Trendclustering op Twitter

Master project BMI

Februari 2011

Martijn Moest

ParaBotS

MASTER PROJECT BMI

TRENDCLUSTERING OP TWITTER

Auteur: Martijn Moest
Studentnummer: 1777750
Opleiding: Business Mathematics and Informatics
Begeleider Parabots: Matthijs Mulder
Eerste lezer VU: Sandjai Bhulai
Tweede lezer VU: Wojtek Kowalczyk

Februari 2011

ParaBotS B.V.
Singel 160
1015 AH Amsterdam

Vrije Universiteit Amsterdam
Faculteit der Exacte Wetenschappen
De Boelelaan 1081a
1081 HV Amsterdam

Voorwoord

Ter afsluiting van de masteropleiding Business Mathematics and Informatics aan de Vrije Universiteit Amsterdam moet er een stage bij een bedrijf worden gedaan. Mijn stage heeft plaatsgevonden bij Parabots (deel van de SMR-groep) en ik heb onderzoek gedaan naar clusteringalgoritmes. Met een van deze algoritmes heb ik een stroom van korte berichten afkomstig van de 'social media' website Twitter geclusterd. Naast de vele kennis die ik opgedaan heb over dit onderwerp, heb ik ook op het programmeervlak veel bijgeleerd.

Ten eerste wil ik bij deze graag mijn stagebegeleider Matthijs Mulder bedanken voor zijn vele adviezen, zijn enthousiaste begeleiding en het telkens meedenken met de problemen waar ik tegenaan liep. Ten tweede wil ik graag mijn begeleider vanuit de universiteit, Sandjai Bhulai bedanken voor zijn begeleiding. Ook zou ik Marten den Uyl willen bedanken voor het mogelijk maken van deze stage en voor zijn inbreng tijdens het project. Als laatst wil ik ook nog Peter Terpstra bedanken voor zijn hulp. Ik heb een erg plezierige tijd gehad tijdens de stage en dit komt mede door mijn aardige collega's van de SMR-groep.

Martijn Moest
Februari 2011

Samenvatting

Parabots doet aan monitoring van de sentimenten op internet over het Nederlandse domein. Voor blogs en nieuwsartikelen wordt dit al gedaan, maar nu moet er ook vastgesteld worden welke trends er op Twitter gaande zijn. In deze scriptie is een online clusteringmethode ontwikkeld die deze analyse tracht te doen. Dit algoritme is gebaseerd op een al bestaand algoritme, genaamd OCTSM. De keuze op dit algoritme is gevallen na een uitgebreide literatuurstudie van de mogelijke algoritmes. Het voordeel van dit algoritme is dat door middel van frases een beter beeld van de context van de tekst gekregen kan worden, waardoor onderwerpen nauwkeuriger geclusterd kunnen worden. Dit algoritme werkt met een offline en online proces. In het offline proces moet informatie van een stroom gehaald worden, waar het algoritme later, in het online proces, zijn voordeel mee kan doen. Stel dat in het offline proces er een woord heel vaak in combinatie met een aantal frases is voorgekomen. Als deze frases zich later dan ook in een cluster bevinden dan kan er met grotere zekerheid gezegd worden dat dit woord meer kans heeft om echt bij dit onderwerp te horen.
Er is ook een dataset van tweets verzameld waarop dit algoritme getest kan worden. De tweets die verzameld zijn, zijn afkomstig van twitteraars die een volger zijn van een politieke partij. Er is immers een grote kans dat deze volgers zelf ook Nederlandse tweets produceren. Deze dataset is gebruikt voor het maken van twee andere datasets. Beide bevatten 20 verschillende trends. Deze trends zijn geïdentificeerd door een hashtag die zich in de tweets bevindt. De ene dataset bevat al deze tweets en de andere dataset bevat dezelfde tweets met uitzondering van de hashtag. Hierdoor wordt het mogelijk te testen of het algoritme genoeg heeft aan de context van de trend.
Dit bleek niet zo te zijn. De hoofdconclusie die uit deze scriptie voortkomt, is dat het erg moeilijk is tweets correct te clusteren, omdat een tweet simpelweg niet genoeg inhoud heeft. Het OCTSM algoritme is niet in staat om een echte datastroom van tweets te clusteren. Als er een beperkte dataset beschikbaar is dan is dit wel mogelijk. Het ’burst algoritme’ dat veel voorkomende woorden in een bepaalde periode identificeert zou hiervoor een uitkomst kunnen bieden.

Inhoudsopgave

1 Inleiding

Dit hoofdstuk begint met achtergrondinformatie over het bedrijf Parabots en de sociale media site Twitter. Dit wordt gevolgd door de probleemstelling en een paragraaf over de onderzoeksvragen waarin de aanleiding en de doelstelling van het onderzoek worden beschreven. Ten slotte zal ook de structuur van het verslag worden besproken.

1.1 ParaBots

ParaBots ontwikkelt programma's en diensten die omgaan met de altijd groeiende informatiestroom op het internet, in kranten en andere media. Dit bedrijf houdt zich vooral bezig met informatie-extractie, text mining en zoeken op het web met behulp van de modernste technieken op gebied van natuurlijke taalverwerking.
ParaBotS is opgericht in 2001, is gevestigd in Amsterdam en heeft momenteel twee werknemers. ParaBotS is onderdeel van de SMRgroep. De SMRgroep bevat ook nog twee andere bedrijven die actief zijn in verschillende gebieden van AI toepassingen:

- Sentient Information Systems
- VicarVision
- ParaBotS

Een belangrijk product voor ParaBots is het XBotSsysteem, waarbij software agents -parabots- het web zelfstandig afzoeken naar door gebruikers opgegeven inhoud. Sentiment monitoring op internet is een ander toepassingsgebied, zie bijv. ParaBotS' 'online barometer'[W6].

1.2 Twitter

In deze scriptie gaat het om de *social mediasite* genaamd *Twitter*. Wikipedia [W9,W10] geeft de volgende definities over deze twee begrippen:

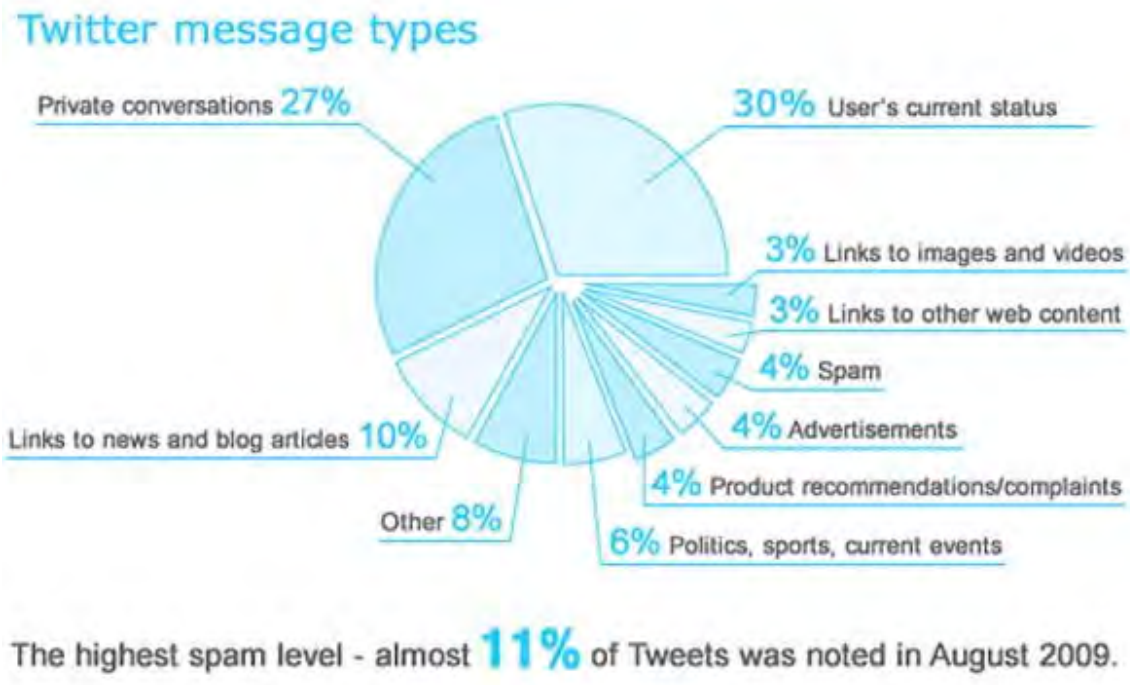
“Social media is de, ook in het Nederlandse taalgebied gangbare, Engelse benaming voor online platformen waar de gebruikers, met geen of weinig tussenkomst van een professionele redactie, de inhoud verzorgen. Tevens is er sprake van interactie en dialoog tussen de gebruikers onderling.”

“Twitter is een internetdienst waarmee gebruikers korte berichtjes publiceren. Het is een sociaal netwerksite waarop men zichzelf, zoals bij Facebook, een profiel en een avatar kan aanmeten.”

Het oorspronkelijke idee achter Twitter was om interesses met elkaar te delen en om er achter te komen wat er zoal speelt in het leven van mensen. Er waren al manieren om met elkaar te communiceren (bijvoorbeeld e-mail, blogs of de telefoon), maar om iedereen te gaan bellen om te vragen hoe het ermee gaat is niet ideaal. Bij Twitter maakt iedereen een eigen profiel waar ze kort aan kunnen geven waar ze mee bezig zijn of wat er speelt in hun leven. Door een dergelijk profiel kunnen er interesses van mensen worden getoond, die voorheen niet zichtbaar waren. Hierdoor kan er makkelijker een connectie ontstaan tussen mensen. De berichten kunnen ook een onderwerp hebben. Dit wordt aangegeven door een woord met een hashtag er voor (bijvoorbeeld #ajax, #explosie of #pvv). Als er over een persoon gesproken wordt dan wordt dit aangegeven met een ‘@’ voor zijn of haar naam. Meestal is dit de naam

van het twitter-account van deze persoon, maar dit is niet noodzakelijk. Ook is het mogelijk om een tweet van iemand anders zelf te tweeten. Dit heet retweeten en komt voor als een tweet erg interresant wordt gevonden. Een retweet is te herkennen aan de letters RE gevolgd door de naam van de auteur van de originele tweet (bijvoorbeeld: 'RE @wilders' gevolgd door het bericht). Elk bericht bestaat maximaal 140 tekens.

De populariteit van de social media zijn de afgelopen jaren met ongekende snelheid toegenomen. Facebook is zelfs de nummer 2 van de meest bezochte websites ter wereld. Ook Twitter doet het niet slecht met een 10de plaats [W8]. Twitter is een nieuwe manier van communiceren geworden. Er wordt niet alleen getweet over wat er in het leven van een gebruiker gebeurt, maar het kan ook een bron van informatie zijn. Twitter staat er namelijk ook om bekend dat het sneller nieuws produceert dan de 'reguliere media'. Nu tegenwoordig het gros van de mensen een telefoon met internet bij zich heeft, kan er binnen mum van tijd een tweet worden geproduceerd. Een voorbeeld is de dood van Michael Jackson. Iemand die het nieuws over zijn dood ergens op had gevangen tweette het bericht en het was binnen de kortste keren wereldnieuws. Zo zijn er nog tal van voorbeelden waar Twitter aan de wieg stond van een nieuwsdoorbraak [W11]. Twitter is vooral populair op het moment dat er een goed bekeken evenement plaatsvindt, zoals het NBA basketbaltoernooi in de Verenigde Staten of het wereldkampioenschap voetbal. In de finale van de NBA was er een productie van 3,085 tweets per seconde geteld [W12]. Dit is echter nog niet het record. Dat staat momenteel namelijk op 3,283 tweets per seconde. Deze enorme stroom aan tweets vond plaats toen tijdens het WK voetbal, Japan won van Denemarken met een nipte overwinning [W13]. Waar Twitter nog meer voor gebruikt wordt is te zien in figuur 1.



Figuur 1. Overzicht van waar Twitter voor gebruikt wordt [W7].

1.3 Probleemstelling

Er komen steeds meer tekst-gebaseerde communicatiemiddelen, waarbij korte teksten, 'posts', bijv. tweets, blog-reacties, msn, chat en krabbels, in een continue stroom gegenereerd worden. Om overzicht te krijgen -en te houden- van die steeds wisselende stromen is het gewenst verschillende posts naar inhoud -en mogelijk ook bron en context- te kunnen groeperen. Daarvoor zijn allerlei clusteringalgoritmen in de statistische en datamininglitteratuur beschikbaar, maar er worden veel eisen en wensen aan praktisch bruikbare algoritmen gesteld.

Continue datastroom

In de eerste plaats zoeken we clusteringmethoden die op een continue stroom van data kunnen werken, in plaats van op een gegeven, statische dataset. Anders gezegd, ga er van uit dat de set van te clusteren data sneller verandert, door nieuwe feeds, dan een standaard clusteringalgoritme voor statische datasets (zoals k-means[1] of kohonen maps) nodig heeft voor het bepalen van de optimale clustering van de huidige dataset. Online clustering betekent aan de ene kant dat de clustering in real time geüpdate kan worden, maar daarnaast ook, dat de huidige clusteringstructuur, de opdeling in groepen of segmenten, steeds een goed beeld geeft van de 'meest recente' posts (afgelopen uren of dagen) in het licht van het verleden (afgelopen weken, maanden, jaren). En dus dat wat eens als uitgangspunt is genomen, maar beperkte invloed mag hebben op wat nu de clusteringresultaten zijn.

Schaalbaarheid

Vervolgens dient de methode schaalbaar te zijn voor echt grote datastromen; in eerste instantie willen we met duizenden feeds per uur of dag kunnen werken, uiteindelijk moet met miljoenen feeds per dag gewerkt kunnen worden. Dat is alleen haalbaar met hiërarchische clustermethoden[1].

Heterogene data

Een enkele 'post' is al een complex dataobject dat op verschillende manieren kan worden gerepresenteerd in de clustering. Het ligt voor de hand uit te gaan van een 'bag of words'-model[2], maar dat is niet genoeg. Een post heeft als bijzondere kenmerken o.a. 'auteur', 'tijdstip' en mogelijk ook 'context', andere posts of artikelen waarop gereageerd wordt. In de tekst van een post kunnen zaken gerepresenteerd worden die niet zomaar in een 'bag of words'-model passen: frasen, named entities, referenties. De clusteringmethode moet dus met heterogene of samengestelde representaties om kunnen gaan.

Representatie

Gevonden clusters moeten automatisch geprofileerd kunnen worden door een korte en representatieve tekst. Inhoud van -samengestelde- clusters moet ook kunnen worden weergegeven door tag clouds (waarin grotere aantallen inhoudswoorden of namen voorkomen.)

[1] Zie voor een nadere uitleg van de basisclusteringsbegrippen paragraaf 3.2.

[2] In dit model, wordt een tekst (zoals een zin of een document) gepresenteerd als een ongeordende verzameling van woorden, afgezien van de grammatica en zelfs woordvolgorde.

Perceptie van de data

Iedere clustering impliceert altijd een perspectief op de geclusterde objecten, ten gevolge van de relatieve weging van verschillende kenmerken. Het is gewenst dat alternatieve wegingen naast elkaar gebruikt kunnen worden, zodat steeds verschillende perspectieven op de datastroom beschikbaar zijn.

1.4 Onderzoeksvragen

Het doel van de stage is primair het ontwikkelen van een online clusteringmethode, daarnaast dient aandacht besteed te worden aan verzameling van posts en aan de online presentatie van resultaten. Er is voor gekozen om voor dit onderzoek alleen tweets als input te gebruiken. Hierbij gaat het alleen om de analyse van tweets uit Nederland en de applicatie moet in programmeertaal Perl worden gemaakt. De onderzoeksvraag die hieruit voorkomt is de volgende:

Hoe kunnen aan de hand van een clusteringmethode de meest voorkomende onderwerpen, op Twitter of andere bronnen van een constante informatiestroom, in kaart worden gebracht?

De deelvragen die hierbij horen luiden als volgt:

- Hoe worden de posts van de constante datastroom verzameld?
- Zijn er al bestaande technieken op het gebied van topicdetectie d.m.v. clustering en zijn er al methodes voor het clusteren van tweets en reacties op forums, nieuwswebsites en blogs?
- Hoe wordt een eventueel gevonden clusteringtechniek toegepast?
- Hoe wordt de gevonden clusteringtechniek geëvalueerd?

Als de vorige vragen beantwoord zijn, kan er ook nog nagedacht worden over de volgende vraag:

- Hoe worden de gevonden onderwerpen gerepresenteerd?

1.5 Structuur van het verslag

In hoofdstuk 2 zal worden gekeken hoe een dataset van tweets kan worden verzameld en kan worden opgeslagen. Hierna zal er in hoofdstuk 3 een literatuuroverzicht worden gegeven van technieken over de representatie van tekst, het clusteren van datastromen en de validatie van clusteringalgoritmes. De implementatie van het algoritme dat als beste uit de literatuurstudie komt, zal in hoofdstuk 4 worden toegelicht. Hoe de gevonden validatiemethodes op dit algoritme worden toegepast zal in hoofdstuk 5 worden beschreven. In het zesde en laatste hoofdstuk zal alles op een rijtje worden gezet en zullen hier conclusie en aanbevelingen aan verbonden worden.

2 Tweets verzamelen

In augustus 2010 ligt het aantal Tweets dat per dag wordt geproduceerd op 90 miljoen [W3]. Er zijn een aantal websites die trends (meest voorkomende woorden) over al deze tweets bijhouden. Hier worden echter de trends overheerst door Engelse tweets. Amerika is namelijk het land met veruit de meeste actieve twitteraars [W4]. De data die echter vooral interessant is voor Parabots zijn de Nederlandse tweets. Deze tweets moeten er voor zorgen dat er een betere sentimentanalyse van Nederland kan worden gemaakt. In dit hoofdstuk wordt vooral gekeken of, en zo ja, hoe, het mogelijk is om al deze Nederlandse tweets te verzamelen.

2.1 Beperking van de verzameling

Twitter ziet graag dat er zoveel mogelijk uit de geproduceerde tweets gehaald wordt. Hierdoor heeft Twitter een stroom van tweets opengesteld voor particulieren en bedrijven die een applicatie willen maken met deze informatie. Het aantal tweets dat je mag ontvangen via de publieke *application programming interface* (API) hangt echter af van het voorstel dat aan Twitter wordt gedaan. Na uitleg van ons plan voor een applicatie, hebben wij toegang gekregen tot het volgen van 100.000 twitteraars. Dit is nog niet genoeg voor alle 168.300 actieve Nederlandse twitteraars [W4], maar is ruim voldoende voor implementatie van dit onderzoek.

Het is echter nog niet bekend welke twitteraars Nederlandse tweets twitteren. De vraag is hoe deze mensen kunnen worden geïdentificeerd. Bij elk Twitteraccount is het wel mogelijk om aan te geven in welke taal de twitteraar het liefst twittert, maar bij deze instelling kan er niet gekozen worden voor de optie ‘Nederlands’. Er zijn ook nog andere instellingen waaruit afgeleid zou kunnen worden in welke taal getwitterd wordt. Alleen zijn deze instellingen niet helemaal betrouwbaar. Dit zijn bijvoorbeeld de instelling *locatie* of *time_zone*. Er zijn heel veel Nederlandse twitteraars die bij *time_zone*, Amsterdam invullen of bij *locatie* invullen dat ze in één van de vele Nederlandse steden wonen. Echter wordt bij *time_zone* ook veel *Greenwich* ingevuld. Hier vallen meerdere landen onder, waardoor deze instelling geen duidelijkheid verschaft. Ook de instelling *locatie* is niet betrouwbaar. Het is voor de computer namelijk moeilijk te onderscheiden welke locaties precies bij Nederland horen. Bovendien zijn er ook nog twee andere nadelen. Ten eerste wil het feit dat een persoon bijvoorbeeld in Amsterdam woont of een Nederlander is, nog niet zeggen dat alle tweets die getweet worden door deze persoon allemaal in het Nederlands worden geschreven. Ten tweede is het nooit zeker of deze informatie naar waarheid is ingevuld.

Een ander probleem is dat van te voren niet bekend is waar alle id’s van de Nederlandse twitteraars vandaan gehaald moeten worden. De oplossing voor deze beide problemen ligt bij het verzamelen van tweets die in zekere zin al zijn gecategoriseerd. Er kunnen namelijk op Twitter lijsten bijgehouden worden voor het categoriseren van bijvoorbeeld familie of een bepaalde vriendengroep. Deze lijsten worden ook gebruikt voor het bijhouden van alle mensen die in een bepaalde politieke partij zitten.

Politiek is altijd een veelbesproken onderwerp en is daarom ook erg interessant voor sentimentanalyse. Een bijkomend voordeel is dat de kans groot is dat de mensen die Nederlandse politieke partijen volgen zelf ook Nederlands twitteren. Daarom is er besloten alleen mensen te volgen die politieke interesse hebben. Iemand wordt hierbij als ‘politieke volger’ geclassificeerd, als deze persoon een politicus volgt uit een van de politieke lijsten of het hoofdprofiel van een politieke partij. Een voorbeeld van een

politieke lijst is ‘pvda/tweede-kamerleden’ [W5]. Op deze lijst bevinden zich alle twitterende Tweede Kamerleden van de PvdA. Maar naast de landelijke politici, worden ook gemeentelijke lijsten op politici gescand. Doordat er per partij wordt gescand, wordt naast iemand als ‘politieke volger’ te classificeren het ook mogelijk om deze twitteraar te classificeren als een aanhanger van een bepaalde politieke partij. Op deze manier wordt het in een later stadium mogelijk om de sentimenten van een bepaald onderwerp van verschillende partijaanhangers tegenover elkaar te zetten.
Twitteraars die Nederlandse politici volgen, hebben dus een grote kans zelf ook Nederlands te twitteren. Dit is echter lang niet altijd het geval. Om dit op te lossen wordt de taal van de tweets van alle twitteraars bepaald. Als een twitteraar veel Nederlandse tweets produceert, wordt deze als zodanig geclassificeerd. Hoe dit precies wordt toegepast is te lezen in paragraaf 2.2.2.
Het onderwerp politiek kan later ook nog altijd worden uitgebreid naar andere onderwerpen. Zo zouden bijvoorbeeld alle volgers van beroemdheden, voetballers, zangers, sporters of een bepaald merk op eenzelfde wijze in kaart kunnen worden gebracht.

