de gehele periode. Er zal in iteraties van ongeveer een week worden gewerkt. In week 5 zullen we het algoritme zoals we dat hebben uitgedacht implementeren. Het doel is om in deze week iets te hebben dat in ieder geval kan worden gebruikt, zodat we het aan bedrijven kunnen tonen en kunnen onderwerpen aan een gebruikerstest. In week 6 willen we de implementatie afmaken en de voorlopige versie aan een gebruikerstest onderwerpen. Aan het eind van deze week zullen we onze code naar de software improvement group sturen. In week 7 en 8 zullen we de verzamelde feedback uit de gebruikerstest en de software improvement group gebruiken om het product te verbeteren.

D.4.4 Afronding (Week 9-10)

Het afronden van het project. Dit bestaat onder andere uit het vinden en opsporen van fouten. Verder zullen slordigheden verbeterd worden en zullen er optimalisaties gedaan worden. Als we tijd over hebben kunnen we aan de presentatie beginnen. In week 9 zullen we nog een keer ons hele product, de bijbehorende documentatie en alle benodigde documenten bekijken en de laatste verbeteringen aanbrengen waar dat nodig is. Ons doel is om aan het einde van deze week ons gehele product naar onze tevredenheid af is. De documenten die we tot dan toe hebben opgesteld, kunnen we dan gelijk gebruiken voor ons eindverslag. We zullen aan het einde van week 9 beginnen met het schrijven en samenstellen van dit eindverslag. Week 10 zullen we compleet besteden aan het afronden van dit verslag. Als alles volgens de planning verloopt, hebben we voor de rest alles af tegen deze tijd.

D.4.5 Presentatie (Week 11 of 12)

In deze laatste week moeten we onze eindpresentatie geven. Afhankelijk van de definitieve datum zullen we in de dagen daarvoor deze presentatie afmaken en voorbereiden.

D.5 Projectinrichting

D.5.1 Organisatie

De organisatie van ons bacheloreindproject ziet er als volgt uit:

- Opdrachtgever vanuit CHAINels: Dr.Ir. S.I. Sudlle
- ICT afdeling CHAINels: Erwin Buckers en Vincent Koeman
- Design afdeling CHAINels: Willem Buys

- TU Delft begeleider: Tomas Klos

D.5.2 Personeel

Het is de bedoeling dat we tien weken lang fulltime aan het bacheloreindproject werken. Op maandag, woensdag en vrijdag zullen we altijd samenkomen op de TU Delft en aan het project werken. Dinsdag en donderdag verschilt of we in de TU zullen werken of thuis. Dit komt doordat Jeroen nog een minor vak in Utrecht moet volgen. Verder verwachten we dat iedereen over de vaardigheden beschikt die geleerd zijn bij vakken zoals Objectgeoriënteerd Programmeren, Software Engineering en Software Kwaliteit en Testen.

D.5.3 Administratieve procedures

Elke twee weken moeten we een voortgangsverslag opsturen naar onze TU-begeleider. Daarnaast hebben we al een oriëntatieverslag gemaakt waarin we ons hebben georiënteerd in ons probleem.

D.5.4 Rapportering

De communicatie tussen onze opdrachtgever zal voornamelijk lopen via Vincent en Erwin. Met hen hebben we ook elke week een paar meetings waar we onze voortgang bespreken. Ook hebben we af en toe een meeting met de opdrachtgever zelf erbij.

D.5.5 Resources

CHAINels heeft op dit moment nog geen eigen werkplek. Daarom zullen onze werkplekken zich in de TU Delft bevinden. Dit is geen vaste locatie, omdat het niet mogelijk is om werkplekken te reserveren voor een langere periode. Voornamelijk zullen onze werkplekken in de bibliotheek of EWI/Drebbelweg zijn.

D.6 Kwaliteitsborging

Om de kwaliteit van ons werk te waarborgen zullen er eens per week vergaderingen plaatsvinden met onze opdrachtgevers. Hierdoor kunnen we in ieder geval van onze werkgevers feedback krijgen. Ook zijn we van plan om, zoals we in de tijdsplanning hebben aangegeven, na de eerste week van de implementatiefase een voorlopige versie van de Journal te laten gebruiken door een aantal bedrijven om te kijken wat zij er tot dan toe van vinden. De feedback die we hierop krijgen, kunnen we vervolgens gebruiken in het vervolg van de implementatiefase om de werking van de Journal beter aan te passen aan de wensen van de gebruikers.

Voorafgaand hieraan kunnen we door middel van enquêtes alvast bij bedrijven nagaan wat hun voorkeur is voor het design van de Journal, zodat we daar

tijdens het ontwerpen van de Journal ook rekening mee kunnen houden. Voor de pure functionele aspecten zoals de werking van een knop, zullen we testcases schrijven, zodat de werking van deze functies gecontroleerd kan worden. Om ervoor te zorgen dat vertrouwelijke informatie niet openbaar wordt, hebben we een geheimhoudingsverklaring ondertekend.

Bibliografie

[1] Gediminas Adomavicius and Alexander Tuzhilin. Towards the next generation of recommender systems: A survey of the state-of-the-art and possible extensions.

[2] Robin Burke. Hybrid recommender systems: Survey and experiments. 2001.

[3] Iván Cantador and Pablo Castells. Semantic contextualisation in a news recommender system.

[4] SongJie Gong. A collaborative filtering recommendation algorithm based on user clustering and item clustering. 2010.

[5] Jianming He and Wesley W. Chu. A social network-based recommender system (snrs).

[6] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks.

[7] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features.

[8] J. Liu, P. Dolan, and E.R. Pedersen. Personalized news recommendation based on click behavior. *INRIA*, 2010.

[9] Stanley Loh, Fabiana Lorenzo, Ramiro Saldaña, and Daniel Licthnow. A tourism recommender system based on collaboration and text analysis. 2004.

[10] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems.

[11] K. Nageswara Rao and V.G. Talwar. Application domain and functional classification of recommender systems–a survey.

[12] Jason D. M. Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction.

[13] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo.

[14] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization.

[15] Hanhuai Shan and Arindam Banerjee. Generalized probabilistic matrix factorizations for collaborative filtering.

[16] Jun Wang, Arjen P. de Vries, and Marcel J.T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion.

[17] Jun Wang1, Arjen P. de Vries, and Marcel J.T. Reinders. Unified relevance models for rating prediction in collaborative filtering.

[18] Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. Scalable collaborative filtering using cluster-based smoothing.

[19] Daqiang Zhang, Jiannong Cao, Jingyu Zhou†, Minyi Guo†, and Vaskar Raychoudhury. An efficient collaborative filtering approach using smoothing and fusing.

[20] Tinghui Zhou, Hanhuai Shan, Arindam Banerjee, and Guillermo Sapiro. Kernelized probabilistic matrix factorization: Exploiting graphs and side information.