2.2 Invulling van de database

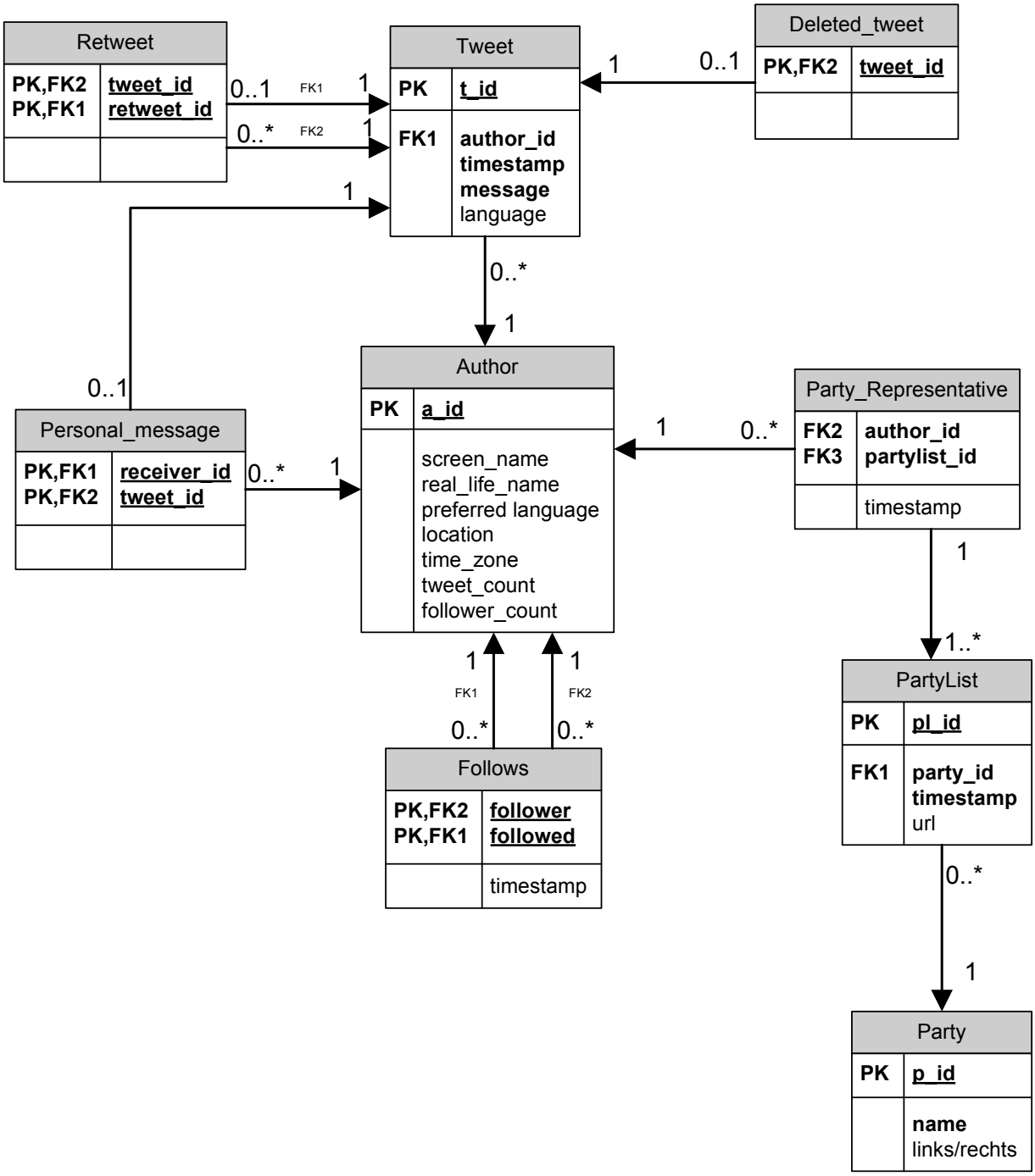
De opslag van alle data gebeurt in de database waarvan het relatiemodel in figuur 2 te zien is. Zoals in de vorige paragraaf is vastgesteld, moeten er eerst id’s verzameld worden van mensen die politieke interesse hebben. Hierna is het pas mogelijk om alle tweets op te vangen. Hoe dit in zijn werk gaat, wordt in deze paragraaf besproken.

2.2.1 Het vinden van politici en hun volgers

Het vinden van politici en hun volgers is een proces dat offline plaatsvindt en ter initialisatie wordt gedaan. Door dit proces komen alle id’s, die met de twitterstroom gevolgd moeten worden, beschikbaar. Deze data kan verouderd raken en zou daardoor ook weer bijgewerkt moeten kunnen worden.

Initialisatie

Alle politieke partijen die zich in Nederland in de nationale politiek bevinden zijn handmatig opgeslagen in de tabel *Party*. Van elke politieke partij zijn er een aantal lijsten opgezocht en wederom handmatig in de tabel *Party_list* geplaatst. Deze *Party_List* is gekoppeld aan de tabel *Party_Representative*. Hierin worden de lijsten gekoppeld aan alle politieke twitteraars (mensen die een partij representeren) die gevonden zijn op de betreffende lijst. In figuur 2 is te zien dat dit gebeurt via de tabel *Author*. Deze tabel, samen met de tabel *Tweet*, vormt de basis van het databasemodel. Deze tabellen worden in de volgende paragraaf besproken. Voor nu is het van belang om te weten dat alle politici eerst als auteurs worden opgeslagen en dat de id’s van deze auteurs worden gekoppeld aan een partijlijst. De volgende stap is het vragen aan de API van Twitter wie de volgers zijn van al deze politieke twitteraars. Al deze personen worden ook als eerst weer opgeslagen in de tabel *Author*. Vervolgens worden alle volgers (*follower*) en de persoon die gevolgd wordt (*followed*) aan elkaar gekoppeld in de tabel *Follows*. Hierbij kan dus een auteur nul of meerdere keren gevolgd worden en kan ook nul of meerdere auteurs volgen. De status *followed* kan dus alleen verkregen worden als deze persoon ook in de tabel *Party_Representatives* bestaat.



Figuur 2. Database relatiemodel van de opslag van de gegevens van Twitter.

Updateproces

Wat voor de rest nog opvalt in de hiervoor genoemde tabellen is dat er bij een aantal hiervan een *timestamp* wordt bijgehouden. Dit heeft te maken met het updateproces. Na verloop van tijd is de opgeslagen data niet meer helemaal correct. Twitteraars kunnen bijvoorbeeld hun volgerslidmaatschap

van een twitteraar opzeggen, of nieuwe mensen hebben zich juist aangemeld voor een lidmaatschap. Ook de mensen die zelf op de lijsten staan kunnen zijn veranderd.
Als politici niet meer voorkomen in een lijst, dan is het zonde om alle data van al de volgers van deze politici weg te gooien. Op een later tijdstip zou het eventueel weer kunnen dat deze twitteraars weer als een volger van iemand anders worden aangemerkt. Op dat moment kan de data van het verleden weer herbruikt worden. Bij elke update wordt de *timestamp* bijgewerkt naar de *timestamp* van de laatste update. Dit maakt het mogelijk alleen twitteraars te selecteren die bij de laatste update als politieke volger waren geclassificeerd.

Op 13-10-2010 is de laatste update geweest. De gegevens die bij deze update zijn verzameld zijn te zien in tabel 1. Hierbij zijn bijvoorbeeld bij GroenLinks vanuit 6 partijlijsten 226 unieke partijleden gehaald. Al deze partijleden hadden bij elkaar 406.527 unieke volgers. Het totaal van alle politieke volgers is hierbij niet ingevuld, omdat dit niet het unieke aantal volgers zijn. Een twitteraar kan namelijk meerdere partijen volgen. Het unieke aantal politieke volgers is 573.652. Het echte aantal partijleden is ook nog minder dan in de tabel genoemde 2.133. Er zijn een paar leden die met verschillende partijen geassocieerd worden. Dit kunnen profielen zijn van mensen/partijen die een samenwerkingsverband hebben met meerdere partijen of een partijlid die van de ene naar de andere partij is overgestapt. Ook is het mogelijk dat de lijstmaker een fout heeft gemaakt, maar de 10 gevallen waar het hier om ging hadden meer met de eerste twee redenen te maken. Het unieke aantal partijleden is namelijk 2.121 (een account zat ingedeeld bij drie partijen). GroenLinks heeft vooral zoveel volgers omdat er twee partijleden zijn die extreem veel volgers hebben. Zo heeft Femke Halsema rond de 100.000 volgers en heeft Arjan El Fassed er rond de 290.000.

Naam	Politieke volgers	Partijleden	Aantal partijlijsten
GroenLinks	406.527	226	6
D66	98.227	380	7
CDA	96.510	256	5
PvdA	85.901	401	9
VVD	78.196	419	8
PVV	60.025	24	4
CU	25.026	185	9
SP	23.468	150	6
TON	12.808	29	4
PvdD	9.706	9	3
SGP	7.317	40	6
Piratenpartij	4.406	12	3
Lijst 17	1.619	2	2
Totaal	-	2133	72

Tabel 1. Statistieken van de politieke twitteraars.

Het is een langdurig proces

Tijdens het updateproces worden er drie verschillende verzoeken aan de Twitter API gedaan:

1. Geef alle informatie van alle leden van een lijst.
2. Geef alle informatie van een betreffend profiel (gebruikt bij een hoofdprofiel van een partij).
3. Geef alle informatie van alle volgers van een bepaald profiel (politicus).

Bij punt 1 moet dit verzoek ook nog meerdere keren gedaan worden als een persoon te veel volgers heeft. Twintig volgers kunnen er namelijk per verzoek worden opgehaald. Hetzelfde geldt voor punt 3 als er teveel personen in de lijst staan. Per honderd volgers moet er weer een nieuwe aanvraag worden gedaan. Dit zorgt ervoor dat het aantal aanvragen aardig oploopt. Dit is relevant want er zit namelijk een limiet op het aantal verzoeken dat er gedaan mag worden. Er mogen namelijk maar 350 verzoeken per uur worden gedaan. De teller van dit uur begint te lopen nadat het eerste verzoek is gedaan. Hierdoor wordt het programma ernstig vertraagd. Het is daarom van belang om zo weinig mogelijk verzoeken te doen. Daarom wordt er ook nog bijgehouden van welk profiel de volgers al zijn opgevraagd. Een politicus kan namelijk in meerdere lijsten zitten en als bijvoorbeeld het profiel van Arjan El Fassed vervolgens in 3 lijsten voorkomt dan is de programmaduur 16 uur (((290.000/100)/350)*2) langer dan nodig. Maar ondanks deze maatregel is de programmaduur nog steeds erg lang. De laatste update duurde ongeveer 40 uur.

2.2.2 Het opvangen van de twitterstroom

Tweets opvangen van de politieke volgers met behulp van een twitterstroom is een proces dat online plaatsvindt. De id's die zijn gevonden bij het initialisatieproces worden opgegeven als te volgen twitteraars. Vervolgens retourneert Twitter van al deze id's niet alleen alle tweets die door deze twitteraars geproduceerd worden, maar ook alle tweets die worden verwijderd, (door derden) aan deze twitteraars zijn gericht (@username…. de rest van het persoonlijke bericht) of zijn geretweet (RT @username…. de rest van de al eerder geschreven tweet). Telkens als er een tweet binnenkomt wordt er gekeken welk van de bovenstaande mogelijke tweets is binnengekomen. Vervolgens gebeurt er het volgende bij de desbetreffende varianten:

- Een gewone (door een van de volgers getweette) tweet:
    - Alle eigenschappen van de auteur worden bijgewerkt.
    - De tweet wordt opgeslagen in de tabel Tweet en gekoppeld aan de zojuist aangemaakte auteur.
- Een (door een van de volgers) verwijderde tweet:
    - Als de betreffende tweet in de tabel Tweet voorkomt, dan wordt het tweet_id in de tabel Deleted_tweet gezet.
- Een tweet van een volger die geretweet is door iemand anders dan een van de volgers:
    - Er wordt een nieuwe Author aangemaakt als de persoon nog niet bestaat en als de persoon wel bestaat dan worden alle eigenschappen van de auteur bijgewerkt.
    - De tweet wordt opgeslagen in de tabel Tweet en krijgt een aparte id en wordt gekoppeld aan de zojuist bijgewerkte of aangemaakte a_id.
    - Een id van de originele tweet wordt als retweet_id opgeslagen in de tabel Retweet en een id van de zojuist gecreëerde tweet wordt als tweet_id opgeslagen en gekoppeld aan tweet_id.

- Een tweet van iemand anders dan een van de volgers die gericht is aan een van volgers:
    - Er wordt een nieuwe Author aangemaakt als de persoon nog niet bestaat en als de persoon wel bestaat dan worden alle eigenschappen van de auteur bijgewerkt.
    - De tweet wordt opgeslagen in de tabel Tweet, krijgt een aparte id en wordt gekoppeld aan de zojuist bijgewerkte of aangemaakte a_id.
    - De id van de zojuist gecreëerde tweet wordt als tweet_id opgeslagen in de tabel Personal_message en een id van de auteur waar het bericht aan gericht is, wordt als receiver_id opgeslagen en gekoppeld aan tweet_id.

Uiteindelijk komen er auteurs in de tabel *Author* die politieke volgers zijn, een tweet van een politieke volger hebben geretweet of een persoonlijk bericht aan een politieke volger hebben gestuurd. Maar niet al deze mensen worden gevolgd. Dit zijn alleen de politieke volgers en politici zelf. In de vorige paragraaf is al aangegeven dat het aantal volgers ruim boven de 500.000 ligt. Dit terwijl er door restricties van Twitter er maar 100.000 mensen gevolgd mogen worden. Er moet daarom een filter worden toegepast op het aantal potentiële volgers. Deze filter heeft drie criteria. De eerste twee zijn de twee eenvoudigste. Deze twee criteria filteren op het aantal tweets dat door een auteur is getweet en hoeveel volgers deze persoon heeft. Het eerste criterium test of deze auteur een frequente twitteraar is. Er zijn genoeg accounts die maar een enkele keer twitteren. Als er nauwelijks tweets geproduceerd worden dan heeft het ook geen nut om deze accounts te volgen. Het tweede criterium kijkt naar de populariteit van iemand. Als iemand weinig volgers heeft betekent dit, dat deze tweets niet interessant zijn voor veel mensen. Als ze niet interessant worden bevonden door het publiek dan hebben deze tweets ook weinig impact. Er wordt minder naar dit criterium gekeken dan naar de eerste, want het is belangrijker dat een persoon veel tweet dan dat deze persoon populair is. Een uitzondering wordt gemaakt voor personen die extreem veel volgers hebben, maar zelf weinig twitteren. Dit zijn blijkbaar belangrijke personen. Het zou misschien later interessant kunnen zijn om te kijken wat er tegen deze personen gezegd wordt. Een voorbeeld hiervan is de huidige partijleider van de VVD, Mark Rutte. Hij twittert zelf nooit maar wordt wel gevolgd door 20.000 volgers. Bij een sentimentanalyse van de VVD zou dit handige informatie kunnen opleveren.
Dit brengt ons tot het belangrijkste criterium. Hier wordt gekeken of de personen die geselecteerd zijn wel Nederlandse tweets produceren. In tegenstelling tot de vorige twee criteria is de geprefereerde taal niet een van de eigenschappen die in het profiel van een account te vinden is. Dit moet worden afgeleid van de tweets die door de accounts worden geproduceerd. Dit gebeurt met behulp van een module in Perl die een tekst kan classificeren op taal. Aangezien de tweets redelijk klein zijn, is het voor heel wat tweets niet mogelijk ze correct te classificeren. Daarom wordt een tweet pas geclassificeerd als er een minimum van 7 woorden in de zin zitten. Hierbij wordt twitterjargon zoals ‘RT @username’, ‘@username’ of ‘#onderwerp’, URL’s en smileys eruit gefilterd, zodat deze niet als woorden worden meegeteld. Een andere reden waarom ze eruit worden gefilterd zijn, is omdat deze woorden geen positief effect hebben op de taalclassificatie. Als er zich te weinig woorden bevinden in de tweet of de taalclassifier geeft geen significante uitslag aan, dan wordt deze tweet als ‘-’ geclassificeerd. Na verloop van tijd heeft een account meerdere tweets geproduceerd. Een percentage van deze tweets heeft waarschijnlijk een taal toegewezen gekregen (bijvoorbeeld ‘nl’ of ‘en’) en een aantal niet (‘-’). Het kan bijvoorbeeld voorkomen dat iemand een aantal tweets in het Nederlands schrijft, een paar in het Engels,

een paar in het Zweeds en een paar die niet geclassificeerd zijn. In welke taal schrijft deze persoon nu het liefst? Deze vraag wordt door de volgende criteria beantwoord:

T_{totaal} : Het totaal van het aantal tweets dat is geclassificeerd (language is niet gelijk aan '-').

T_{max} : Het aantal tweets van de taal die het meest voorkomt.

$(T_{max}/T_{totaal})>0.8$ en $T_{totaal}>4$

De taalclassifier is ook niet 100% betrouwbaar, vandaar dat de hiervoor opgenoemde criteria van belang zijn. Telkens als er een tweet wordt toegevoegd, wordt het *preferred_language* attribuut van de tabel *Author* bijgewerkt. Als een bepaald persoon niet aan de bovenstaande taalcriteria voldoet dan blijft dit attribuut ongeclassificeerd ('-'). Dit heeft als nadeel dat het wat langer duurt voordat er bekend is wie de Nederlandse twitteraars zijn. Vandaar dat alle mensen worden gevolgd die momenteel '-' of 'nl' zijn geclassificeerd. Naarmate de applicatie langer loopt zullen er meer taalvoorkeuren bekend zijn. Om een indicatie te geven: Op het moment dat er 4.600.000 tweets waren verzameld, waren 26.000 mensen geclassificeerd als 'nl'. Deze mensen worden in een later stadium gebruikt om trendclustering op toe te passen. Het is daarom van belang dat de datastroom een tijdje heeft gedraaid voordat het clusteringalgoritme in gebruik kan worden genomen. Hoe de te volgen mensen uiteindelijk bepaald worden is te zien in Bijlage I. Het aantal mensen dat overbleef was 150.000, maar neemt elke dag af aangezien telkens meer Engelse twitteraars worden geclassificeerd. De twitteraars zijn gerangschikt op aantal *followers* en aantal tweets en daarvan worden de eerste 100.000 geselecteerd om te volgen. Deze twitteraars, produceren ongeveer 28.000 tweets per uur.

De invoerstroom is nu geprepareerd. Er kan nu een clusteringalgoritme op worden toegepast om trends te herkennen. Welk algoritme dit precies zal zijn, zal in het volgende hoofdstuk worden besproken.

3 Zoeken naar een efficiënt trend-detecteringsalgoritme

Het hoofdzakelijke probleem dat opgelost moet worden in deze scriptie is het vinden van een algoritme dat vanuit een continue stroom van kleine berichten de belangrijkste onderwerpen detecteert. Dit heeft tot gevolg dat het algoritme niet op een vaste dataset kan worden getraind, maar op een datastroom. Het gevaar dat een algoritme het niet meer bij kan benen door een te grote toevoer ligt hierdoor op de loer. Er moet dus opgelet worden dat het algoritme efficiënt werkt. In dit hoofdstuk zullen dan ook verschillende algoritmes besproken worden die de literatuur beschrijft over het omgaan met deze datastromen.
Het volgende probleem is het detecteren van onderwerpen uit een tekst. Een extra moeilijkheid hierbij is dat de teksten waar de onderwerpen uit gefilterd moeten worden erg klein zijn. Als er eenmaal een algoritme gevonden is moet er uiteindelijk ook getest worden of dit algoritme aan bepaalde voorwaarden, met betrekking tot precisie en totaal aantal ontdekte trends, voldoet. Vandaar dat er ook wordt gekeken hoe dit precies gemeten kan worden. Voordat er überhaupt wat met tekstdata kan gebeuren moet er eerst de juiste representatie worden gevonden. Hier zal dit hoofdstuk dan ook mee beginnen.

3.1 Tekstrepresentatie

In deze paragraaf wordt gekeken naar manieren om verschillende documenten met elkaar te vergelijken. Hierbij kan een document uit een woord bestaan of uit een of meerdere zinnen. Tweets bestaan uit maximaal 140 tekens en vormen dus kleine documenten. Als eerst wordt het TF*IDF representatiemodel besproken en zal worden gevolgd door de uitleg van het extended Semantic smoothing model.

3.1.1 TF*IDF

Term-frequency inverse-document-frequency is een methode die vaak gebruikt wordt voor de representatie van woorden en ziet er als volgt uit: w = tf*idf. Hierbij wordt een gewicht w per woord berekend dat zwaarder wordt naarmate het woord een grote frequentie heeft in het document zelf en lichter wordt naarmate het woord ook nog veel voorkomt in de overige documenten in het corpus. Om ook nog documenten van verschillende grootte met elkaar te kunnen vergelijken vindt er ook nog een normalisatie plaats.

3.1.2 Extended Semantic smoothing model

De meeste algoritmes die met tekst werken, maken gebruik van het TF*IDF-algoritme om documenten te representeren. TF*IDF let hierbij alleen op de frequentie van de woorden. Dit model houdt echter geen rekening met de context waarin de woorden geschreven zijn. Woorden die een dubbele betekenis hebben worden bijvoorbeeld als dezelfde gezien. Een model dat hier wel rekening mee houdt, is het extended Semantic smoothing model [32] (gebaseerd op het Semantic smoothing model van [33][3]). Dit

[3] Het verschil met de 'extended' versie is dat het model zo is aangepast dat er nu met datastromen gewerkt kan worden. Om het 'language model' te maken wordt er namelijk gebruik gemaakt van alle woorden in het cluster en alle woorden van alle documenten. Alle woorden van alle documenten zijn echter nog niet beschikbaar als er met datastromen wordt gewerkt. Vandaar dat er nu, in de 'extended' versie bij dit model, alleen gekeken wordt naar de woorden in het cluster zelf.

model kan bijvoorbeeld naast het werken met frases van een enkel woord ook met frases van meerdere woorden werken.

Semantic smoothing is een samengesteld model $p_{bt}(w|c_j)$. Hierbij staat p_{bt} voor de kans op woord w als cluster c_j gegeven is. Het bestaat uit twee gedeeltes: Het gedeelte waar verschillende woordcombinaties eruit worden gehaald is het 'language model' $\mathrm{p_b}(\mathrm{w}|\mathrm{c_j})$. Dit is in principe een kans die alleen kijkt naar frequentietelling van frases. In het andere gedeelte worden verschillende frases aan elkaar gelinkt. Als bijvoorbeeld de frases 'ster' en 'oneindig heelal' in hetzelfde document worden gebruikt dan gaat het over een ster in de lucht en niet over een bekendheid. Dit wordt gedaan in het 'translation model' $\mathrm{p_t}(\mathrm{w}|\mathrm{c_j})$. Uit dit model komt een waarde dat het verband tussen de woordcombinatie beschrijft. De invloed van de twee modellen wordt gecontroleerd door de translatiecoëfficiënt λ. Het Semantic smoothing model ziet er uiteindelijk als volgt uit:

$$p_{bt}(w|c_j) = (1-\lambda)p_b(w|c_j) + \lambda p_t(w|c_j) \quad (1)$$

3.2 Inleiding clusteren/basisbegrippen

In de rest van dit hoofdstuk zullen er nog een aantal begrippen met betrekking tot clusteren de revue passeren. Deze zullen hier kort worden uitgelegd. Het gaat in deze scriptie vooral over clustering van stromen, vandaar dat niet alle standaard clusteringmethodes behandeld zullen worden.

Dimensie

Het aantal lagen waaruit de data bestaat heet de dimensie. Als de data tekst is, dan is het aantal dimensies bijvoorbeeld gelijk aan het aantal woorden dat het document bevat. Aangezien documenten uit veel woorden bestaan, heeft tekstdata normaal gesproken een hoge dimensie. Het gemiddelde van het aantal woorden per tweet in de dataset van hoofdstuk 2 is ongeveer 13 met een maximum van ongeveer 32. Dit is al redelijk veel, maar als de tweets samengevoegd worden dan kan dit alleen maar oplopen. Ook is het mogelijk dat niet naar de woorden wordt gekeken, maar naar het aantal zinnen of letters wordt gekeken. Als dit het geval is dan verandert ook het aantal dimensies. In de context van het clusteren van tweets is dit echter niet waarschijnlijk.

k-means

k-means [6][7] is een clustering-benadering waarbij elk cluster een centraal punt heeft. Hierbij moet het aantal clusters (k) op voorhand worden gespecificeerd en wordt elk punt in de dataset toegewezen aan het dichtstbijzijnde centrale punt. Deze centrale punten worden vertegenwoordigd door het gemiddelde van elk van de k clusters te nemen. Het basis algoritme luidt als volgt:

1. Selecteer k punten als de eerste groep centroides.

Herhaal de volgende twee stappen net zolang totdat de centroides zich niet langer verplaatsen.

2. Clusters worden gevormd door elk punt aan het dichtstbijzijnde centroide toe te kennen.
3. Als alle items zijn toegewezen moeten de posities van de k centroides worden herberekend.

Initiële centroides zijn vaak willekeurig gekozen. Centroides in de volgende iteraties worden bepaald door het gemiddelde van de punten in de desbetreffende clusters te nemen. Hoe dicht een punt bij een centroide is, kan worden gemeten door het bepalen van de Euclidische afstand, correlatie, etc. k-means

heeft echter een paar beperkingen. k-means heeft het namelijk lastig als clusters verschillen in grootte, dichtheid of als het om niet-bolvormige clusters gaat.

Hiërarchisch clusteren

Hiërarchisch clusteren produceert een set van clusters die worden gestructureerd in een hiërarchische boom. Een groot voordeel van deze techniek is dat er van te voren niet hoeft te worden aangegeven hoeveel clusters er nodig zijn. Elk gewild aantal clusters kan worden verkregen door de boom op het juiste niveau 'af te snijden'. Het basisalgoritme van Johnson [35] luidt als volgt:

Als een verzameling van N punten die moeten worden geclusterd, en een N * N afstand (of gelijkenis) matrix zijn gegeven, dan gaat de basis van hiërarchische clustering als volgt:

1. Elk van de N items wordt aan een cluster toegewezen. Wat dus resulteert in N clusters met elk een item. Maak vervolgens de afstanden (gelijkenissen) tussen de clusters even groot als de afstanden tussen de objecten die ze bevatten.

Herhaal de volgende twee stappen totdat alle items zijn geclusterd in een cluster van grootte N.

2. Zoek het dichtstbijzijnde (meest gelijkende) paar van clusters en voeg ze samen in een enkel cluster.
3. Bereken de afstanden tussen het nieuwe cluster en elk van de oude clusters.

De zojuist beschreven methode is de klassieke methode. Dit is de agglomeratieve soort van hiërarchische clusteren. Hier begint het algoritme met de punten als afzonderlijke clusters. Verdeeld hiërarchische clusteren begint echter met een cluster dat alle punten bevat. Bij elk van de volgende iteraties wordt dit cluster net zo lang opgesplitst totdat het aantal clusters gelijk is aan het aantal punten.
Om de gelijkenis tussen clusters te definiëren kunnen er verschillende maten worden gebruikt:

- De minimale afstand tussen twee punten uit twee verschillende clusters.
- De maximale afstand tussen twee punten uit twee verschillende clusters.
- Het groepsgemiddelde van de afstand tussen alle punten van de twee clusters.
- De afstand tussen centroides van twee clusters.
- Andere methoden met behulp van een doelfunctie.

Een nadeel van deze techniek is dat wanneer twee clusters worden gecombineerd tijdens het proces, het niet meer ongedaan gemaakt kan worden. Bovendien is het algoritme gevoelig voor ruis en kan grote clusters opbreken.

3.3 Het clusteren van datastromen

In deze paragraaf zullen acht verschillende clusteringalgoritmes op het gebied van datastromen worden besproken. Dit zijn de volgende algoritmes: Birch, Cobweb, Stream, Clustream, Fractionele clustering, Dichtheidsclustering, het *burst*-algoritme en OCTS & OCTSM. Voordat deze algoritmes zullen worden besproken zal er eerst worden stilgestaan bij de noodzakelijke benodigdheden van een goed datastroom-clusteringalgoritme die door Daniel Barbará [1] worden voorgesteld.

3.3.1 Noodzakelijke benodigdheden van een goed datastroom-clusteringalgoritme

Een goed datastroom-clusteringalgoritme moet volgens Barbará aan de volgende drie eisen voldoen:

Compacte representatie: Een compacte representatie is noodzakelijk om zoveel mogelijk geheugen te besparen. De beslissing van waar het volgende punt moet komen in het model mag niet afhangen van alle tot op heden verwerkte punten. Hierdoor zou de rekentijd lineair (of zelf sneller) omhoog gaan met het aantal punten dat gevonden is en dit zou dan resulteren in veel te lange rekentijd naarmate de datastroom veel groter wordt. De representatie moet dus verschillende groepen datapunten samenvatten.

Snelle en incrementele verwerking van nieuwe datapunten: Het gebruikte algoritme moet aan de volgende twee condities voldoen:

- De plaatsing van nieuwe punten mag niet besloten worden op basis van een algoritme die een vergelijking maakt met alle punten geplaatst in het verleden. Ook moet deze representatie kunnen worden gebruikt voor de evaluatie van het model.
- Het algoritme moet goed presteren, de complexiteit mag niet te groot zijn en zou bijvoorbeeld lineair kunnen toenemen met de grootte van de representatie. Met de snelheid dat de data wordt geproduceerd, moet het ook verwerkt worden.

Een duidelijke en snelle manier van identificeren van uitschieters en het herkennen van nieuwe clusters: De uitschieters staan hier voor de punten die niet passen in de tot dusver gemaakte clusters. Als er genoeg uitschieters zijn dan moet er een nieuw cluster worden gemaakt. Het te gebruiken algoritme moet deze punten kunnen herkennen. Als het algoritme te veel uitschieters constateert dan is het mogelijk dat er een nieuwe cluster gevormd moeten worden. Om te bepalen of er te veel uitschieters zijn waardoor het algoritme niet meer valide is, zouden de grenzen van Chernoff [2] gebruikt kunnen worden.

De grenzen van Chernoff worden als volgt toegepast: er wordt een random variabele X_i (elke *i* refereert naar een nieuw punt) gedefinieerd, die de waarde 0 krijgt als het een uitschieter is en de waarde 1 krijgt als het punt in een van de clusters ligt. De kans (p) dat de waarde 1 wordt, is niet bekend en moet worden geschat. Dit gebeurt door de som te nemen van alle X_i (met i= 1,…,n waarbij n het aantal geclassificeerde punten is), gedefinieerd als *X*, en vervolgens te delen door n. Deze schatting zal vervolgens gekoppeld worden aan de daadwerkelijke kans in vergelijking (2).

$$PR\left[\left|\frac{X}{n} - p\right| \leq \varepsilon\right] > 1 - \delta \quad (2)$$

In vergelijking (2), staat ε voor de beoogde maximale afwijking van de schatting ten opzichte van de echte waarde en δ geeft een willekeurig kleine kans aan. Vergelijking (2) zorgt er eigenlijk voor dat de geschatte waarde en de echte waarde dicht bij elkaar kunnen komen te liggen door δ goed te kiezen.

Door de ongelijkheid van Chernoff te gebruiken, kan de schatting van de succeskans worden begrensd. Dit wordt gedaan door te zeggen dat $\frac{X}{n}$ groter moet zijn dan (1+ ε)p (zie vergelijking 3)

$$PR\left[\frac{X}{n} > (1+\varepsilon)p\right] \leq e^{\left(-\frac{pn\varepsilon^2}{3}\right)} \quad (3)$$

Er is een bewijs [36] dat laat zien dat vergelijking (2) overeind blijft als het aantal succesvol geclassificeerde waardes (s) begrensd wordt door vergelijking (4) en n begrensd wordt door vergelijking (4).

$$s > \frac{3(1+\varepsilon)}{\varepsilon^2} \ln\left(\frac{2}{\delta}\right) \text{ (4)}$$

$$n \leq \frac{3(1+\varepsilon)}{(1-\varepsilon)\varepsilon^2\, p} \ln\left(\frac{2}{\delta}\right) \text{ (5)}$$

Door vergelijking (4) en (5) kan er gekeken worden of het huidige model nog valide is, naarmate er nieuwe data binnenkomt. Als er na n ontvangen punten, s succesvol kunnen worden geclassificeerd dan is het model nog steeds valide. Als dit niet het geval is dan moet er een nieuw model worden gemaakt. In dit geval zijn er twee mogelijkheden om dit te doen:

- Her-clusteren van alle datapunten tot nu toe. Dit is misschien erg geheugenintensief, maar hoeft dat niet te zijn, mits er een efficiënte representatie is van de data dusver.
- Niet kijken naar alle voorgaande data en alleen met de n laatste punten een nieuwe set van clusters maken. Deze techniek is al succesvol toegepast op datastromen in een ander onderzoek van Daniel Barbará [3].

Samenvattend zijn de algemene specificaties waar een datastroom-clusteringalgoritme aan moet voldoen de volgende:

- Het algoritme moet een compacte representatie hebben.
- Het algoritme moet snel en incrementeel nieuwe datapunten kunnen verwerken.
- Er moet een duidelijke en snelle manier zijn om de uitschieters te identificeren.
- Nieuwe clusters moeten herkend worden.

Naast de algemene specificaties moet het datastroom-clusteringalgoritme voor het onderzoek naar tweets ook nog aan andere specificaties voldoen:

- Het algoritme moet efficiënt genoeg zijn om elke tweet die binnenkomt te verwerken. Hierbij mag niet de situatie ontstaan waar er een niet kleiner wordende wachtrij van tweets ontstaat.
- Het algoritme moet met tekst om kunnen gaan.
- Het algoritme moet clusters van onderwerpen maken in 1 of in meerdere woorden.
- De clusters die gemaakt zijn moeten de trends van het laatste uur, dag, week en maand kunnen tonen (of een willekeurige periode, als dit mogelijk is).
- Het algoritme moet om kunnen gaan met grote dimensies. Tekstrepresentatie bestaat namelijk uit veel woorden en met elk woord groeit de dimensie.

Welke verschillende datastroom-clusteringalgoritmes er al zijn zal in de volgende paragrafen worden besproken. Aan de hand van de specificaties die in deze paragraaf gegeven zijn, kunnen deze algoritmes worden beoordeeld.

3.3.2 Birch

Birch [4] is een incrementeel en hiërarchisch clusteringalgoritme voor erg grote databases en is daardoor in potentie ook geschikt op het gebied van datastromen. Het sterkste punt van dit algoritme is het efficiënte geheugengebruik. Het algoritme bestaat uit een hiërarchische clustercomponent en een geheugenstructuur-component:

- Hiërarchische component: Het algoritme start met alle punten in de database te bekijken als een cluster. Vervolgens groepeert het de dichtstbijzijnde punten totdat er nog maar één cluster over

is. Het uitrekenen van de clusters wordt gedaan door een afstand matrix (O(n^2) in grootte en O(n^2) in tijd).

- Geheugenstructuur-component: Birch gebruikt een geheugen data structuur genaamd CF-boom ('clustering feature'-tree). Deze boom is zo georganiseerd dat elk blad in de boom een cluster representeert. Naarmate er meer data beschikbaar komt, wordt er steeds gekeken of limiet R niet wordt overschreden. Hierbij is *R* de limiet die aangeeft wat de grootte van elk cluster moet zijn. R hangt hierbij af van het gemiddelde en het gemiddelde van de som van de kwadraten, die bij elke node van de CF-boom worden bijgehouden. Het algoritme start met een dataset en probeert de grootte van R te gokken zodat de boom in het geheugen past. Als de boom niet in het geheugen past dan wordt R verkleind en dan wordt vervolgens de boom opnieuw gemaakt. Dit proces gaat door totdat de boom in het geheugen past.

Als de grootte van het cluster te groot wordt, dan wordt het in tweeën gesplitst en worden de clusters geredistribueerd. De punten worden vervolgens constant ingevoegd aan de clusters. Met elke node van de boom houdt de CF-boom bij wat het gemiddelde van het cluster is en wat het gemiddelde van het somatische kwadraat van de foutmarge is om de grootte van de clusters efficiënt te bepalen. De boomstructuur hangt ook samen met de vertakkingsparameter T, dat het maximale aantal kinderen per node bepaalt.

Tot slot kan dit algoritme ook na het construeren van de boom wat aanpassingen doen om uitschieters te verwijderen en het clusteren te verbeteren.

De literatuur zegt het volgende over de prestaties van dit algoritme:

- Uit de bevindingen van [1] blijkt dat naarmate de data meer dimensies krijgt, (bij tekstrepresentatie zijn dit er erg veel) het gebruik van de harde schijf erg snel toeneemt. Birch is zo ontworpen dat de clusterrepresentatie eerst op de harde schijf moet worden opgeslagen en dat de initiële clusters in het werkgeheugen worden opgeslagen. Echter, naarmate het cluster groter wordt, dan moet de CF-boom groeien aan de hand van data op de harde schijf. Hierdoor is deze techniek dus niet geschikt voor datastromen. [30] zegt ook dat Birch alleen goed werkt op data met lage dimensies.
- De functie is gemaakt om het geheugengebruik van de clustering van een nieuw punt te minimaliseren, maar het duurt toch nog erg lang [1]. Het zou dan efficiënter zijn om de punten in groepen te verwerken.
- Dit algoritme is in potentie wel erg goed in het spotten van trends [1]. Birch her-evalueert op een bepaalde tijd telkens of de gevonden uitschieters in de huidige clusters passen, zonder dat de boom in grootte toeneemt. In potentie mag dit dan wel zo zijn, maar het onderzoek van Wei [W1] geeft aan dat dit niet het geval is. Hij zegt dat alle oude punten als even belangrijk worden gezien als de nieuwe punten, waardoor trends in veranderlijke datastromen niet kunnen worden gedetecteerd. Ook [12] is het op dit punt eens met Wei.
- Birch kan alleen met numerieke data omgaan [W2].
- Birch is gevoelig voor de volgorde waarin data wordt ingevoerd [W1, 30].
- Hiërarchische algoritmes zoals Birch staan er om bekend dat als clusters eenmaal gesplitst of samengevoegd worden, dat er geen mogelijkheid is om deze actie ongedaan te maken [28].
- Birch presteert alleen goed op data met sfeervormige clusters [30, 31].

3.3.3 Cobweb

Cobweb [11] is een incrementeel algoritme dat bedoeld is om begrijpbare patronen in data te ontdekken. Dit algoritme maakt gebruik van de 'category utility'-functie [5] voor classificatie en voor het opzetten van een boom. Cobweb houdt een hiërarchisch clusteringmodel bij in de vorm van een classificatieboom. Elke node van de boom bevat een op kans gebaseerde beschrijving, die een samenvatting geeft van de node. Voor elk nieuw punt volgt Cobweb een pad van de boom naar beneden. Tijdens de wandeling op dit pad wordt er een update gedaan voor elke tegengekomen node. Met behulp van de 'category utility'-functie wordt de beste node gekozen voor het te plaatsen punt.

De literatuur zegt het volgende over de prestaties van dit algoritme:

- De classificatieboom wordt niet op hoogte gebalanceerd [1]. Dit zorgt ervoor dat de ruimte- en tijdcomplexiteit drastisch verslechtert, waardoor Cobweb niet goed te gebruiken valt voor datastromen.
- De slechte clusterrepresentatie zorgt er voor dat de tijdcomplexiteit door het toevoegen van een nieuw punt aan de bestaande clusters erg verslechtert [1].
- In vergelijking met andere hiërarchische methodes, heroverweegt Cobweb wel zijn eerder gemaakte beslissingen [10]. De manier van clusteren is echter zo efficiënt gedaan dat de kwaliteit van clusters mogelijk van slechte kwaliteit zijn.
- Elke keer als er een nieuw punt bijkomt kijkt de 'category utillity'-functie of het beter is om het punt aan een bestaande cluster toe te voegen of om het punt een nieuw cluster te laten beginnen [1]. Uitschieters en dus ook trends worden hierdoor goed in de gaten gehouden.

3.3.4 Stream

Stream is de naam van het algoritme dat beschreven wordt in [29]. Stream verdeelt datastromen in groepen van punten, net zo lang totdat het werkgeheugen vol is. Alle punten van elke groep B_i worden geclusterd en van elke groep worden de centra C_i van het cluster bijgehouden. Het gewicht dat C_i heeft, wordt bepaald door het aantal punten dat zich in het cluster bevindt. C_i is hierbij een set van centra dat verwijst naar puntengroep B_i. Vervolgens maakt Stream een clusteringmodel door de gewogen centra van elk tot nu toe onderzochte groep (C_o,…, C_i) te clusteren. Bij zowel het clusteren van de puntengroepen en het clusteren van de centra van de puntengroepen, gebruikt het algoritme het 'Local search'-algoritme. Dit algoritme garandeert dat de oplossing niet slechter is dan het aantal keer het optimale [29]. Hierbij is de optimale oplossing de minimale gemeten afstand (tussen de punten en de k clustermedianen of clustercentra) gedeeld door de sommatie van de kwadraten van deze afstand (SSQ). Dit doel is hetzelfde als het doel van het k-means algoritme. Dit algoritme kan echter geen grenzen garanderen. Local search loopt in tijd asymptotisch lineair met het aantal punten.

De literatuur zegt het volgende over de prestaties van dit algoritme:

- Aangezien de representatie van de huidige clusters alleen maar bestaat uit een lijst van verschillende centra C_i, maakt dit een compacte manier [1]. De grootte van de representatie neemt echter wel toe met de hoeveelheid puntgroepen die erbij komen. Hoe meer data er aanwezig is, des te minder efficiënt de representatie.
- De eerste clusteringstap die plaatsvindt, neemt niet toe in tijd naarmate er meer punten bijkomen. De tweede clusteringstap neemt echter wel evenredig toe met het aantal nieuwe punten [1]. De makers van

het algoritme [7] geven ook toe dat het probleem oplossen met het k-means algoritme sneller gaat dan het oplossen door middel van een begrensde oplossing.

- Clusters kunnen verschillende vormen aannemen. Door het minimaliseren van SSQ wordt er echter voor gezorgd dat er alleen hypersferen kunnen worden gecreëerd, waardoor andere vormen niet gedetecteerd kunnen worden [1].
- Bij deze techniek wordt geen rekening gehouden met uitschieters [1]. De uitschieters zullen dus deel gaan uitmaken van de clusters en zullen uiteindelijk in een hypersfeer worden geclusterd met de niet uitschieters waardoor er dus geen goede clusters uitkomen. Hierdoor kan geconcludeerd worden dat dit algoritme niet goed gebruikt kan worden voor het ontdekken van trends.
  Wei [W1] zegt dat ook bij dit algoritme alle oude punten als even belangrijk worden gezien als de nieuwe punten, waardoor trends in veranderlijke datastromen niet kunnen worden gedetecteerd.
- De gemaakte clusters zijn, net als bij het Birch-algoritme, gebaseerd op de gehele data waardoor recente veranderingen niet worden opgemerkt [12, 14].

3.3.5 CluStream

CluStream [12] is een algoritme dat niet alleen rekening houdt met de grote hoeveelheid data dat geclusterd moet worden, maar kijkt ook naar het veranderende gedrag van de datastroom. Er kan met dit algoritme namelijk een bepaalde periode (bijvoorbeeld de laatste dag, maand of jaar) worden geselecteerd waarvoor de clusters van de datastroom moeten worden gemaakt. Dit algoritme voorkomt dat de vele hoeveelheid oudere data, verzameld over een lange periode, niet zo zwaar meeweegt met de analyse op de huidige data.

CluStream maakt gebruik van een online en een offline component. De online component maakt, verzamelt en legt statistische samenvattingen van de inkomende data vast. De offline component gebruikt deze samenvattingen gecombineerd met de input van de gebruiker (het geprefereerde tijdsinterval), om inzicht te krijgen in de clusters. De samenvattingen worden gemaakt aan de hand van de in paragraaf 3.3.2 beschreven ‘cluster feature’-vector. Deze worden dan opgeslagen in microclusters. Deze clusters zijn in eerste instantie ontstaan door in de offline component een k-means algoritme toe te passen op de als eerst inkomende data. Dit gebeurt net zo lang totdat er geen extra clusters meer in het geheugen passen. Om de hoeveel tijd een samenvatting moet worden opgeslagen, hangt af van de minimale gewenste tijdsinterval waarover trends moeten worden gedetecteerd. Door deze microclusters vervolgens in een piramidevormig tijdframe te plaatsen kunnen er voor verschillende periodes clusters worden opgevraagd. Hierbij wordt voor alle gecreëerde clusters een identificatienummer bijgehouden. Als er verschillende clusters vervolgens moeten worden samengevoegd dan kan er aan de bijgehouden id-lijst worden gezien uit welke cluster het huidige cluster bestaat. Het clusteren zelf gebeurt aan de hand van een aangepast k-means algoritme [13].

In het artikel zelf [12] wordt het volgende geconcludeerd over het CluStream algoritme:

- CluStream creëert hogere kwaliteit clusters dan de traditionele datastroom-cluster algoritmes (hier vallen onder andere de hiervoor beschreven algoritmes onder). Dit geldt vooral als er grote veranderingen in de data zitten.
- Door het gebruik van het piramidevormig tijdframe en microclusters ontstaat er op een efficiënte manier een hoge accuratesse in de clusters.

- CluStream kan goed omgaan met hoge dimensies, datastromen en grote hoeveelheden clusters.

De literatuur zegt het volgende over de prestaties van dit algoritme:
CluStream is gebaseerd op het k-means algoritme. Dit zorgt er voor dat er alleen sfeervormige clusters kunnen worden gedetecteerd. Clusters van een andere vorm kunnen hierdoor minder goed worden gevonden [14, 31]. Willekeurige grenzen van de clusters vinden is belangrijk omdat op deze manier zo efficiënt mogelijk de onderliggende data wordt beschreven, waardoor er datareductie plaatsvindt.
- k-means zorgt er ook voor dat ruis en uitschieters niet kunnen worden gedetecteerd [14].
- Het algoritme is afhankelijk van parameters. *K* en de tijdsperiodes moeten namelijk door de gebruiker worden ingevoerd [14, 31].
- Door de k-means structuur van dit algoritme gaat accuratesse van de clusters naar beneden als er veel uitschieters zijn [31].

3.3.6 Fractionele clustering

Fractionele clustering wordt beschreven in [8] en is een algoritme dat clustert op basis van overeenkomsten in de data [9]. Fractionele clustering (FC) clustert punten incrementeel door ze te plaatsen in een cluster waar ze de minste fractionele impact hebben.
Het algoritme start met een initiële stap en gaat vervolgens verder met een incrementele stap.
Bij de initiële stap worden eerst alle punten als ongeclusterd gekenmerkt, waarbij alle punten staan voor het maximale aantal punten dat verwerkt kan worden door het werkgeheugen. Vervolgens wordt er een random punt als cluster gekozen. Dan wordt gekeken welke punten hierbij passen door te kijken of ze dichterbij liggen dan afstandslimiet k. Als geen punten meer voldoen aan dit criteria dan wordt het volgende random punt uitgekozen voor de start voor een nieuw cluster. Dit gaat door totdat er geen punten meer over zijn. Elke keer als er een punt bijkomt, wordt k hier ook op aangepast. Waar k op het begin een statische variabele is, wordt deze parameter nu afgesteld aan de hand van de gemiddelde afstand van het cluster.
Elk bij de initiële stap gevonden cluster wordt gerepresenteerd door een set van boxen (cellen in een raster). Elke box heeft hierbij zijn eigen populatie van punten (hoe meer punten, des te hoger de (fractionele) dimensie). Bij de incrementele stap wordt een groep van punten (dat past in het werkgeheugen) gefit op de al bestaande clusters die gemaakt waren in de initiële stap. Voor elk punt in de groep wordt gekeken of hij binnen een bepaalde afstand ligt. Als dit het geval is dan wordt het punt toegevoegd aan het betreffende cluster en zo niet dan wordt dit punt als een uitschieter gezien. Als er te veel uitschieters zijn dan moet het aantal clusters opnieuw worden geëvalueerd en geherstructureerd. Dit gebeurt aan de hand van het splitsen en samenvoegen van clusters.
FC is in staat clusters van verschillende vormen te vinden.
Bij dit algoritme heeft [1] heeft het volgende te zeggen over zijn eigen algoritme: Het voldoet aan alle punten die in paragraaf 3.3.1 zijn beschreven:
Compactheid: Na het maken van de clusters heeft FC de daadwerkelijke punten niet meer nodig. Het gaat namelijk uit van de fractionele dimensie. Populaties van clusters in hogere reeksen worden namelijk berekend door aggregatie van de clusters in de lagere reeksen. De grootte van de representatie hangt niet af van het aantal punten dat al verwerkt is, maar kijkt alleen naar de dimensionaliteit van de set en

de grootte van het kleinste cluster (in de laagste reeks). FC kan goed met hogere dimensies omgaan, maar naarmate het aantal dimensies groeit, zijn er meer boxen nodig voor de representatie. Bij hoge dimensie kan het handig zijn om boxen die een lage populatie hebben te verwijderen. In een experiment van [10] veranderde bijvoorbeeld het resultaat niet na verwijdering van boxen met maar één punt.

Functie: De tijd dat de functie erover doet gaat gepaard met het aantal boxen (deze grootte van boxen hangt samen met de dimensionaliteit van de set). De berekeningen hoeven alleen maar rekening te houden met de gevulde boxen, waardoor de complexiteit laag blijft.

Uitschieters: Als de fractionele impact de afstandslimiet overschrijdt dan wordt dit gezien als een uitschieter.

[10] constateerde het volgende over dit algoritme:

De positieve punten:

- Incrementele structuur (groepen data worden verwerkt in het werkgeheugen).
- FC is van nature eenvoudig uit te breiden, hierdoor is hij altijd klaar voor online opdrachten.
- FC is in staat clusters van verschillende vormen te vinden.
- O(N) complexiteit.

De negatieve punten:

- Afhankelijk van de volgorde van de data.
- Sterk afhankelijk van de initialisatie van de clusters.

Bij de vorige 2 punten wordt verder geen uitleg gegeven, maar het lijkt mij dat als de data erg uiteen loopt (zoals bij twitter het geval is), dat de in het begin gevormde clusters niet heel relevant zijn, waardoor de inkomende data wordt gefit op random clusters.

- Het algoritme is afhankelijk van parameters (gebruikt afstandslimiet tijdens initialisatie).
- Ondanks dat alleen bezette cellen in het geheugen worden bewaard, vormt het geheugengebruik nog steeds een significant probleem.

[31] merkt het volgende op over de methode. Deze methode kan clusters vinden met willekeurige vormen. Het blijkt ook een efficiënt algoritme met het beheer van uitschieter. Hoewel het niet optimaal is voor datastroom clustering, kan het algoritme wel goed gebruikt worden in datastromen met een hoge dimensie van categorische en ordinale data types. Dit algoritme kan niet goed omgaan met tekstdata. Ook [10] zei in een bijzin dat dit algoritme vooral is bedoeld voor numerieke data.

Het is echter niet helemaal duidelijk waarom dit algoritme niet geschikt is voor tekstuele data. Maar twee verschillende artikelen die aangeven dat dit algoritme niet optimaal is voor tekstuele data, is genoeg om dit algoritme te elimineren voor dit onderzoek. Om het risico te vermijden een algoritme te implementeren dat niet werkt, wordt dit algoritme niet gebruikt.

3.3.7 Op dichtheid gebaseerd clusteren (D-Stream)

Bij fractioneel clusteren wordt er gebruik gemaakt van rooster-gebaseerd clusteren. Hier gaan we verder met dichtheid-gebaseerd clusteren. Zowel [14] als [15] hebben een algoritme beschreven wat hierover

gaat. Het D-Stream algoritme, beschreven in [14], zal in deze paragraaf behandeld worden en maakt net als het CluStream algoritme gebruik van een off- en online gedeelte.
Kenmerken van dichtheid-gebaseerd clusteren is dat het willekeurig gevormde clusters kan vinden, de ruis aan kan, het maar één scan nodig heeft om de data te onderzoeken, en in tegenstelling tot k-means geen voorgaande kennis nodig heeft over het aantal clusters.
Het D-Stream algoritme werkt als volgt: Elk document (data-punt) dat binnenkomt, heeft verschillende dimensies. Aan elke dimensie wordt een dichtheidscoëfficiënt toegewezen. Het onlinegedeelte van D-Stream verdeelt per binnenkomend punt al de verschillende dimensies ook weer in verschillende partities. Hierdoor ontstaan er x en y coördinaten (de dimensie en de partitie van de dimensie), die vervolgens in een dichtheidsraster kunnen worden geplaatst. Als dit coördinatenpaar nog niet bestaat, dan wordt het toegevoegd en anders wordt het toegevoegd aan het al bestaande paar. Hierna wordt de dichtheid van het coördinatenpaar bepaald. Dit wordt gedaan door alle punten die aan een coördinatenpaar zijn toegewezen, te sommeren. Hoe langer een coördinatenpaar erin zit, des te minder het gewicht van deze waarde is. De waarde van elk coördinatenpaar wordt alleen geüpdate als er een nieuw punt bijkomt. Tijdens deze update worden naast de waarde ook de andere karakteristieken geüpdate. Door alle ruwe data in dit raster te stoppen hoeft alleen het raster onthouden te worden en niet alle data.
Periodiek wordt er gekeken of er zich uitschieters in het rooster bevinden. Dit is te zien als een coördinatenpaar een te kleine dichtheid heeft. Een te kleine dichtheid kan echter ook voorkomen als er wel veel data inzit maar dat deze data niet recentelijk meer is geüpdate. Als deze verwijderd wordt dan kan dit nadelig uitpakken voor de kwaliteit van de clusters. In het algoritme wordt hier op gelet door middel van een functie die deze twee gevallen kan onderscheiden. Hoe groot de periode is hangt af van de data en is dus variabel. Deze waarde hangt samen met de vervalfactor en wordt berekend in een aparte functie.
Het offline gedeelte clustert alle coördinatenparen. Bij het initiële clusteren worden de coördinatenparen met een hoge dichtheid in een apart cluster gezet en degenen met een te lage dichtheid krijgen het label ‘No_Class’ mee. Voor het bepalen van de dichtheidsgrenzen zijn functies gedefinieerd.
Het dynamisch clusteren gebeurt periodiek. Er wordt hierbij gekeken welke coördinatenparen het dichtst bij elkaar zitten en die worden dan geclusterd. Het coördinatenpaar dat de hoogste dichtheid heeft geeft het label aan het cluster. Het coördinatenpaar dat een te lage dichtheid heeft wordt uit het cluster verwijderd.

De conclusie uit [14] is dat D-Stream het mogelijk maakt om op hoge snelheid datastromen te clusteren zonder te moeten inleveren op clusterkwaliteit. Bij het CluStream algoritme moet er van te voren worden opgegeven wat de tijdsintervallen moeten zijn, maar bij het D-Stream algoritme wordt dit gedaan door de gewichten van de functie te veranderen (vervalfactor). In het artikel vindt ook een vergelijking tussen D-Stream en CluStream plaats. Hieruit blijkt dat D-Stream op het gebied van kwaliteit en snelheid de clusters significant verbetert. Ook kan D-Stream veel beter met hogere dimensies omgaan.

Dit algoritme kan clusters van verschillende vormen ontdekken en kan veel uitschieters rond de clusters detecteren. Dit algoritme kan echter niets met data dat bestaat uit tekst of dat categorisch is [31].

Het is echter niet helemaal duidelijk waarom dit algoritme niet geschikt is voor tekstuele data. Het kan zijn dat dit algoritme data nodig heeft met hetzelfde aantal dimensies. Dit algoritme vergelijkt data namelijk per dimensie. Categorische en tekstuele data moet echter op een andere wijze worden gerepresenteerd. Dit zorgt ervoor dat de data niet per dimensie vergeleken kan worden. Om het risico te vermijden een algoritme te implementeren dat niet werkt, wordt dit algoritme niet gebruikt.

3.3.8 Detecteren van pieken en hiërarchische structuren in datastromen (burst algorithm)

In [16] beschrijft Kleinberg een algoritme dat een datastroom van berichten verwerkt door elke *burst* van activiteit te zien als een toestand en vervolgens deze toestanden op te nemen in het oneindige cellulaire toestands-automaten model. Een *burst* van activiteit is hierbij een sterke toename in een korte tijd van de frequentie van een bepaald onderwerp. Het oneindige van het toestands-automaten model is echter niet relevant. Hiermee zou er een hiërarchische structuur kunnen worden gemaakt, waarmee de intensiteit en periode van de *burst* inzichtelijk wordt gemaakt. Dit is echter meer informatie dan nodig is, aangezien alleen belangrijk is om te weten of het een *burst* is en hoe belangrijk deze *burst* is in vergelijking met andere *bursts*. Daarom wordt er hier alleen nog het 2-toestands-automaten model besproken.

Documenten (bv. Tweets of blogs) komen in een datastroom binnen. Om het algoritme efficiënter te maken worden alle documenten per tijdseenheid in verschillende groepen opgedeeld (bv. elk uur of elke dag). Vervolgens wordt er gebruik gemaakt van een twee-toestand-automaat. Met toestanden q_0 en q_1, die respectievelijk staan voor een lage en een hoge gelijkheidsfractie. Met deze fractie wordt de verhouding van overeenkomstige en irrelevante documenten aangegeven. Als er langdurigere periodes zijn waar de groepen een lage fractie hebben en in een relatief korte periode een hoge, dan is er sprake van een *burst*. Het moment dat er een toestandsovergang plaatsvindt van een lage naar een hoge fractie van *burst*, wordt door de eindige toestandsautomaat gezien als het begin van een *burst*.

De basis toestand van de automaat heeft een kans p_0 dat van toestand veranderd wordt:

$$p_0 = \frac{R}{D} \quad (6)$$

Hierbij is:
R de som van alle relevante documenten van de groep.
D de som van alle documenten in deze groep.

Ook voor toestand q_1 bestaat er een verwachte fractie van relevante documenten p_1. Voor de basis toestand p_0 = R/D en $p_1 = p_0\, s$ (waar s de schalingsparameter is).
Nu produceert toestand q_i relevante en irrelevante berichten volgens een binomiale verdeling.

De kosten van een toestandsovergang q = (q_1,….., q_n) wanneer de t^{de} groep arriveert en wanneer de automaat in staat q_i is:

$$\sigma(i, r_t, d_t) = -\ln\left[\binom{d_t}{r_t} p_i^{r_t} (1 - p_i)^{d_t - r_t}\right] \quad (7)$$

Als er gegeven is dat er een *burst* bekend is (het toestands-automaten model bevindt zich in q_1) dan kan de belangrijkheid van de *burst* worden aangegeven door het gewicht uit te rekenen:

$$\sum_{t=t_1}^{t_2} (\sigma\ (0, r_t, d_t) - \sigma(1, r_t, d_t)) \quad (8)$$

Door alle *bursts* te rangschikken op gewicht, worden de belangrijkste trends geordend. Dit algoritme is ook succesvol toegepast door Fekkes [17]. Hij zegt dat de benadering van Kleinberg bewezen heeft erg snel, robuust en efficiënt te zijn.

3.3.9 OCTS & OCTSM

OCTS [32] is een algoritme dat tekst datastromen clustert. Dit algoritme bestaat uit een offline initialisatie proces en wordt gevolgd door een online clusteringsproces (dit is gebaseerd op het CluStream van paragraaf 3.3.5). Het offline proces begint met het maken van frases van de inputdata met behulp van een externe tool genaamd Xtract [34]. Hiermee kan het 'translation model' en het 'language model' en dus ook het extended Semantic smoothing model (zie paragraaf 3.1.2) gemaakt worden. Met het eerste gedeelte van de inputstroom worden de initiële clusters gemaakt. Dit gebeurt aan de hand van clusterprofielen. In een profiel zit een 'translation model', een 'language model', een sommatie van de waardes van alle frase sleutels (een woordcombinatie dat een trend typeert) en een tijdstip van wanneer het cluster als laatst is bijgewerkt. Het aantal clusters dat gemaakt wordt moet van te voren worden opgegeven. Dit geldt ook voor de gelijkheidskans die als drempelwaarde voor een nieuw cluster dient.
Het onlineproces wordt elke keer uitgevoerd als er weer een document binnenkomt. Als eerst wordt er een 'fading'-functie toegepast op alle clusters. Deze functie zorgt ervoor dat een cluster in waarde afneemt als deze niet wordt bijgewerkt. Vervolgens worden de frases uit het document gehaald om het document te representeren. Hierna wordt voor elk cluster *j*, $p_{bt}(w|c_j)$ uitgerekend. Dit geeft de kans weer dat een woord *w* bij cluster c_j hoort. Deze kans wordt gebruikt om de gelijkenis $p(d|c_j)$ tussen elk cluster en het document te berekenen.
Als vervolgens de gelijkenis van het cluster waarmee het document het meest overeenkomt kleiner is dan de drempelwaarde, dan wordt de meest inactieve cluster verwijderd en wordt er een nieuw cluster gecreëerd. Mocht de gelijkenis groter zijn dan de drempelwaarde dan wordt het clusterprofiel bijgewerkt met dit document.

Het nadeel van dit algoritme is dat te snel inactieve clusters (historische data) worden weggegooid. Vandaar dat [32] een uitbreiding hebben gemaakt genaamd OCTSM. Het verschil met OCTS is dat er nu een extra 'merge'-stap is toegevoegd. Op het moment dat een document wordt toegevoegd wordt er eerst gekeken of de bestaande clusters erg op elkaar lijken. Mocht dit het geval zijn dan worden deze

clusters samengevoegd. Hierdoor wordt het clusteringsproces meer stabiel en krijgt de historische data nog een tweede kans.

Het nadeel van dit algoritme is dat er 3 inputvariabelen zijn en dat er niet in een specifieke tijd gekeken kan worden. Alleen de recentste clusters kunnen bekeken worden. De oudere clusters zijn namelijk al weggegooid.

3.3.10 Welk clusteringsalgoritme is het meest geschikt voor twitterberichten

In deze paragraaf zullen alle hiervoor besproken algoritmes tegenover elkaar worden gezet. Dit zal gebeuren aan de hand van alle specificaties die in de eerste paragraaf zijn opgesomd. Een overzicht hiervan is gegeven in tabel 2. Hierbij zijn de specificaties gerangschikt op relevantie. De specificaties in het donkere gedeelte zijn, in tegenstelling tot het lichtere gedeelte, noodzakelijk. N.B. de beoordelingen in tabel 2 zijn gebaseerd op informatie gehaald uit de in dit hoofdstuk genoemde artikelen. De genoemde specificaties zijn niet altijd expliciet beschreven en zijn daarom gebaseerd op de impressie die ik hieruit kreeg.

	Birch	Cobweb	Stream	CluStream	FC	DC	BA	OCTSM
Algoritme kan met tekst (grote dimensies) omgaan.	-	?	?	+	-	-	+	+
Er kunnen snel en incrementeel nieuwe datapunten verwerkt worden (efficiënt genoeg om elke binnenkomende tweet te verwerken).	-	+/-	-	+	+	+	+	+
Compacte representatie	-	-	+	+	+	+	+	+
Er wordt rekening gehouden met de tijd (nieuwe bestanden krijgen hogere prioriteit).	?	-	-	+	?	+	+	+
Er is een duidelijke en snelle manier om uitschieters en ruis te identificeren.	+	+	-	-	+	+	+	-
Resultaat onafhankelijk van de volgorde van de input.	-	+	?	?	-	-	+	?
Er kunnen verschillende vormen clusters ontdekt worden.	-	?	-	-	+	+	n.v.t.	-
Er kan teruggekeken worden naar historische data.	+	+	+	+/-(1)	?	+/-(1)	+	-
Onafhankelijk van parameters.	?	+	?	-	-	+	+	-

(1) Het is wel mogelijk om historische data te bekijken alleen is dit beperkt, doordat deze data gedeeltelijk wordt gefilterd om ruimte te besparen.

Tabel 2. Een samenvatting van alle specificaties ten opzichte van alle algoritmes. Met een + wordt aangegeven dat het algoritme aan de specificatie voldoet en een – staat voor het niet voldoen aan de specificatie.

Wat vooral opvalt in het bovenste gedeelte van de eerste drie kolommen is het spervuur aan minnetjes. Hierbij komt ook nog het feit dat het Clustream-algoritme heeft aangetoond beter te presteren dan deze

algoritmes. Birch, Coweb en Stream vallen daarom ook af. Het op dichtheid gebaseerd clusteren met het D-Stream is een erg veel belovende methode. Dit algoritme beweert beter te presteren dan het Clustream en heeft dit vooral te danken aan de precisie waarmee de clusters gevormd kunnen worden. Dit is waarschijnlijk ook het beste algoritme beschreven in deze scriptie. Echter gaf [31] aan dat dit algoritme niet met tekst kan omgaan, waardoor dit algoritme afvalt. Hetzelfde geldt voor het fractioneel clusteren. Ook dit algoritme kan niet met tekst omgaan, waardoor het niet gebruikt kan worden in dit onderzoek. OCTSM is gebaseerd op CluStream en ze hebben beide het nadeel dat de uitschieters minder goed geïdentificeerd kunnen worden omdat er gebruikt wordt gemaakt van het k-means algoritme. Het voordeel van OCTSM is dat deze methode is toegepast op een tekstdatastroom. Er wordt hier rekening gehouden met de context van de tekst. Het Burst algoritme voldoet aan alle eisen, maar kan eigenlijk niet een op een vergeleken worden met de andere algoritmes. Dit algoritme identificeert alleen trends en is geen clusteralgoritme. Als er een frequentie van een woord in vergelijking met het verleden opeens erg omhoog schiet dan wordt dit gezien als een trend. Hierdoor kunnen verschillende woorden eenvoudig als trends worden geïdentificeerd. Het nadeel is dat er woorden in plaats van onderwerpen als trends worden geïdentificeerd. Als er bijvoorbeeld twee grote onderwerpen tegelijk erg populair worden, dan zullen woorden van deze twee trends door elkaar heen gaan lopen. Hierdoor zullen uiteindelijk de sentimenten per onderwerp ook niet goed kunnen worden geïdentificeerd. Ook moet er, om te kunnen identificeren of er een piek in de frequentie van een woord voorkomt, historische data zijn waar dit uit moet blijken. Hoeveel en hoe lang er tweets verzameld moeten worden is niet helemaal duidelijk. Bij de implementatie van Fekkes [17] wordt het algoritme op blogs toegepast en wordt er voor elke blogger een top 100 van meest voorkomende woorden opgeslagen. Op deze manier wordt er data gecreëerd, die de gemiddelden van de woordfrequenties aangeven. Dit is een proces dat erg veel tijd kost en ruimte op de harde schijf in beslag neemt. Als dit algoritme later, naast politiek, uitgebreid wordt naar andere segmenten, moet dit proces wellicht meerdere keren herhaald worden aangezien de historische data van deze groep twitteraars wellicht weer anders is. Een voordeel van dit algoritme is dat Fekkes het algoritme bij Parabots heeft gemaakt, waardoor de code hierdoor al beschikbaar is. Echter is het van groot belang dat de trends wel goed geïdentificeerd kunnen worden, waardoor er toch voor het OCTSM algoritme gekozen is. De representatie van het OCTSM model neemt de context van de tekst in ogenschouw, waardoor er duidelijke trends ontstaan. De sentimenten van gedetecteerde trends worden ook goed vastgelegd aangezien door middel van het ‘semantic smoothing model’ de relatie van de trend met andere woorden kan worden vastgesteld. Hoe dit algoritme wordt geïmplementeerd is beschreven in hoofdstuk 4.

3.4 Clusterevaluatie/validatie

Als er eenmaal een clusteringalgoritme is geïmplementeerd, dan is het vervolgens de vraag of dit algoritme wel doet wat hij moet doen. Clustert het algoritme dus wel goed genoeg. Om dit subjectieve begrip te kwantificeren, is er een evaluatietechniek nodig. In de literatuur zijn er een aantal artikelen (bv. [18-22]) die clusteringvalidatietechnieken beschrijven. Het komt in deze artikelen telkens neer op de volgende categorieën van validatie: interne-, externe- en relatieve-validatie:

- De interne-validatie is gebaseerd op berekende eigenschappen van de gevormde clusters. Hierbij wordt gekeken naar de compactheid, scheiding en de vorm van de clusters. Het heet interne

validatie omdat er geen additionele informatie nodig is over de data. Een typische vraag die bij deze validatie beantwoord wordt, is de volgende:
    - Passen de geïdentificeerde clusters goed bij de data?
- De relatieve-validatie is gebaseerd op het vergelijken van clusters gegenereerd door het algoritme op basis van verschillende parameters of gedeeltes van de dataset. Ook bij deze soort validatie is er geen additionele informatie nodig. Een typische vraag die bij deze validatie beantwoord wordt, is de volgende:
    - Hoe goed komen de uiteindelijke clusters overeen als het algoritme op een ander gedeelte van de dataset wordt toegepast?
- De externe-validatie vergelijkt de gevormde clusters met de daadwerkelijke labels van de clusters. Hierbij is er dus wel additionele informatie nodig, aangezien van te voren bekend moet zijn tot welke klasse een bepaald item behoort. Deze validatietechniek zit erg dicht bij de daadwerkelijke error van het algoritme. Een typische vraag die bij deze validatie beantwoord wordt, is de volgende:
    - Komen de uiteindelijke clusters overeen met de vooraf bekende kennis over het probleem?

Methodes die bij deze verschillende vormen van validatie horen komen in de rest van het hoofdstuk aan bod.

3.4.1 Interne-validatie

Bij interne-validatie wordt getest hoe goed de geïdentificeerde clusters bij de data passen en wordt alleen gekeken naar informatie die inherent is aan de data. De technieken die hier onder vallen zijn de Davies-Bouldin index, de dunn index, de silhoutte index. Hier wordt vooral getest hoe compact en goed verspreid de cluster zijn. [31] geeft aan dat dit het belangrijkste probleem is bij de kwaliteit van clusters.

Davies-Bouldin Index

De Davies-Bouldin Index (DB) [23] meet hoe compact en goed verspreid de clusters zijn. Hierbij moet de spreidingsmaatstaf voor elke cluster zo klein mogelijk zijn terwijl de verschilmaatstaf tussen de clusters juist zo groot mogelijk moet zijn. Hoe deze maatstaven eruit zien, volgt hieronder:
Verspreidingsmaatstaf s_i van cluster C_i, waarbij n het aantal elementen in C_i is, x de waarde van het element in cluster C_i en $\overline{x_i}$ is het gemiddelde van alle waardes in cluster C_i:

$$s_i = \sqrt{\frac{1}{n}\sum_{x \epsilon C_i} |x - \overline{x_i}|^2} \quad (9)$$

Verschilmaatstaf d_{ij} van cluster C_i en cluster C_j in een l-dimensionale ruimte, met $\overline{x_{jk}}$ als gemiddelde van cluster C_j met element *k*:

$$d_{ij} = \left|\overline{x_i} - \overline{x_j}\right| = \sqrt{\sum_{k=1}^{l} |\overline{x_{jk}} - \overline{x_{ik}}|^2} \quad (10)$$

Gelijkheidsindex:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \quad (11)$$

$$R_i = \max_{j=1,..,m,i=1,...,m,i \neq j} R_{ij}$$

Kleine waardes van DB geeft aan dat er compacte en goed verspreide clusters aanwezig zijn:

$$DB_m = \frac{1}{m}\sum_{i=1}^{m} R_i \qquad (12)$$

Dunn index

De Dunn index meet hetzelfde als de Davies-Bouldin Index: hoe compact en goed zijn de clusters verspreid. Hoe hoger de Dunn index hoe beter dit van toepassing is. Hierbij wordt ook weer een vergelijking gemaakt tussen de verschilmaatstaf en de verspreidingsmaatstaf:

Verschilmaatstaf $d(C_i, C_j)$ tussen cluster C_i en cluster C_j in een l-dimensionale ruimte:

$$d(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y) \quad (13)$$

En een verspreidingsmaatstaf in de vorm van een diameter:

$$diam(C) = \max_{x,y \in C} d(x, y) \quad (14)$$

Dit resulteert uiteindelijk in de volgende Dunn index:

$$D_m = \min_{i=1,..,m} \{ \min_{j=i+1,..,m} \left(\frac{d(C_i, C_j)}{\max_{k=1,...,m} diam(C_k)} \right) \} \quad (15)$$

Hierbij is d(x,y) de euclidische afstand tussen punten x en y.

Silhouette index

Het silhouet van een cluster werd in eerste instantie gebruikt om visueel te laten zien welke punten wel en welke punten niet in een cluster lagen [24]. Dit werd gebaseerd op een clusterbreedte.

De silhouetbreedte van een bepaald punt definieert de nabijheid van haar eigen cluster *j* ten opzichte van de nabijheid van andere clusters. De silhouetbreedte van een gegeven punt *i* is als volgt gedefinieerd:

$$S(i) = \frac{b(i) - a(i)}{\max [b(i), a(i)]} \qquad (16)$$

Hierbij staat *a(i)* voor de gemiddelde afstand tussen i en alle andere punten in het cluster en *b(i)* is hierbij het minimum van de gemiddelde afstand tussen *i* en de punten van de andere clusters.

Het silhouet van een cluster *j* is gedefinieerd als de gemiddelde silhouetbreedte van alle punten in het cluster:

$$S_j = \frac{1}{m}\sum_{i=1}^{m} S(i) \qquad (17)$$

Aggregatie van alle breedtes geeft de globale silhouet index aan (*GS*). Dit is gemiddelde silhouet van alle clusters. Deze luidt als volgt:

$$GS = \frac{1}{c}\sum_{j=1}^{c} S_j \qquad (18)$$

Voor een gegeven punt x is de silhouetbreedte een bereik van -1 tot 1. Als de *GS* dichterbij -1 zit dan wilt dit zeggen dat het punt *i* gemiddeld dichter bij een ander cluster zit. Als de *GS* echter dichterbij de 1 zit

dan zit punt *i* gemiddeld wel dichter bij zijn huidige cluster dan elk ander cluster. Dus hoe hoger het silhouet des te compacter en verspreider de clusters zijn.

3.4.2 Relatieve-validatie

De interne indices uit de vorige paragraaf geven alleen de evaluatie van een enkele realisatie van een clusteringalgoritme, waardoor er geen indicatie over de stabiliteit van het algoritme kan worden gegeven als er variatie in de data bestaat. Ook wordt er niet op inconsistentie getest, dat kan ontstaan door overbodige data. Relatieve-validatie indices proberen hier wel op te testen door de clusters onder verschillende condities te vergelijken. Eerst zal de 'Figure of Merit' worden onderzocht en daarna zal er iets over de instabiliteitindex worden gezegd.

Figure of Merit

De 'Figure of Merit' (FOM) [25] gaat ervan uit dat er zich overbodige data bevindt in de data die gebruikt wordt om de consistentie te testen. Heeft één bepaalde feature dus juist veel impact op het cluster of voegen alle features een beetje hetzelfde toe aan het uiteindelijke gemaakte cluster? De FOM test per feature hoe groot de impact op het cluster is. Hierbij wordt er meerdere keren het clusteringalgoritme gedraaid, maar elke keer wordt er telkens één feature weggelaten. Elke keer wordt er ook bijgehouden wat de grootte is van het cluster (de gemiddelde afstand tussen alle samples en de bijhorende clustercentra). Als deze gemiddelde afstand klein is, dan wilt dit zeggen dat deze feature een kleine impact heeft, waardoor het algoritme consistent lijkt te zijn. Dit kan pas echt bepaald worden als alle features één voor één worden weggelaten en vervolgens de totalen hiervan gesommeerd worden.

Een lage FOM wilt dus zeggen dat het algoritme een hoge stabiliteit heeft. Echter wordt de FOM prestatie gebaseerd op gesimuleerde data dat meestal lager presteert dan simpele indices (inclusief interne indices). Hierbij komt ook nog dat dit algoritme meerdere keren herhaald moet worden, waardoor het niet vaak als validatie techniek wordt gebruikt [26].

Instabiliteitindex

De instabiliteitindex meet het vermogen van een geclusterde dataset om de clustering van een andere dataset van dezelfde bron te voorspellen [27].
Het meten van de stabiliteit gaat als volgt: als eerst wordt de set van te clusteren punten verdeeld in twee groepen. Het clusteringalgoritme dat onderzocht wordt, wordt op de eerste groep toegepast. De labels die hierdoor geproduceerd worden, worden gebruikt om een classificatiemodel te trainen over de gehele data. Nu wordt dit algoritme en het originele clusteringalgoritme op de tweede datagroep toegepast. Hierdoor worden er twee sets met labels gegenereerd. De onenigheid tussen deze labels, gemiddeld over herhaalde willekeurige partities van de punten, definieert de instabiliteit van het clusteringalgoritme.
De instabiliteitindex hangt af van het aantal clusters en moet daarom genormaliseerd worden als modelselectie wordt toegepast [27]. De normalisatie wordt verkregen door te delen door de instabiliteit verkregen bij gebruik van een willekeurige schatter als het classificatiemodel.
Een andere factor waar de instabiliteitindex vanaf hangt is de selectie van de classificatieregel. Deze kan namelijk sterk de resultaten beïnvloeden [26, 27]. Ook tonen andere simulatiestudies aan dat deze index

niet beter presteert dan andere relatieve en interne indices. Hierbij komt ook nog dat het één van de meest tijdrovendste indices is, omdat de applicatie meerdere keren herhaald moet worden en een classificatiemodel moet worden getraind.

3.4.3 Externe validatie

Goede interne en relatieve validatiescores vertalen zich nog niet noodzakelijkerwijs in een effectieve applicatie. Externe validatie vergelijkt de waardes die uit het algoritme komen met de waardes die aan desbetreffende items zijn toegekend. Een dataset bestaat dan uit mogelijk verschillende variabelen en een label dat aangeeft bij welke klasse het betreffende item hoort. In een ideale situatie wordt de toekenning van een dergelijk label door een mens bepaald. Deze paragraaf zal vier technieken van externe validatie toelichten. Als eerst zal Purity worden beschreven gevolgd door NMI, de rand index en de F-measure. De bron voor deze informatie is [W14].

Purity

Purity geeft de 'zuiverheid' van de gegeven clusters weer. Om de zuiverheid te berekenen, is elk cluster toegewezen aan de klasse die het meest frequent in het cluster voorkomt. Vervolgens wordt de juistheid van deze opdracht gemeten door het tellen van het aantal correct toegewezen documenten en te delen door het totale aantal items *N*. De formule die hieruit voortkomt is de volgende:

$$purity(\Omega, C) = \frac{1}{N}\sum_{k} \max_{j} |w_k \cap c_j| \qquad (19)$$

Hierbij is $\Omega = \{w_1, w_2, \dots, w_K\}$ een set van clusters, $\mathrm{C} = \{c_1, c_2, \dots, c_j\}$ een set van klassen, w_k is een set van documenten in klasse *k* en c_j een set van documenten in cluster *j*.

Slechte clustering resulteert in waardes dichtbij 0 en een perfecte clustering heeft een purity van 1. Een hoge zuiverheid is echter eenvoudig te behalen door het aantal clusters erg groot te maken. Als bijvoorbeeld elk item zijn eigen cluster zou krijgen dan is de purity 1, terwijl er eigenlijk helemaal geen clustering heeft plaatsgevonden. De kwaliteit van het clusteren wordt op deze manier dus niet gemeten.

NMI

Normalized mutual information (NMI) geeft een uitweg voor de problemen waar de purity-meeteenheid tegenaan loopt. De formule ziet er als volgt uit:

$$NMI(\Omega, C) = \frac{I(\Omega, C)}{[H(\Omega) + H(C)]/2} \qquad (20)$$

Hierbij is *I* de wederzijdse informatie en H de entropie [W15]. *I* wordt berekend met de volgende formule:

$$I(\Omega, C) = \sum_{k}\sum_{j} P(w_k \cap c_j)\log \frac{P(w_k \cap c_j)}{P(w_k)P(c_j)} \qquad (21)$$

$$= \sum_{k}\sum_{j} \frac{|w_k \cap c_j|}{N}\log \frac{N|w_k \cap c_j|}{|w_k||c_j|} \qquad (22)$$

Met $P(w_k)$, $P(c_j)$ en $P(w_k \cap c_j)$ als de kans dat een document respectievelijk in cluster w_k, in klasse c_j en in doorsnede van w_k en c_j zit.

H wordt berekend met de volgende formule:

$$H(\Omega) = -\sum_k P(w_k) \log P(w_k)$$
$$= -\sum_k \frac{w_k}{N} \log \frac{w_k}{N} \quad (23)$$

Bij zowel *H* als *I* is de tweede vergelijking een maximum likelihood schatting van de kansen. De schattingen zijn in andere woorden, de relatieve frequenties.

$I(\Omega, C)$ meet de hoeveelheid informatie waarmee onze kennis over de klassen toeneemt wanneer we te horen krijgen wat de clusters zijn. Het minimum van $I(\Omega, C)$ is 0. Dit gebeurt als de clustering random is. In dit geval geeft het model geen nieuwe informatie over de classificatie van een document aan een klasse. Ook hier wordt een maximale waarde bereikt als het aantal clusters gelijk is aan het aantal punten. Vandaar dat er ook nog gedeeld moet worden door $[H(\Omega) + H(C)]/2$. De NMI waarde wordt hierdoor genormaliseerd. Hierdoor wordt er dus ook rekening gehouden met het aantal clusters. Ook hier variëren de waardes van 0 tot 1.

Rand index

De rand index beschouwt het clusteringsproces als een reeks van beslissingen. Hierbij wordt gekeken naar het aantal correcte paren in alle clusters in vergelijking met het totaal aantal paren in de collectie (N(N-1)/2). Twee documenten moeten alleen aan een cluster worden toegevoegd als de twee documenten dezelfde klassen hebben. Een algoritme kan de volgende beslissingen maken:

- TP (true positive): twee documenten van dezelfde klasse worden aan hetzelfde cluster gekoppeld.
- TN (true negative): twee documenten van verschillende klassen worden aan verschillende clusters gekoppeld.
- FP (false positive): twee documenten van verschillende klassen worden aan hetzelfde cluster gekoppeld.
- FN (false negative): twee documenten van dezelfde klasse worden aan verschillende clusters gekoppeld.

De rand index meet hoeveel procent van de beslissingen er correct zijn genomen:

$$RI = \frac{TP + TN}{TP + FP + FN + TN} \quad (24)$$

F-measure

F-measure is een uitbreiding op de rand index. De rand index geeft gelijk gewicht aan false positives en false negatives. De scheiding van soortgelijke documenten is soms 'erger' dan het samenvoegen van paren van ongelijke documenten. Hiervoor kan er gebruik gemaakt worden van de F-measure. Deze maatstaaf bestraft false negatives sterker dan false positives door het selecteren van een $\beta > 1$. Dit gebeurt in de volgende formules:

$$P = \frac{TP}{TP+FP} \qquad R = \frac{TP}{TP+FN} \qquad F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P+R} \quad (25)$$

3.4.4 Validatie van OCTSM

De hiervoor besproken methodes van de interne en relatieve validatie geven een goede indicatie van de kwaliteit van de clusters maar geven geen indicatie of het algoritme wel de clusters eruit haalt die eruit gehaald moeten worden. De bedoeling van het algoritme is om de trends die zich voordoen te detecteren. Echter is van te voren niet bekend hoe een trend eruit ziet. De data is namelijk erg groot en erg variërend. Het is daarom zowat onmogelijk om zelf te bepalen wat de trends van de dataset zijn. De resultaten die er uit het algoritme komen, kunnen er goed uitzien maar kunnen zomaar totaal incorrect zijn. Om er achter te komen wat de klasse is van een bepaalde tweet wordt er naar de hashtag gekeken. Op deze manier is het mogelijk om externe validatie toe te passen. Hoe het maken van de dataset precies in elkaar zit en waarom er naar hashtags wordt gekeken, wordt uitgelegd in paragraaf 5.1.

Welke externe validatiemethode?

De oorspronkelijke makers van OCTSM [32] hebben destijds de NMI methode gebruikt voor de validatie van het algoritme. In deze scriptie is niet voor deze methode gekozen. NMI houdt ook rekening met de spreiding van de documenten. Bij het clusteren van tweets met het OCTSM algoritme is dit minder relevant. Stel dat er bijvoorbeeld zes clusters zijn die allemaal rond de 20 tweets van dezelfde klasse bevatten. Bij de NMI methode resulteert dit in een slechte score. Dit hoeft echter niet slecht te zijn. In een later stadium kan het namelijk zo zijn dat deze clusters bij elkaar worden gevoegd. Ook al zouden deze clusters zo groot worden dat ze als een trend worden opgemerkt, dan is dit geen ramp. Het is beter dat een klasse zich in meerdere clusters bevindt, dan dat er twee verschillende klassen in een cluster worden gestopt. Als twee clusters met verschillende klassen eenmaal bij elkaar zijn gevoegd, is er een kans dat deze erg verschillende woorden verzamelt. Hierdoor ontstaat er een cluster uit woorden waar het verband niet in te zien is terwijl dit 'onderwerp' wel als een trend wordt aangezien.

Ook de F-measure meet de spreiding van de clusters, maar het instellen van de $\beta < 1$ zorgt ervoor dat er minder streng wordt gekeken naar de false positives en strenger naar de false negatives. Met andere woorden: het is beter om twee tweets van dezelfde klasse in een ander cluster te hebben dan twee verschillende in hetzelfde cluster. Als er namelijk twee verschillende klassen in een cluster zitten, dan betekent dit dat er een cluster is ontstaan met woorden die niet bij elkaar horen. Als dit cluster ook nog groot wordt, ontstaat er een trend met woorden uit verschillende trends die dus niets met elkaar te maken hebben.

Om alleen de F-measure toe te passen is ook niet voldoende. Het komt namelijk nogal eens voor dat bij bepaalde instellingen van de parameters van OCTSM, het algoritme bijna niet tot het samenvoegen van clusters of toevoegen van tweets toekomt. Hierdoor bestaat het eindresultaat uit clusters die niet veel tweets bevatten, terwijl de F-measure nog steeds een goede waarde geeft. Daarom is als maatstaf de volgende formule gekozen:

$$aantal\ geclusterde\ tweets * Fmeasure$$

Hierbij is gekozen voor $\beta = 0.5$. Als $\beta = 0$ zou zijn dan wordt er helemaal niet gekeken naar de false negatives. Hier is niet voor gekozen, omdat het wel beter is dat de klassen niet helemaal verspreid zijn. 0.5 is net genoeg om meer negatief gewicht te geven aan de false positives.

Ontdekking van trends

De hiervoor beschreven maat meet het aantal correct geclusterde tweets. De zuiverheid van het cluster wordt op deze manier ook gemeten. Hierdoor is er nog geen indicatie hoeveel trends er nu eigenlijk ontdekt zijn. In paragraaf 5.2 worden de parameters van het OCTSM-algoritme geoptimaliseerd. Bij deze optimalisatie moeten er veel testen worden gerund. Hiervoor kan een snelle indicatie van de trends erg handig zijn.

Wat is een trend?

Een trend is een onderwerp dat bestaat uit een of meerdere woorden, waar meer dan andere onderwerpen over wordt getweet. Hoe meer er over een trend getweet wordt des te belangrijker deze trend is.

Meetbaarheid van trends

Als er in het OCTSM-algoritme al een paar tweets bij elkaar worden geclusterd, betekent dit dat hij een onderwerp heeft gevonden. Als deze tweets ook nog van dezelfde klasse zijn dan kan er gesteld worden dat een trend gevonden is. Om een indicatie te geven van hoeveel trends er gevonden zijn is er het volgende bedacht: als een cluster meer dan vijf tweets van eenzelfde klasse bevat, wordt dit beschouwd als een detectie van een trend. Mocht het zo zijn dat er meerdere clusters meer dan vijf tweets van eenzelfde klasse bevatten, dan wordt dit nog steeds beschouwd als een detectie van één klasse. Het kan immers nog steeds zo zijn dat deze later worden samengevoegd. Deze maat let dus niet op het feit of er ook nog veel tweets van een andere klasse zich in het cluster bevinden. Hier is immers de maatstaf in het hiervoor besproken stukje voor.

4 De implementatie van OCTSM op een stroom van tweets

Uit de analyse van het vorige hoofdstuk is gebleken dat het OCTSM algoritme [32] de beste keuze is om trends uit een stroom van tweets te halen. De representatie van het OCTSM model neemt de context van de tekst in ogenschouw, waardoor er gemakkelijker trends ontdekt kunnen worden. De sentimenten van de trends worden waarschijnlijk ook goed vastgelegd aangezien door middel van het 'semantic smoothing model' de relatie van de trend met andere woorden kan worden vastgesteld. Hoe het OCTSM model precies wordt geïmplementeerd wordt in dit hoofdstuk beschreven. In de eerste paragraaf zal er gekeken worden hoe de binnenkomende tweets worden verwerkt zodat het OCTSM model er mee kan werken. Hierna kunnen de bewerkte tweets dienen als invoer voor respectievelijk het off- en online proces. Deze twee fasen zullen in de hier opvolgende paragrafen worden beschreven. Uiteindelijk zal er in de laatste paragraaf besproken worden tegen welke problemen er aangelopen werd.

4.1 Tweets verwerken tot correcte invoer

In een tweet kunnen verschillende onderwerpen zitten en een onderwerp kan gedefinieerd zijn door meerdere woorden. De vraag die in deze paragraaf beantwoord moet worden luidt: hoe haal ik uit een tweet de belangrijkste woorden? Om deze vraag te beantwoorden moet een tweet drie verschillende processen doorstaan:

- Een tweet moet bewerkt worden door een tokenizer.
- Relevante woorden en frases moeten gedetecteerd worden.
- Van alle woorden uit een tweet moet de stam bepaald worden.

4.1.1 Tokenizer

Een tokenizer is een algoritme dat bepaalde punctuatie uit een zin haalt, waardoor woorden kunnen worden onderscheiden. Dit is de eerste stap in het ontleden van een tweet. Als dit niet zou gebeuren dan zouden de woorden niet als afzonderlijk kunnen worden gedetecteerd. Een woord wordt namelijk gedetecteerd doordat er twee spaties (inclusief begin en eind van de zin) omheen staan. Een voorbeeld van een tweet waar een dergelijk algoritme op toegepast kan worden is de volgende:

#PSV-#Spakenburg slechts 3-0. Maar goed dat #Feyenoord niet tegen Spakenburg hoefde te spelen... ;)

Moeten alle leestekens weg?

Het is niet verstandig belangrijke twitter-tekens als '#' en '@' te verwijderen. Als woorden deze tekens bevatten dan gaat het juist over onderwerpen en personen die erg belangrijk zijn bij het definiëren van een trend. Andere tekens zoals `[] {} ? > < , . ' " ; : * & ^ % $ !` hebben echter geen speciale betekenis in twitter. In de reguliere Nederlandse taal is het echter wel mogelijk dat woorden gevormd worden door een punt, een streepje of een apostrof. Voorbeelden zijn V.S., Zuid-Korea, zee-eenden, 's avonds en bio-industrie. Als de leestekens weggehaald zouden worden dan zou dit kunnen leiden tot twee verschillende woorden die apart iets anders betekenen. Twitter is echter een platform waar correct Nederlands niet altijd wordt nagestreefd. V.S. word al gauw gespeld als 'vs'. Daarbij komt een punt of een apostrof niet erg vaak voor. Het komt echter wel veel voor dat een '-' als tussenvoegsel wordt gebruikt. Dit wordt echter niet consequent door elke twitteraar gedaan. Als het cluster dan

bijvoorbeeld Zuid-Korea tegenkomt en vervolgens zegt iemand “Zuid Korea”, dan wordt dit niet als een dubbelde hit op zuid-korea gezien, maar op een hit op ‘zuid’, ‘korea’ en ‘zuid-korea’. Bij *#PSV-#Spakenburg* is het veel belangrijker om te weten dat het onderwerp over #PSV en #Spakenburg gaat dan *#PSV-#Spakenburg*, dat wederom bijna nooit voorkomt. De woorden komen wel in combinatie met elkaar voor, waardoor dit wel een topic kan worden als er genoeg over getweet wordt. Dit woord krijgt echter bijna geen waarde als het woord aan elkaar blijft, omdat dit woord niet vaak gebruikt wordt. Hetzelfde geldt eigenlijk voor de andere voorbeelden. Als er echter een ‘#’ de eerste letter is (waarmee het onderwerp wordt aangegeven), dan zou dit kunnen leiden tot een heel ander onderwerp. Als dit woord gesplitst wordt dan hoort het tweede gedeelte namelijk niet meer tot het onderwerp. Het komt echter bijna niet voor dat een onderwerp een ‘-’ bezit. Een woord dat een hashtag bevat is meestal zo geschreven dat er geen punctuatie in voorkomt. Hierdoor kan het makkelijker overgenomen worden, waardoor meerdere mensen deze hashtag gaan gebruiken. Er hoeft dan nooit nagedacht worden wat er nu tussen de woorden moet komen. Een voorbeeld: het wordt niet #Vittesse-Feyenoord, maar #vitFey.
Een uitzondering wordt gemaakt voor woorden die beginnen met een ‘@’ (naam van een account van een persoon op twitter). Door een verandering in een twitter-naam zou er namelijk naar een heel ander persoon verwezen kunnen worden. Hierdoor wordt er voor twitter-namen een uitzondering gemaakt.

Naast alle leestekens, worden de volgende tekstgedeeltes ook verwijderd:

- Urls
- “rt @name” Dit geeft een retweet aan en zegt niets over het onderwerp. Echter wordt “@name” niet verwijderd. Het komt namelijk nog wel eens voor dat het onderwerp bijvoorbeeld wilders is en dat er in de tweet staat @wilders.
- Alle woorden met maar een letter.
- Cijfers die een bedrag vormen of een tijd (bijvoorbeeld $23,12; 45.50; 12:30 of 12 uur). Jaartallen of cijfers boven de 9 zijn wel behouden gebleven. De eerst genoemde voorbeelden kunnen op erg veel onderwerpen betrekking hebben, terwijl normale cijfers nog wel eens specifiek bij een bepaald onderwerp kunnen horen. Voorbeeld: “het wordt even wennen om bordjes met 130 naast de weg te zien staan”, “de 11 steden tocht wordt niets dit jaar” of “ 21 doden bij vliegtuig ramp”.
- Woorden met als combinatie een van de letters ‘h’ en ‘o’ en met de combinatie de letters ‘h’ en ‘a’ worden ook verwijderd. Als mensen wat grappig vinden dan tonen ze dit vaak door te beginnen met bijvoorbeeld hahaha, haha of hhahahahaa. Een ander stopwoord is oh, oooh of ohh. Dit wordt meestal gebruikt bij een reactie op iemand anders (bijvoorbeeld ‘oh zo’, ‘ohh dat meen je niet’ of ‘oh dat zal je niet zeggen’). Deze woorden worden hier specifiek genoemd omdat deze niet door de Brill tagger uit de volgende paragraaf worden geïdentificeerd (deze paragraaf bespreekt de verdere verwijderingen van stopwoorden.

Als laatste zullen alle hoofdletters naar kleine letters worden geconverteerd. De tweet die aan het begin werd getoond ziet er na de tokenizer als volgt uit:

#psv #spakenburg slechts maar goed dat #feyenoord niet tegen spakenburg hoefde te spelen

4.1.2 Woord en frase extractie

Bij OCTSM wordt in elk cluster bijgehouden welke frases er in de geclusterde tweets zijn voorgekomen. Hierbij is een frase een combinatie van twee of drie woorden die onderwerp-waardig zijn[4]. OCTSM gaat ervan uit dat als er naar de combinatie van woorden en frases wordt gekeken, dat de betekenis van het woord makkelijker kan worden bepaald. Als bijvoorbeeld het woord 'ster' en de frase 'oneindig heelal' in hetzelfde document worden gebruikt dan gaat het over een ster in de lucht en niet over een bekendheid. Het aantal woord-frase-combinaties wordt geteld en op basis daarvan wordt er een kans berekend dat een bepaald woord bij een frase past. Als vervolgens weer de frase "oneindig heelal", maar niet het woord 'ster' in een tweet voorkomt, dan is er al een bepaalde kans bepaald dat het woord 'ster' bij deze frase hoort. Dit is het extended Semantic smoothing model beschreven in paragraaf 3.1.2. In deze paragraaf zal er alleen naar de extractie van frases en de belangrijke woorden worden gekeken.

In het oorspronkelijke artikel [32] wordt in deze fase van het proces een tool genaamd Xtract [34] gebruikt. Een implementatie hiervan is ook te vinden op internet [16]. Dit is echter alleen te gebruiken als het om Engelse tekst gaat. Xtract is een statistische extractie tool met een aantal syntactische beperkingen. Deze tool maakt het mogelijk veel voorkomende frases bestaande uit zelfstandige naamwoorden uit een corpus, zonder enige kennis van buiten, te detecteren. Met deze tool is het ook mogelijk frases uit de tweets te extraheren. Ook onderwerp-waardige woorden[4] zouden hierdoor gedetecteerd kunnen worden. Met een onderwerp wordt hier niet het taalkundige onderwerp bedoeld, maar het algemene onderwerp (van een verhaal). Xtract voor de Nederlandse taal bestaat echter niet, maar er is wel een tool die dit proces gedeeltelijk zou kunnen overnemen. Dit is de tool genaamd 'Brill Tagger' [W17,W18]. Deze tool is veel simpeler dan Xtract. Deze tool geeft bij invoer van een zin alleen een classificatie voor elk woord in de opgegeven zin. De grovere classificaties die er mogelijk zijn, zijn te zien in tabel 3. Naast de tag gegeven in tabel 3 wordt er ook nog een specifiekere classificatie aan het woord gegeven. Dit zijn de labels die tussen haakjes staan. Brill tagger zelf maakt hier geen onderscheid in. Voor Brill tagger is bijvoorbeeld `V(hulp,..)` of `V(intrans,inf)` iets heel anders. Als voorbeeld wordt de classificatie van de in de vorige paragraaf besproken tweet hieronder gegeven. Speciaal voor deze stap worden de hashtags tijdelijk verwijderd. Een hashtag zegt over het algemeen veel over het onderwerp, maar het zou nog steeds kunnen dat dit niet het geval is. #leuk is bijvoorbeeld niet zo belangrijk als #wilders.

#psv/NN **#spakenburg**/NN **slechts**/Adv(gew,geen_func,stell,onverv)
maar/Adv(gew,geen_func,stell,onverv) **goed**/Adj(adv,stell,onverv)
dat/Conj(onder,met_fin) **#feyenoord**/NN
niet/Adv(gew,geen_func,stell,onverv) **tegen**/Prep(voor) **spakenburg**/NN
hoefde/V(hulp,ovt,1_of_2_of_3,ev) **te**/Prep(voor_inf)
spelen/V(intrans,inf)

Tag	Woordsoort	Voorbeelden
Pron	Voornaamwoord	Je, dat, ik
Adj	Bijvoeglijk naamwoord	Mooi, lelijk, vies
N	Zelfstandig naamwoord	Auto, fiets
NN	Niet bekend	Adfsadf, #pvv, start, gozer

[4] Woorden die een hoge kans hebben een onderwerp van een zin te zijn.

Num	Heeft met cijfers te maken	Veel, tweede, 3, 21ste
V	Werkwoord	Doen, gaan
Prep	Voorzetsel	In, op, tegen
Adv	Stopwoorden	Wel, alleen, toch, niet, daar
Conj	Voegwoord	En, maar, nu
Art	Lidwoord	De, het, een
Misc	Buitenlandse woorden[5]	The, to, and
Int	?	Ja, nee, hoor, he

Tabel 3. De verschillende classificaties van de Brill tagger.

De volgende stap is om met behulp van de Brill tagger zelf te bepalen wat de onderwerp-waardige woorden[4] en frases in een tweet zijn. Het is echter veel werk (als het al niet om een studie op zich gaat) om hier een goed algoritme voor te maken. Dit werk valt daarom ook buiten dit onderzoek. Er is echter wel een oplossing voor dit probleem gemaakt. Deze oplossing is niet optimaal en zou in een later stadium vervangen moeten worden. De Brill tagger is ook nog niet helemaal correct verwerkt in de applicatie. Om de Brill tagger aan te roepen moet er een bijgeleverd 'batch'-bestand worden gedraaid. Dit bestand moet aangeroepen worden bij elke tweet. Deze aanroep kost telkens veel tijd. Er worden namelijk bestanden aangemaakt en verwijderd, waardoor het proces wordt vertraagd. Als de Brill tagger in de applicatie zou zitten (communicatie via de API van de Brill tagger), kost dit haast geen tijd. De Brill tagger is echter niet het enige hulpmiddel dat gebruikt wordt om de onderwerp-waardige woorden[4] te detecteren. Ook worden er lijsten met niet relevante woorden bijgehouden.
Het vormen van frases is niet de enige reden dat onderwerp-waardige woorden[4] gemaakt moeten worden. Bij OCTSM wordt namelijk niet naar de 'relatieve' afstand tussen clusters gekeken. Als er een woord veel in de andere clusters zou voorkomen dan betekent dit niet dat dit woord een mindere waarde krijgt en wordt hierdoor niet beschouwd als een minder belangrijk woord. Het algoritme kijkt namelijk constant naar de gelijkenis van de clusters. Als een woord veel voor zou komen in de andere clusters, dan ziet het algoritme dit juist als een teken dat deze clusters misschien samengevoegd moeten worden (zie paragraaf 4.3). Om deze reden is het belangrijk dat alleen de relevante woorden worden geselecteerd. Dit gebeurt met de Brill tagger.

Brill tagger

De twee belangrijkste tags uit tabel 3 zijn 'NN' en 'N'. De 'NN'-tag is belangrijk, omdat een woord dat niet herkend wordt door de Brill tagger, meestal een woord is dat niet veel voorkomt. De erg algemene woorden zoals: 'man', 'hond', 'schande' of 'succes' worden wel herkend als zelfstandige naamwoorden, terwijl bij zelfstandige naamwoorden woorden zoals: 'wilders[6]' (of andere namen), 'beroepsdemagoog' of 'massaontslag' dit niet wordt gedaan. Hoe specifieker het onderwerp, hoe minder kans er is dat het woord geclassificeerd kan worden tot een woordsoort. Hoe specifieker een woord is des te belangrijker dit woord voor het detecteren van een nieuw onderwerp kan zijn. Een woord dat fout gespeld is, behoort echter ook tot deze tag. Het komt echter niet vaak voor dat een woord op dezelfde wijze fout

[5] De buitenlandse woorden die herkend worden zijn erg beperkt.
[6] Naam van een Nederlandse politicus.

geschreven wordt. Hierdoor zal een woord ook niet veel waarde krijgen in een cluster. Er geldt immers: hoe minder een woord voorkomt des te minder waarde er aan het woord wordt toegekend. Zelfstandige naamwoorden ('N'-tag) geven over het algemeen wat algemenere onderwerpen aan. Ook deze woorden kunnen kenmerkend voor een tweet zijn. Ook al is dit in mindere mate. Een tweet bevat namelijk al weinig informatie. Als deze categorie ook nog weggelaten zou worden dan blijft er al gauw niet veel meer over.
Naast de 'NN'- en 'N'-tag kunnen werkwoorden en bijvoeglijke naamwoorden ook wel iets over een onderwerp zeggen. De woorden die voortkomen uit de 'NN'- en 'N'-tag zijn echter al ruim voldoende. Het toevoegen van andere woorden zou alleen maar leiden tot een grotere kans op een match tussen clusters en/of tweets, die niet veel met elkaar te maken hebben. Hoe meer woorden er namelijk overeenkomen, des te groter de kans is dat clusters en/of tweets worden samengevoegd. In het ergste geval zijn er bijvoorbeeld twee clusters, waar zich alleen maar de 'V'- of 'Adj'-tags in bevinden. Hierdoor vindt er alleen maar een match plaats op basis van algemene woorden. Dit zal de accuratesse van het clusteren niet bevorderen. Het is echter zonde om de woorden met een 'V'- of 'Adj'-tag weg te gooien. Een bijvoeglijk naamwoord geeft namelijk juist de sentimenten van een onderwerp aan en wat er met onderwerpen wordt gedaan, wordt vaak duidelijker als werkwoorden ook vernoemd worden. Daarom zullen deze woorden wel meegenomen worden om in het cluster te belanden, maar op deze woorden zal niet gematcht worden. Met andere woorden: als er twee clusters of een tweet en een cluster met elkaar vergeleken worden dan zal er alleen naar de woorden met een 'NN'- of een 'N'-tag gekeken worden. Ook zullen er alleen frases worden gevormd met woorden met een 'NN'- of een 'N'-tag. In het uiteindelijke cluster zullen de werkwoorden en de bijvoeglijke naamwoorden wel te zien zijn. Een uitzondering hierop zijn hulpwerkwoorden (zie het label bij het werkwoord 'hoefde' in het voorbeeld van een tweet classificatie met Brill tagger).

Stopwoorden
Naast de woordselectie met behulp van de classificatie van de Brill tagger vindt er nog een selectieronde plaats. Parabots is in het bezit van een lijst met woorden die erg algemeen zijn en die niet specifiek op een onderwerp betrekking hebben. De meeste van deze woorden worden al gefilterd door de Brill tagger, maar aangezien de lijst toch beschikbaar is, worden woorden die zich op deze lijst bevinden, ook verwijderd.

De stam bepalen
De laatste stap die gedaan moet worden is alleen de stam van het woord bepalen. Een woord kan namelijk op verschillenende manieren geschreven worden. Als het bijvoorbeeld over 'kasten' gaat en als het daarna over een 'kast' gaat dan wordt dit niet als hetzelfde woord beschouwd hoewel dit het wel het geval is. Als alleen naar de stam van woorden wordt gekeken dan is dit probleem er niet meer. 'kast' blijft gewoon 'kast' en 'kasten' wordt ook 'kast' en is op deze manier wel als hetzelfde woord geïdentificeerd.

De woorden en frases die uiteindelijk van de eerder genoemde tweet over zijn gebleven zijn de volgende:

Woorden: #psv, #spakenburg, goed, #feyenoord, spakenburg en spelen.

Frases: “#psv #spakenburg”, “#feyenoord #psv”, “#psv spakenburg”, “#feyenoord #spakenburg”, “spakenburg #spakenburg” en “spakenburg #feyenoord”.

4.2 Het offline proces

De woorden en frases in een tweet die een onderwerp zouden kunnen vormen zijn nu bekend. Voordat er daadwerkelijk een stroom van tweets kan worden verwerkt, vindt er eerst het offline proces plaats. Hierbij wordt het eerste gedeelte van de stroom onderschept om informatie te verzamelen over de inhoud van de stroom. De informatie die uit deze stroom wordt gehaald bestaat uit twee modellen: het ‘language model’ en het ‘topic signature translation model’. Het ‘language model’ kijkt naar de frequentie woorden en het ‘topic signature (frase) translation’-model kijkt naar de combinaties van frases en woorden. Naarmate de tijd vordert zal dit offline proces vernieuwd moeten worden. Er komen namelijk constant nieuwe onderwerpen bij. Onderwerpen die de ene week erg veel besproken worden zouden de volgende week helemaal niet meer aan bod kunnen komen, maar vervangen zijn door weer totaal nieuwe onderwerpen. Het ‘topic signature ‘translation model’ zal als eerst worden besproken en dit zal gevolgd worden door een uitleg over de representatie over een cluster (het clusterprofiel).

4.2.1 Het ‘topic signature translation’-model

Als een frase t_k (topic signature) is gegeven dan is er een kans $p(w|t_k)$ dat woord w bij deze frase hoort. Deze kans wordt bepaald door het ‘topic signature translation model’.

Waarom zou er een ‘topic signature translation’-model nodig zijn?

Stel dat in het offline proces er een woord heel vaak in combinatie met een aantal frases is voorgekomen. Als deze frases zich later dan ook in een cluster bevinden dan kan er met grotere zekerheid gezegd worden dat dit woord meer kans heeft om echt bij dit onderwerp te horen.

De implementatie van het ‘topic signature translation’-model

Om kans $p(w|t_k)$ te bepalen moeten eerst alle woord-frase-combinaties geteld worden die zich in de offline dataset bevinden. De documenten waar een frase t_k in voorkomt wordt genoteerd als D_k. Niet al de woorden die in deze set van documenten voorkomen hebben te maken met frase t_k. Er wordt hierbij vanuit gegaan dat D_k gegenereerd is door een ‘mixture language model’. De woorden in D_k hebben of met $p(w|t_k)$ te maken of hebben een verband met het ‘background collection model’ $p(w|C)$. Om het ‘topic signature translation model’ te bepalen moeten de volgende formules met het iteratieve expectation maximisation-proces worden berekend:

$$p(w|D_k) = (1-\beta)p(w|t_k) + \beta p(w|C) \quad (26)$$

$$p^n(w) = \frac{(1-\beta)p^n(w|t_k)}{(1-\beta)p^n(w|t_k) \; + \; \beta p(w|C)} \quad (27)$$

$$p^{(n+1)}(w|t_k) = \frac{c(w, D_k)p^n(w)}{\sum_i^N c(w_i, D_k)p^n(w_i)} \quad (28)$$

$$p(w|C) = \frac{c(w,C)}{\sum_{w_i \in C} c(w,C)} \qquad (29)$$

De variabelen die hierbij gebruikt zijn hebben de volgende betekenis:

- C is het 'collection model' dat staat voor alle woorden in het corpus[7]. Het corpus bestaat hierbij uit alle woorden die in het offline proces zijn gevonden.
- $c(w, C)$ is het aantal keer dat woord w voorkomt in het 'background collection model' C.
- $c(w, D_k)$ is het aantal keer dat woord w voorkomt in de set van document D_k.
- β is een coëfficiënt dat bepaald hoe zwaar het 'background collection model' meegerekend moet worden.
- n is het aantal iteraties
- N is het totaal aantal woorden dat voorkomt in D_k.

Hoe aan deze formules gekomen is, is te lezen in [33,37]. $p(w|t_k)$ is de kans die voor elke woord-frase combinatie berekend moet worden. Hierbij kan de beginwaarde van $p(w|t_k)$ een random kans zijn. Deze random kans moet vervolgens in formule (27) worden gestopt. De uitkomst hiervan wordt hierna gebruikt voor formule (28), waardoor er uiteindelijk een nieuwe waarde voor $p(w|t_k)$ ontstaat. Dit iteratieve proces vindt net zo lang plaats totdat de verandering $p(w|t_k)$ niet groter is dan een erg kleine waarde (afhankelijk van de precisie die er nodig is).

4.2.2 Het clusterprofiel

Het 'language model' wordt bepaald door $p(w|C)$ voor elk woord uit te rekenen. Hiermee zijn de berekeningen in het offline proces afgerond en moeten alleen de initiële clusters nog gemaakt worden. OCTSM is een algoritme dat gebaseerd is op k-means en net als bij k-means moet hier het aantal clusters vooraf bepaald worden. Elk cluster bestaat hierbij uit een clusterprofiel met de elementen *DF2*, *DF1*, *s* en *l*. *DF2* en *DF1* zijn allebei een vector met *wb* elementen. Hierbij is *wb* het aantal verschillende woorden dat zich in het cluster bevinden. Voor elk element *i* (met $1 \leq i \leq wb$) zien de vectoren er als volgt uit:

$$DF2_i = \sum_{d \in c} f(t - T_d) c(w_i, d) \qquad (30)$$

$$DF1_i = \sum_{k}^{N} p(w_i|t_k) \sum_{d \in c} f(t - T_d) c(t_k, d) \qquad (31)$$

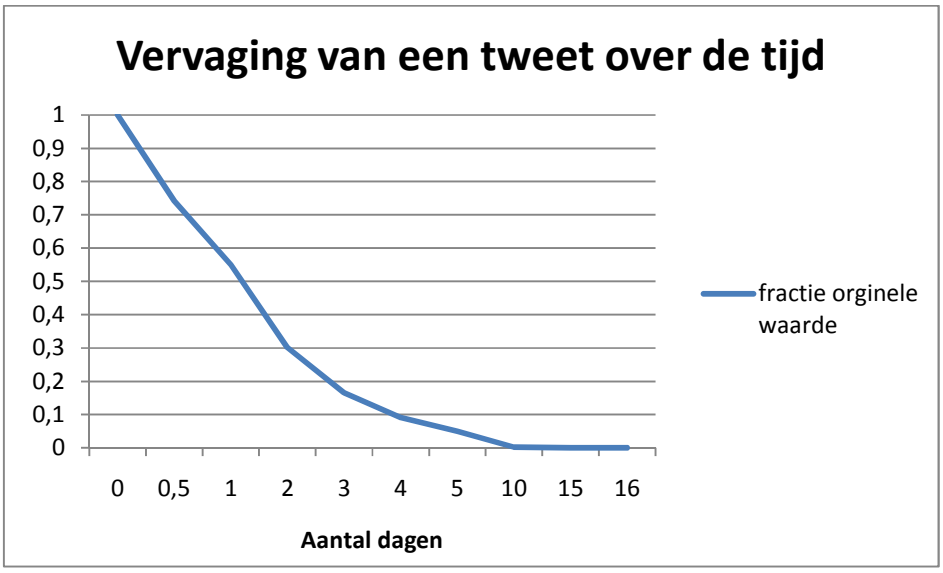
Hierbij staat d voor een document, c voor een cluster en N voor het totale aantal frases die in combinatie met w_i in cluster c zijn gevonden. $f(t - T_d)$ is hierbij de 'fading function' waar t staat voor het huidige tijdstip en T_d voor het tijdstip van toevoeging van document d. Deze functie zorgt ervoor dat oudere documenten minder zwaar meetellen dan de documenten die net zijn toegevoegd en ziet er als volgt uit.

$$f(t) = 2^{-\frac{1}{\zeta}t} \qquad (32)$$

Hierbij is ζ de factor voor de snelheid van de vervaging van een document. Als een tweet net wordt toegevoegd dan wordt elk woord en elke frase voor waarde 1 meegerekend, maar deze waarde wordt minder naarmate de tijd verstrekt. Het doel is immers om te detecteren waar er op dit moment veel over wordt getweet. Daarbij moeten daarom de tweets die nu binnenkomen meer waarde krijgen dan

[7] Het corpus bestaat uit alle woorden die gevonden zijn in het offline proces.

degene die een uur geleden waren binnengekomen. Als ζ bijvoorbeeld 0,000007 zou zijn dan zouden de woorden van een tweet over een dag nog maar de helft waard zijn. Over tien dagen zou deze tweet helemaal niet meer worden meegerekend. Dit is ook in figuur 3 te zien.



Figuur 3. De waarde van een tweet afgezet tegen de tijd met ζ = 0,000007.

De clusterelementen *s* en *l* worden als volgt gedefinieerd:

$$s = \sum_{d \in c} f(t - T_d)c(t_k, d) \qquad (33)$$

l = het tijdstip waarop de laatste tweet is toegevoegd.

Met deze definities kan uiteindelijk het extended Semantic smoothing model van paragraaf 3.1.2 worden toegepast. Vergelijking (1) ziet er nu als volgt uit (de afleiding van deze formule is te vinden in [32]):

$$p_{bt}(w|c_j) = (1 - \lambda)\frac{\mathrm{DF2_i}}{\sum_i \mathrm{DF2_i}} + \lambda\frac{\mathrm{DF1_i}}{\mathrm{s}} \qquad (34)$$

Hiermee is de kans $p_{bt}(w|c_j)$ uitgerekend dat een woord w bij een gegeven cluster c_j hoort. Translatiecoëfficiënt λ moet hierbij als parameter aan het algoritme worden meegegeven. Hoe hoger deze coëfficiënt, des te meer er rekening wordt gehouden met het 'topic signature translation model'.

Door dit clusterprofiel op deze manier te definiëren, kunnen er ook gemakkelijk tweets worden toegevoegd, clusters worden samengevoegd en de 'fading function' worden toegepast. Dit blijkt uit de volgende eigenschappen:

Clusters samenvoegen (eigenschap 1)

Stel er zijn twee clusters

$$D(t, c_1) = (DF2(c_1), DF1(c_1), s_{c_1}, l_{c_1})$$
$$D(t, c_2) = (DF2(c_2), DF1(c_2), s_{c_2}, l_{c_2})$$

Dan is de samengevoegde cluster:

$$D(t, c_1 \cap c_2) = (DF2(c_1) + DF2(c_2), DF1(c_1) + DF1(c_2), s_{c_1} + s_{c_2}, \max(l_{c_1}, l_{c_2}))$$

Tweet toevoegen aan een cluster (eigenschap 2)

Stel er is een cluster

$$D(t, c_1) = (DF2(c_1), DF1(c_1), s_{c_1}, l_{c_1})$$

Dan ziet het cluster er na toevoeging van een tweet er als volgt uit:

$$D(t, c_1) = (DF2(c_1) + c(w_i, d), DF1(c_1) + \sum_k^N p(w_i|t_k)c(t_k, d), s_{c_1} + \sum_k^N c(t_k, d), l_{c_1} = t)$$

De 'fading function' toepassen op een cluster (eigenschap 3)

Stel dat op tijdstip t_1 er het volgende cluster is:

$$D(t_1, c_1) = (DF2(c_1), DF1(c_1), s_{c_1}, l_{c_1})$$

Dan ziet het cluster er na het verstrijken van een tijdseenheid op tijdstip t_2 als volgt uit:

$$D(t_2, c_1) = (DF2(c_1) * 2^{-\frac{1}{\zeta}(t_2 - t_1)}, DF1(c_1) * 2^{-\frac{1}{\zeta}(t_2 - t_1)}, s_{c_1} * 2^{-\frac{1}{\zeta}(t_2 - t_1)}, l_{c_1} = t_2)$$

De volgende stap is deze clusterprofielen aan te maken voor elk cluster. In deze initiële stap wordt er een tweet bij elk cluster toegevoegd. In het online proces worden hier de rest van de tweets uit de datastroom aan toegevoegd. Dit wordt beschreven in de volgende paragraaf.

4.3 Het online proces

Het offline proces is afgerond en nu moet een contante stroom van tweets verwerkt worden. Dit is waar het online proces voor dient. Figuur 4 geeft een overzicht van alle stappen in het OCTSM-algoritme. Stappen 1 en 2 zijn in de vorige paragraaf al aan bod gekomen. Ook zijn de drie eigenschappen van het clusterprofiel besproken. Deze zullen worden toegepast op het online proces. Het online proces begint bij stap 3. Alle stappen in deze procedure zullen eerst besproken worden en daarna zal apart stil gestaan worden bij het samenvoegen van de clusters en het toevoegen van een tweet aan een cluster.

4.3.1 De procedure die een binnenkomende tweet ondergaat

Per stap zal worden beschreven wat er in figuur 4 gebeurd.

Stappen 5 en 6

Op tijdstip t komt een nieuwe tweet binnen (stap 5). Eigenschap 3 gebruikt dit tijdstip vervolgens om elk cluster up to date te houden (stap 6). Dit komt neer op een devaluatie van de woorden die tot nu toe al waren toegevoegd aan de clusters. De grootte van deze devaluatie is afhankelijk van de laatste update die heeft plaatsgevonden. Hoe langer deze update geleden is, des te minder waard de bestaande woorden worden.

Stap 7-9

Vervolgens wordt op elk woord in elke cluster, formule (34) toegepast (stap 7-9). Als dit de eerste tweet is dan bestaan deze waardes nog niet en als dit niet de eerste tweet is dan zijn deze waardes veranderd door de toevoeging van een tweet in de vorige run. Dit betekent dat niet alleen de elementen in het clusterprofiel worden bijgehouden, maar dat er ook per cluster een aparte lijst wordt bijgehouden dat de kans $p_{bt}(w|c_j)$ voor elk woord beschrijft.

Algorithm. OCTSM
Inputs: A stream of text data, D, k is the number of clusters, *MinFactor* and *MergeFactor* are the thresholds.
Output: A set of k cluster profiles $(D(t, c_1), \ldots, D(t, c_k))$
Method:

1. Extract words and multiword phrases, and build the translation model for the training data set D.
2. Read in the first k documents and generate k cluster profiles $(D(t, c_1), \ldots, D(t, c_k))$;
3. **While** (the stream is not empty) **do**
4. **Begin**
5. t = GetCurrentTimestamp ();
6. Update all cluster profiles with t;
7. **For** each cluster j **do**
8. **For** each word w_i **do**
9. The cluster model with semantic smoothing $p'_{bt}(w_i|c_j)$ is estimated;
10. **If** Merge(*MergeFactor*) then goto Step 22.
11. Read in the next document d and extract words and multiword phrases from d;
12. **For** each cluster j **do**
13. The similarity $p(d|c_j)$ between d and c_j is estimated;
14. $AssignID = \text{argmax}_j p(d|c_j)$;
15. **If** $(p(d|c_{assignID}) < MinFactor)$ **then** {
16. $NID = \text{argmin}_j D(t, c_j).l$;
17. Delete the most inactive cluster c_{NID};
18. Create a new empty cluster with $ID = NID$ and build its cluster profile;
19. Assign document d to the new cluster profile; }
20. **Else**
21. Assign document d to cluster profile with $ID = AssignID$;
22. **End**

Figuur 4. Overzicht van het OCTSM-algoritme (bron: [32]).

Stap 10

Hierna wordt gekeken of er clusters zijn die zo erg op elkaar lijken dat deze samengevoegd kunnen worden (stap 10). Als de gelijkenis van twee clusters zo groot is dat er een samenvoeging plaatsvindt, dan worden ze met de samenvoegingeigenschap bij elkaar gevoegd. Hierdoor ontstaat er een clusterplaats die niet bezet is. Aan deze cluster wordt vervolgens de nieuwe tweet toegevoegd. Op deze manier krijgt deze tweet genoeg tijd om zich weer aan een ander cluster toe te voegen als het daar op lijkt of juist een nieuwe trend te vormen. Mocht het zo zijn dat er geen van de clusters overeenkomt dan gebeurt er in deze stap niets met de nieuwe tweet. Wanneer clusters wel of niet samengevoegd worden, wordt apart besproken in paragraaf 4.3.2.

Stap 11-22

De volgende stap is om te kijken of het document zelf genoeg overeenkomsten met een van de clusters bevat, zodat deze direct bij een cluster gevoegd kan worden. Als dit wel het geval is dan wordt deze tweet bij het cluster gevoegd. Mocht dit niet het geval zijn, dan wordt er gekeken welk cluster het langst geleden een tweet toegewezen heeft gekregen. Er wordt hierbij vanuit gegaan dat dit meest inactieve cluster geen trend bevat dat op dit moment relevant is en wordt daarom verwijderd om plaats te maken voor een tweet die wellicht wel een nieuwe trend bevat. Hoe bepaald wordt of een document overeenkomt met een van de clusters wordt beschreven in paragraaf 4.3.3.

4.3.2 Samenvoegen van clusters

De te volgen procedure bij het bepalen of er tussen twee clusters een samenvoeging moet plaatsvinden, is te zien in figuur 5. Zoals in deze figuur te zien is, wordt elk woord uit het corpus V met elkaar vergeleken. Als een woord echter niet bestaat in een cluster dan is de kans dat dit woord in ditzelfde

cluster voorkomt erg klein. Dit is echter niet nul, omdat elk woord een kans moet krijgen om in een cluster te belanden. Deze kans wordt onder verschillende omstandigheden verschillend bepaald:

- Een woord komt wel in het cluster voor: dit is de eerder in formule (34) berekende kans $p_{bt}(w|c_j)$.
- Een woord komt niet in het cluster voor, maar wel in het corpus:

$$p(w|c_j) = \frac{w_N + 1}{V_N + V_c} \qquad (35)$$

- Een woord komt niet in het cluster voor en niet in het corpus:

$$p(w|c_j) = \frac{1}{V_N + V_c} \qquad (36)$$

Hierbij is w_N de frequentie van het woord w in het corpus, V_N het totale aantal woorden in het corpus en V_c het aantal verschillende woorden in het corpus.

In formule (35) wordt 'add one smoothing' (Laplace's law of succession [39]) toegepast: elk verschillend woord krijgt een frequentie 1 erbij. Hiermee wordt er rekening gehouden met woorden die helemaal niet in de trainingsset (corpus) voorkomen. Dat een woord niet in de trainingsset voorkomt, betekent niet dat het woord helemaal niet kan voorkomen. Als een woord niet in de trainingsset voorkomt dan wordt formule (36) gebruikt.

Algorithm. Merge (*MergeFactor*)
Method:
1. Calculate the similarity between cluster i and cluster j $(1 \leqslant i \neq j \leqslant k)$ using the following formula $(\Delta(c_i, c_j) \equiv \sum_{w \in V^*} p(w|c_i) \log [p(w|c_i)/p(w|c_j)]$ and $\Delta(c_j, c_i) \equiv \sum_{w \in V^*} p(w|c_j) \log [p(w|c_j)/p(w|c_i)]$;
2. **If** $(1/\min(\Delta(c_i, c_j), \Delta(c_j, c_i)) > MergeFactor)$ {
3. Clusters i and j are merged as new cluster i using Property 1;
4. A new empty cluster with ID $= j$ is created using Definition 5 and d is assigned to cluster j using Property 2;
5. Update cluster i and j with Property 3;
6. **Return** true;}
7. **Return** false;

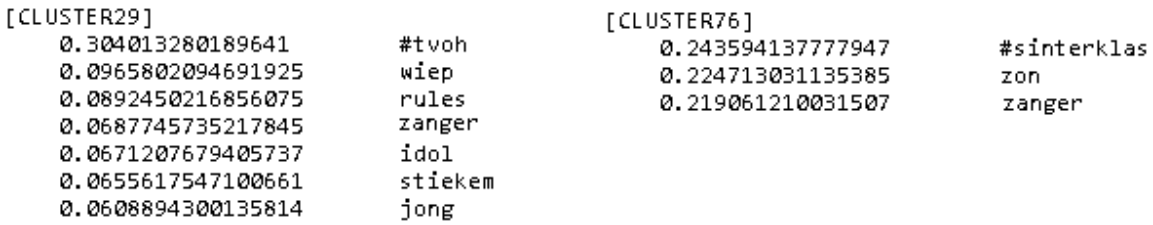
Figuur 5. Een overzicht van het algoritme dat de clusters met elkaar vergelijkt. 'Definition 5' refereert hierbij naar het clusterprofiel.

De volgende stap is 1 te delen door het minimum van cluster c_i en cluster c_j. Als deze waarde groter is dan een waarde *Mergefactor* dan wordt clusters c_i en c_j samengevoegd. *Mergefactor* is hierbij een parameter die van te voren moet worden opgegeven.

Deze vergelijkingsmethode werkte echter niet optimaal. Artikel [32] geeft aan dat er ongeveer 20.000 documenten[8] in 225 seconden verwerkt kunnen worden. Bij de implementatie die in deze scriptie gebruikt wordt, worden er slechts 500 tweets in ongeveer 1812 seconden geclusterd. Naast dat het algoritme erg langzaam gaat, zijn de prestaties van het algoritme ook niet goed. Hoe de prestaties zo laag blijven wordt geïllustreerd in het volgende voorbeeld.

Voorbeeld:

Er zijn twee clusters te zien in figuur 6.

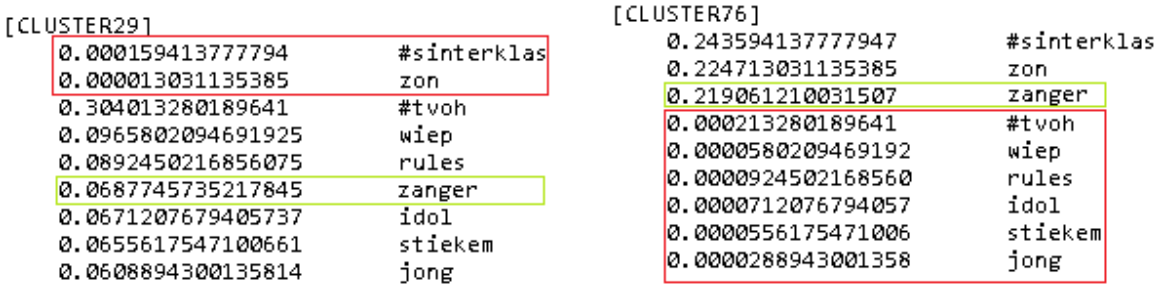


[CLUSTER29]	
0.304013280189641	#tvoh
0.0965802094691925	wiep
0.0892450216856075	rules
0.0687745735217845	zanger
0.0671207679405737	idol
0.0655617547100661	stiekem
0.0608894300135814	jong

[CLUSTER76]	
0.243594137777947	#sinterklas
0.224713031135385	zon
0.219061210031507	zanger

Figuur 6. Voorbeeld van twee clusters, waarvan de woorden gestemd zijn.

Als de formule van figuur 5 op de clusters in figuur 6 wordt toegepast, dan zegt het algoritme dat deze twee samengevoegd moeten worden (mits de *mergefactor* niet proportioneel hoog[9] is), omdat hij een match heeft gevonden met het woord 'zanger'. De waardes die bij deze vergelijking gebruikt worden zijn in figuur 7 te zien.



[CLUSTER29]	
0.000159413777794	#sinterklas
0.000013031135385	zon
0.304013280189641	#tvoh
0.0965802094691925	wiep
0.0892450216856075	rules
0.0687745735217845	zanger
0.0671207679405737	idol
0.0655617547100661	stiekem
0.0608894300135814	jong

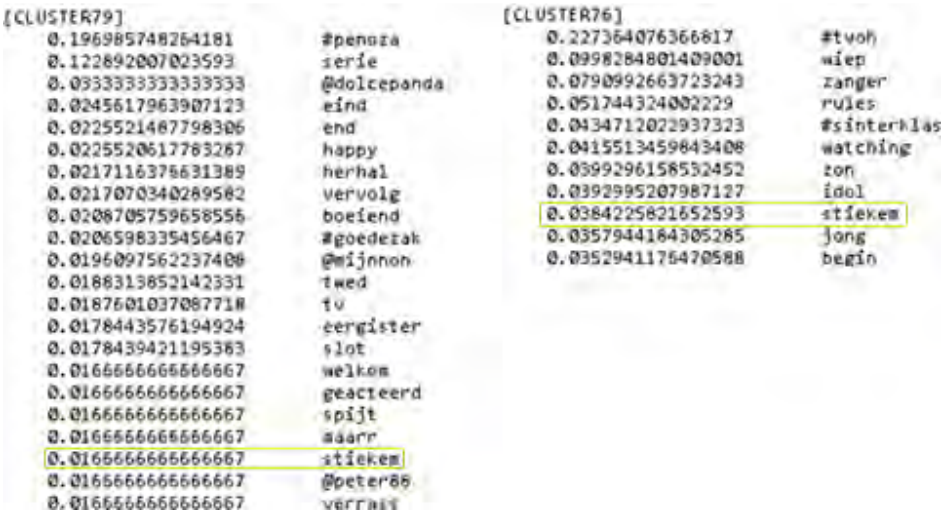
[CLUSTER76]	
0.243594137777947	#sinterklas
0.224713031135385	zon
0.219061210031507	zanger
0.000213280189641	#tvoh
0.0000580209469192	wiep
0.0000924502168560	rules
0.0000712076794057	idol
0.0000556175471006	stiekem
0.0000288943001358	jong

Figuur 7. Voorbeeld van twee clusters, waarvan het algoritme aangeeft dat ze samengevoegd moeten worden. Het rode kader geeft hierbij de woorden aan die zijn toegevoegd door formule (35 of 36) en de woorden in het groene kader zijn de woorden die in de twee clusters overeenkomen.

Een tweet bevat niet veel woorden. Het zou zelfs kunnen dat een tweet maar drie of zelfs maar een belangrijk woord bevat. Doordat een tweet maar een woord kan bevatten, is het minimum aantal woorden in een cluster ook maar een. Bij een cluster met weinig woorden, zijn de woorden die het cluster wel bevat erg belangrijk. Deze woorden krijgen daarom een erg belangrijke rol in het cluster. Net zoals bij het voorbeeld, is de kans dat een cluster dan al erg snel samengevoegd wordt met een ander cluster, erg groot. Het ligt echter niet alleen aan te weinig informatie in een cluster. Dit is te zien in figuur 8. Cluster 79 bevat tweeëntwintig verschillende woorden en cluster 76 bevat er elf. Ook hier geeft het algoritme aan dat deze twee clusters samengevoegd moeten worden. Dit terwijl het woord dat wel overeenkomt ook nog eens een van de minst belangrijke woorden uit beide clusters is.

[8] De documenten waarnaar hier gerefereerd wordt, zijn artikelen. Het is erg waarschijnlijk dat deze documenten veel groter zijn dan tweets.

[9] In [32] wordt een waarde tussen 0 en de 0.8 gebruikt. De waarde die hier bedoeld wordt ligt veel hoger.



Figuur 8. Een voorbeeld van twee clusters, waarvan het algoritme zegt dat er een samenvoeging moet plaatsvinden. Dit gebeurt op basis van de match die gevonden is van het woord omkaderd met een groene lijn.

Dit waren slechts voorbeelden, maar dit symptoom bleef constant voorkomen. Daarom is deze procedure iets aangepast. In figuur 5 is ook te zien dat naar V^* wordt verwezen en niet naar V. Er wordt namelijk niet meer naar het hele corpus gekeken, maar alleen naar de belangrijkste woorden in het cluster. Dit zijn woorden die het grootste gewicht in een cluster hebben. Hoeveel woorden van het grootste gewicht nodig zijn, wordt aangegeven met de parameter *top %*. Deze parameter geeft aan naar welk percentage van de belangrijkste woorden gekeken moet worden. Als deze parameter bijvoorbeeld op 30% gezet zou worden dan zou bij cluster 76 naar de bovenste 7 woorden moeten worden gekeken. Hierbij wordt 30% genomen van 22 en vervolgens naar boven afgerond. Dit naar boven afronden zorgt er ook voor dat er minimaal naar een woord wordt gekeken. Dit resulteert voor cluster 76 tot de bovenste 4 woorden. Deze parameter zorgt er voor dat er alleen een samenvoeging tussen de twee clusters kan plaatsvinden als er een match is tussen de belangrijkste woorden van de betreffende clusters.

Als er een foute samenvoeging van twee clusters ontstaat, dan kan dit leiden tot een opeenvolging van incorrecte samenvoegingen. Er komen namelijk verschillende onderwerpen door elkaar heen te lopen, waardoor er een hoop verschillende woorden in het cluster ontstaan waarmee gematcht kan worden. Zonder de parameter *top %* komt dit erg vaak voor, maar met deze parameter is dit ook niet helemaal te vermijden. Een voorbeeld van een cluster die erg snel random clusters bij zich heeft gevoegd, is te zien in figuur 9. In deze figuur is een cluster te zien die bestaat uit 1496 verschillende woorden. Als hier dan naar de bovenste 30% wordt gekeken, dan komt dit nog steeds neer op 449 woorden waarop gematcht kan worden. Hierdoor ontstaat weer dezelfde situatie als voorheen. Ook hier zullen weer op random woorden worden gematcht. Daarom is er gekozen om maximaal naar de bovenste 10 woorden te kijken. Er is er hierbij vanuit gegaan dat een trend door 10 woorden gedefinieerd kan worden.

De reden dat de originele procedure in figuur 5 erg langzaam is, is omdat er al gauw een cluster ontstaat zoals in figuur 9. Als dit cluster vergeleken moet worden met elk ander cluster dan moeten er telkens 1496 woorden met elkaar vergeleken worden. Als er in totaal dan ook nog 100 verschillende clusters zijn om vergeleken mee te worden, dan loopt de rekentijd al gauw op. Het zou wel kunnen dat [32] speciale technieken (zoals [W19]) gebruikt heeft, die efficiënt matrixberekeningen kunnen uitvoeren. Het zou kunnen dat het vergelijken van verschillende clusters hierdoor sneller gaat. Voordat er werd ingezien dat

er speciale technieken nodig waren om clustervergelijkingen te doen, was het te laat om deze ook, in de bij deze scriptie ontwikkelde software, te gebruiken. Of het gebruik van [W19] ook daadwerkelijk geholpen had is niet duidelijk. Door de parameter top % te gebruiken is het aantal vergelijkingen dat gedaan moet worden toch beperkt. Er hoeven namelijk maar maximaal 10 woorden per cluster vergeleken te worden. In de originele procedure liep de rekentijd op met het aantal woorden dat in zich in het cluster begaf. Met de parameter top % blijft dit grotendeels gelijk.

```
[CLUSTER66]
    0.0103496868559995        #pvv
    0.0103194632818106        #blijdorp
    0.0100490801608656        #sinterklas
    0.00869514434408257           #ns
    0.00859194356923191           #ipad
    0.00832604365915249           #ohohcherso
    0.00811022106638489           #tnt
    0.00777663089811781           #f1
    0.00762351608157          #museumnacht
    0.00761271051292791           #obama
    0.00680802422924421           #nkafstand
    0.00675213675853821           #feyenoord
    0.00650294687499944           #penoza
    0.00624716864768816           #gtst
    0.00600123418385536           #mulisch
    0.00570657313758014           aub
    0.00560722799791844           blijdorp
    0.00555390092751747           petitie
    0.00554155230641954           tek
    0.00543278082030632           pot
    0.00515261210969037           #zetjepootvoorblijdorp
    0.00512683573327603           #fail
    0.00424199175505329           #vvd
    0.00389501377598104           matsoe
    0.00358894824783782           really
    0.00346390220461059           #durftevrag
    0.00333730052328792           #cda
    0.00326187402554029           just
    0.00314413482840868           top
    0.00275447973385549           stop
    0.00272887597275479           ret
    0.00258620689655172           bericht
    0.00253899200953533           sterretj
    0.00251338800156838           keuring
    0.00251328850215498           vinger
Dit cluster bevat nog 1461 andere woorden die niet worden getoond.
```

Figuur 9. Een voorbeeld van een cluster na tal van samenvoegingen van clusters met irrelevante onderwerpen.

4.3.3 Het toevoegen van een tweet aan een cluster

Met behulp van de kans berekend in formule (34), kan ook de kans uitgerekend worden dat een document (tweet) d bij een cluster c_j hoort. De formule die in [32] is gegeven luidt als volgt:

$$\log p(d|c_j) = \sum_{w \in V} c(w,d) \log p(w|c_j) \qquad (37)$$

Hierbij is $c(w,d)$ de frequentie van woord w in document d. In het artikel is verwezen naar [33]. Daar wordt gezegd dat op basis van een studie [38] er voor het multinomial model is gekozen. [38] laat zien dat multinomial model consequent beter presteert dan het multivariate Bernoulli model. Als echter de

bovenstaande formule wordt toegepast dan zou dit betekenen dat hoe meer woorden een tweet bevat hoe kleiner de kans wordt dat dit document bij een cluster hoort. Het volgende voorbeeld illustreert dit.

Voorbeeld

Gegeven zijn de volgende gegevens:

$p(w|c_j)$ = De kans dat woord w bij cluster c_j hoort (deze waarde is voor alle woorden van document *d* bekend).

$$\begin{aligned} \mathrm{p}('\mathrm{a}'|\mathrm{c_j}) &= 0{,}2 \\ \mathrm{p}('\mathrm{b}'|\mathrm{c_j}) &= 0{,}3 \\ \mathrm{p}('\mathrm{c}'|\mathrm{c_j}) &= 0{,}5 \end{aligned}$$

Als er vervolgens de kansen $\mathrm{p}(\mathrm{w}|\mathrm{c_j})$ van alle woorden *w* in het document worden vermenigvuldigd[10] dan resulteert dit in de kans dat document *d* bij cluster c_j hoort. Dit wil zeggen dat een document met veel woorden automatisch minder kans heeft om bij c_j te horen dan een document met weinig woorden.
Als vervolgens namelijk de kansen worden berekend van de documenten 'b c' en 'a', dan zien de kansen er als volgt uit:

$$\begin{aligned} \mathrm{p}('\mathrm{a}'|\mathrm{c_j}) &= 0.2 \\ \mathrm{p}('\mathrm{b\ c}'|\mathrm{c_j}) &= 0.3 * 0.5 = 0.15 \end{aligned}$$

Dit betekent dat document 'a' meer kans heeft om bij cluster c_j te horen dan document 'b c'. Terwijl beide woorden 'b' en 'c' individueel al meer kans hebben om bij het cluster te horen dan 'a'. Dit zou eigenlijk juist een grotere kans moeten krijgen. Daarom is er voor gekozen om het gemiddelde te nemen van de woorden:

$$\log p(d|c_j) = \frac{\sum_{w \in V} c(w,d) \log p(w|c_j)}{N} \qquad (38)$$

Als de formule vervolgens wordt toegepast dan krijgt document 'b c' wel een grotere kans:

$$\begin{aligned} \log \mathrm{p}('\mathrm{a}'|\mathrm{c_j}) &= \frac{\log(0{,}2)}{1} = -0.70 \\ \log \mathrm{p}('\mathrm{b\ c}'|\mathrm{c_j}) &= \frac{\log(0{,}3) + \log(0{,}5)}{2} = -0{,}41 \\ \mathrm{p}('\mathrm{a}'|\mathrm{c_j}) &= 10^{-0{,}7} = 0{,}20 \\ \mathrm{p}('\mathrm{b\ c}'|\mathrm{c_j}) &= 10^{-0{,}41} = 0{,}39 \end{aligned}$$

Bij het vergelijken van documenten met een cluster is er dus afgeweken van de formule beschreven in [32].

4.4 Bevindingen

In deze paragraaf zullen nadelige gevolgen die zijn gevonden tijdens de implementatie van het algoritme beschreven worden en worden toegelicht met een eventuele oplossing.

In paragraaf 4.1 is zelf een procedure ontwikkeld die de relevantere woorden uit de tweets filtert. Dit was nodig omdat het algoritme dat hier in [32] voor gebruikt is, Xtract [34], niet voor de Nederlandse taal beschikbaar is. Deze procedure is een onderzoek op zich waard. Nu is de Brill tagger gebruikt om dit

[10] Dit is hetzelfde als de som van de log-waardes.

op te lossen, dit is niet noodzakelijkerwijs de beste oplossing. Er zijn namelijk een hoop woorden die nog niet herkend worden. Ook zou er nog meer gelet kunnen worden op woorden met spelfouten erin. Nu wordt er alleen naar de stam gekeken naar het woord, waardoor hier al enigszins rekening mee wordt gehouden. Google gaat echter nog iets verder. Zij hebben een applicatie [W20] ontwikkeld die spelfouten uit woorden kan halen. Hoe goed dit product werkt is nog niet duidelijk maar zou misschien in deze situatie een aanvulling zijn.
Bij het trainen van het 'topic signature translation model' aan de hand van 'expectation maximisation' op 20.000 tweets viel het op dat er erg veel werkgeheugen nodig was. Bij ongeveer 30.000 tweets was er niet genoeg geheugen over om al deze tweets te trainen. Dit is afhankelijk van de computer die er gebruikt wordt en van de wijze van implementatie, maar als er meer tweets moeten worden getraind bij de huidige implementatie dan zal dit in stapjes moeten gebeuren.
Als er in een later stadium dit algoritme op een echte stroom van tweets moet worden toegepast, dan is het 'topic signature translation model' op een gegeven moment niet meer up to date. Bij Twitter worden er zowat elke dag weer nieuwe hashtags verzonnen. Van de tweets die deze nieuwe hashtags bevatten is bij het maken van het model nog helemaal geen informatie beschikbaar. Dit geldt niet alleen voor hashtags, maar ook voor 'normale' woorden. Onderwerpen die de ene week erg veel besproken worden zouden de volgende week helemaal niet meer aan bod kunnen komen, maar vervangen zijn door weer totaal nieuwe onderwerpen.
Een oplossing voor dit probleem zou het dagelijks (of misschien wekelijks) updaten van de modellen gemaakt in de offline fase zijn. Hierdoor ontstaat er op een gegeven moment een erg groot model. Het voordeel hiervan is dat er over veel onderwerpen informatie beschikbaar komt. Een nadeel is dat het raadplegen van dit model steeds langzamer gaat. Een kans opzoeken in een kleinere database gaat namelijk sneller dan dat deze vol zit met data van het afgelopen jaar. Artikel [32] heeft hier zelf al een oplossing voor gegeven: alleen de waardes van $p(w_i|t_k)$ die het meest recent zijn opgevraagd worden in het werkgeheugen opgeslagen. Mocht een kans op een gegeven moment niet meer wordt opgevraagd dan wordt deze kans uit het werkgeheugen gehaald en zou bij de volgende opvraag uit de database moeten worden gehaald. Voor een preciezere beschrijving moet het artikel zelf geraadpleegd worden.
De oplossing die is gegeven voor het te snel samenvoegen van clusters in paragraaf 4.3.2 is een mogelijke oplossing. Het zou ook nog kunnen dat door het aanpassen van de formule getoond in figuur 5 er wel een efficiëntere samenvoeging kan ontstaan. Misschien zou er een manier bedacht kunnen worden waardoor minder grotere waardes per woord worden gegeven als een cluster maar uit een paar woorden bestaat.
Andere invloeden op het algoritme kunnen de wijze van implementatie zijn of aan het gebruik van de programmeertaal Perl. Perl is namelijk goed in het omgaan met tekst, maar staat niet bekend om zijn snelheid.

5 De testfase van OCTSM

In het vorige hoofdstuk is een beschrijving van de implementatie van het OCTSM–algoritme gegeven. Hoe goed dit algoritme op tweets werkt is echter nog niet duidelijk. Dit zal pas duidelijk worden als het algoritme is getest. Hoe dit testen in zijn werk gaat, zal in dit hoofdstuk beschreven worden. Als eerst zal er gekeken worden welke testset geschikt is voor het testen van het algoritme. Vervolgens zullen de optimale waarden van de vijf parameters van het OCTSM-algoritme bepaald worden. Dit wordt gedaan door elke parameter af te zetten tegen alle combinaties van de overige parameters. De clusters die ontstaan bij het gebruik van deze optimale waarden zullen daarna nog eens verder worden onderzocht. Hierbij zullen de eventuele tekortkomingen van het algoritme of de testomgeving tot uiting komen. Hoe snel het algoritme werkt zal ook in deze laatste paragraaf besproken worden.

5.1 Een testomgeving creëren

Hoofdstuk 2 beschrijft het verzamelingproces van de uit tweets bestaande dataset. Als OCTSM echter op deze dataset wordt toegepast, kan er niet vastgesteld worden of de clusters die uit het algoritme komen, correct geformeerd zijn. In deze paragraaf wordt beschreven waarom dit is en wordt er gekeken welke dataset er wel gebruikt moet worden.

5.1.1 Welke data te gebruiken?

In optimale omstandigheden zou een menselijk oordeel over het onderwerp van elke tweet moeten worden gegeven, er van uitgaande dat mensen goed zijn in classificeren. Elke tweet zou hierdoor een klasse krijgen. Wanneer er vervolgens veel clusters ontstaan met overeenkomstige klassen, dan zou er sprake zijn van goede clustering. Dit is echter ondoenlijk. In deze dataset zitten namelijk ongeveer 11.8 miljoen tweets over verschillende periodes, waardoor er een leger aan mensen aan te pas zou moeten komen. Daarom is er voor een andere methode gekozen. Veel tweets hebben eigenlijk al een menselijk gekozen onderwerp in zich. Dit zijn namelijk de hashtags. Niet elke tweet heeft een hashtag, waardoor er al een groot gedeelte van de 11.8 miljoen tweets afvallen. Met het gebruik van deze dataset wordt er echter ook nog geen goede testomgeving gecreëerd. Als er bij deze dataset wat mis gaat, is het moeilijk te achterhalen waar het aan zou kunnen liggen. Er zijn immers erg veel onderwerpen waarbij het fout zou kunnen gaan. Om deze reden is er voor gekozen een beperkt aantal hashtags te selecteren. In het artikel waar OCTSM beschreven staat [32], werd het algoritme getest op 20 categorieën. Ook is er in deze situatie voor gekozen om dit aantal aan te houden. De hypothese die hieruit voortkomt, luidt als volgt:

Met het OCTSM-algoritme is het mogelijk klassen te clusteren die met hashtags zijn gedefinieerd.

5.1.2 Aannames bij het selecteren van de categorieën

Bij het selecteren van de categorieën wordt er gekeken naar categorieën die vaak voorkomen en die zoveel mogelijk van elkaar verschillen. De hoeveelheid is hierbij belangrijk omdat er een dataset moet ontstaan die groot genoeg is om testen op te doen. De verscheidenheid in de dataset is nodig zodat, wanneer er twee verschillende categorieën in een cluster belanden, dit ook als incorrect kan worden

bestempeld. Het is bijvoorbeeld correct om tweets van de categorieën '#ajax' en '#feyenoord' met elkaar samen te voegen. Deze twee categorieën hebben namelijk wel degelijk wat met elkaar te maken. Bepalen wat compleet verschillende categorieën zijn, is niet gemakkelijk meetbaar. Bovendien is er een beperkte keuze. Over televisieprogramma's wordt bijvoorbeeld veel getweet en kunnen erg verschillen qua onderwerp. Dit is in tegenstelling tot bijvoorbeeld '#ajax' en '#feyenoord' waar het vaak over dezelfde woorden gaat. Afgezien van het feit dat zij dezelfde sport beoefenen, spelen ze ook in dezelfde competitie en hebben hierdoor dezelfde tegenstanders. Politieke tweets werden helemaal veel in combinatie met elkaar gebruikt. Veel politieke partijen hebben vaak een mening over hetzelfde onderwerp. Tabel 4 laat zien welke categorieën er uiteindelijk uitgekozen zijn. De tweets-aantallen die hierin vernoemd zijn, zijn exclusief de tweets waarvan de klasse een combinatie vormde met een of meerdere klassen van deze tabel. Een voorbeeld van een dergelijke tweet luidt als volgt: '#ohohcherso en #penoza komen morgen op tv'. In deze tabel is te zien dat niet elke categorie even groot is. Dit komt voornamelijk omdat er niet veel categorieën waren die meer dan duizend tweets hadden en ook nog eens niet te veel op elkaar leken. Door deze verscheidenheid in grootte, kan er wel iets realistischer getest worden. In het echt zijn de categorieën ook niet altijd even groot en moet bijvoorbeeld een grote categorie zoals '#tvoh' niet zorgen voor een grote verslechtering van de resultaten.

Categorie nummer	**Hashtag/categorie**	**Aantal tweets**	**Omschrijving**
1	#tvoh	37.563	The voice of Holland: tv-programma over muziektalenten
2	#pvv	12.854	Partij voor de vrijheid: politieke partij
3	#ns	7.645	Treinmaatschappij
4	#ohohcherso	4.532	Tv-programma: Haagse jongeren op vakantie in Chersonisso.
5	ipad	2.929	Tablet van Apple: technologisch snufje
6	#sinterklaas	2.749	Feestdag voor kinderen
7	#f1 #formule1	2.337	Autosport
8	#moerdijk	2.128	Plaats van een ramp met een chemische fabriek
9	#tnt	1.710	Postbedrijf
10	#vuurwerk	1.629	Vuurwerk (tijdens oud en nieuw)
11	#tmobile	1.587	Telecombedrijf
12	#gtst	1.466	Tv-programma: soap
13	#feyenoord	1.452	Voetbalclub
14	#mulisch #harrymulisch	1.198	Overleden/legendarische Nederlandse schrijver
15	#lagerhuis	611	Tv-programma: discussieprogramma
16	#museumnacht	595	Een nacht waar de entree van musea gratis is
17	#blijdorp	579	Dierentuin
18	#nkafstanden	462	Nationale kampioenschappen schaatsen
19	#penoza	357	Tv-programma: actie-serie
20	#obama	277	President van de Verenigde Staten

Tabel 4. Een overzicht van verschillende categorieën die zich in de dataset bevinden. In dit overzicht bevinden zich ook categorieën die uit meerdere hashtags bestaan. Dit zijn twee hashtags waarvan is aangenomen dat deze naar hetzelfde onderwerp verwijzen en zijn voor deze reden samengevoegd.

5.1.3 Het creëren van meerdere datasets

Het nadeel van de hiervoor besproken aanpak is dat er nu in elke tweet van de testset een hashtag zit, die het clusteren voor het algoritme (als het goed is) erg eenvoudig maakt. Dit gecombineerd met het feit dat het correct onderscheiden van onderwerpen door alleen naar de context te kijken, een van de eigenschappen van OCTSM moet zijn, leidt tot de volgende hypothese:

OCTSM kan de tweets uitstekend bij elkaar clusteren als de hashtag in de tweet blijft zitten en kan deze tweets nog steeds redelijk goed clusteren als de hashtag eruit is gehaald.

Deze hypothese zal dus getest worden op twee verschillende testsets. Een testset, waarbij niets aan de tweets veranderd is en een testset waarbij de hashtag eruit is gehaald. Er wordt hierbij ook rekening gehouden met het aantal tweets dat per categorie wordt toegelaten. Als dit niet zou worden gedaan dan zou de categorie '#tvoh' erg dominant worden, waardoor bijvoorbeeld categorie '#obama' niet meer als een trend wordt gezien. Het doel om zoveel mogelijk trends te ontdekken zou hierdoor niet bereikt kunnen worden. Daarom is er voor gekozen om een limiet van 1.500 tweets per categorie in te stellen.
Bij het bepalen van de dataset moet er ook rekening gehouden worden met de hoeveelheid van het totaal aantal tweets. In de volgende paragraaf zullen namelijk heel wat verschillende parameters met verschillende waardes met elkaar vergeleken worden. Bij een dataset met een limiet van 1.500 tweets zijn er in totaal nog 23.497[11] tweets om te clusteren. Als dit bij elke run doorlopen moet worden, dan gaat het proces te langzaam om alles te kunnen testen. Daarom is er voor gekozen om bij het online proces in totaal maar maximaal 1.500 tweets te clusteren. Er wordt hierdoor naar het percentage per categorie gekeken. Van de categorie #obama zullen er bijvoorbeeld (277/23.497) * 1.500 ≈ 18 tweets en bij categorie #tvoh (1.500/23.497)*1.500 ≈ 96 tweets in de dataset zitten. Dit verkleint ook de kans dat als er in een later stadium een fout wordt aangetroffen, er niet heel veel tijd verloren is gegaan aan een incompleet model. Ook het offline proces wordt om dezelfde reden ingeperkt (tot 25.000 tweets). Bovendien is er bij een echte datastroom ook maar een beperkt aantal tweets voor het offline proces beschikbaar. Hierbij is er ook vanuit gegaan dat er 80% van alle tweets gebruikt wordt voor het offline proces en 20% voor het online proces.
Een van de specificaties waaraan het algoritme moet voldoen, is dat het algoritme om moet kunnen gaan met een stroom van data. De tweets die zich in de testset bevinden hebben daarom ook een tijdstip van aankomst. In de beschikbare collectie van tweets, was het echter niet mogelijk om een dataset te creëren van categorieën die dichtbij elkaar lagen. In plaats daarvan kan het voorkomen dat de tijdsintervallen tussen de tweets binnen een categorie erg van elkaar kunnen variëren. Dankzij de aanname dat wanneer er eenzelfde hashtag in meerdere tweets voorkomt, deze hetzelfde onderwerp hebben, zou het niet moeten uitmaken in welke volgorde de stroom tweets komt. Het nadeel hiervan is dat het vermogen van het algoritme om oude clusters minder te laten mee tellen hierdoor niet getest wordt. Het is echter belangrijker dat het clusteren wordt getest, aangezien er minder mis kan gaan bij

[11] De som van het aantal tweets van categorie 12 tot 20 + (11*1.500).

het vervagen van clusters. Het gaat hierbij immers alleen maar om een gewicht die de belangrijkheid van de tweets reguleert (zie paragraaf 4.2.2). Om toch nog een beetje rekening te houden met de volgorde van de tweets is de chronologische volgorde van de tweets wel aangehouden. De echte tijdstippen zijn echter niet gebruikt. Dit zou er toe leiden dat categorieën niet door elkaar maar achter elkaar aan de clusters worden toegevoegd. In een echte situatie ontstaan trends ook door elkaar. Bovendien zouden er anders ook clusters weggegooid worden, omdat ze een tijdje niet zijn geüpdate. Hierdoor zou het niet duidelijk worden welke trends er kunnen worden onderscheiden.
Als laatste moet er ook nog rekening gehouden worden met de spreiding van de tweets. Als dit niet zou gebeuren dan bestaat er een kans dat een categorie alleen in het begin voorkomt. Dit zou kunnen leiden tot een verwijdering van de clusters met deze categorie. Deze categorie is immers dan niet meer actief genoeg. Deze situatie komt voor omdat er verschil in grootte is tussen de categorieën. Als van elke categorie random een tweet aan de testset zou worden toegevoegd dan zouden de tweets van categorie '#obama' eerder op zijn dan van een grotere categorie zoals '#tvoh'. Hierbij er van uitgaande dat het random proces een verdeling gebruikt die er voor zorgt dat de categorieën even vaak aan bod komen.
In paragraaf 5.3 wordt er ook nog een test gedaan waarbij het maximale aantal te gebruiken tweets 4.600 bedraagt. Dit aantal zal in het volgende proces gebruikt worden om verwarring met het limiet van 1.500 per categorie te voorkomen. Houd ook in gedachte dat er 20 categorieën zijn en dat er maximaal 210 tweets gebruikt zullen worden voor de initiële tweets. Dit zijn de tweets die gebruikt worden voor het initiëren van de clusters. Van te voren is nog niet bekend hoeveel clusters er gebruikt gaan worden, maar het maximum is vastgesteld op 220. Hoe uiteindelijk deze gegevens gebruikt worden om een testset te maken, is beschreven in de volgende pseudocode:

1. Doe voor elke categorie het volgende:
    a. AT = neem maximaal 1.500 tweets
    b. Neem 80% van AT en gebruik deze voor de trainingset
        i. Neem de laatste (220/20≈) 11 tweets van de betreffende categorie en stop ze bij de initiële tweets
    c. Stop de overige 20% van AT in de testset

//Dit resulteert in 20 verschillende testsets (van elke categorie), 1 trainingsset en 1 set van 220 initiële //tweets.

2. INDEX = 0
3. Doe voor elke categorie het volgende:
    a. FRAQ = totaal aantal tweets in de testset voor deze categorie/ de som van het aantal tweets van de testsets van alle categorieën
    b. ATT = FRAQ x 4.600
    c. Neem ATT tweets van deze categorie en stop deze in een array met index INDEX
    d. INDEX = INDEX +1

//Dit resulteert in 20 verschillende arrays met een index variërend van 0 tot 19.

4. Creëer een array genaamd INDEX_ARRAY
5. Doe voor elk gecreëerde array:
    a. A = aantal tweets van de betreffende array met index INDEX
    b. Voeg A keer het index nummer INDEX toe aan de INDEX_ARRAY

//Dit resulteert in een INDEX_ARRAY met 4.600 cijfers variërend van 0 tot 19. Hoe meer een categorie //voorkomt hoe meer er van deze INDEX in deze array zitten.

6. Doe het volgende net zolang totdat er geen element meer in INDEX_ARRAY over is:
    a. K = neem een random element uit de INDEX_ARRAY
    b. T = neem tweet uit de array met index K
    c. Voeg T toe aan de uiteindelijke testset

//Uiteindelijk resulteert dit in een testset van categorieën die random verspreid zijn.

7. Afhankelijk van het aantal cluster dat nodig is, pak een aantal random tweets uit initiële tweets

De uiteindelijke aantallen van elke categorie per testset zijn in tabel 5 te zien. Deze testsets worden in de volgende paragraaf gebruikt om optimale parameters van het OCTSM-algoritme te vinden en om de clusterkwaliteit te testen.

Categorie	Aantal tweets bij een limiet van 4.600	Aantal tweets bij een limiet van 1.500
#pvv	296	73
#tvoh	296	77
#ns	292	76
#ohohcherso	297	68
#ipad	295	95
#sinterklaas	293	78
#f1	296	88
#moerdijk	296	75
#tnt	293	73
#vuurwerk	297	92
#tmobile	295	88
#feyenoord	283	81
#gtst	273	85
#mulisch	233	81
#museumnacht	118	65
#lagerhuis	120	65
#blijdorp	114	59
#nkafstanden	91	57
#penoza	68	70
#obama	55	55

Tabel 5. Het totaal aantal tweets dat geselecteerd is per categorie per testset.

5.2 Het optimaliseren van de parameters

Nu er een testomgeving is gecreëerd, kan het OCTSM-algoritme worden getest. Het doel hiervan is om erachter te komen hoe de resultaten reageren in verschillende omstandigheden. Hierbij zal gekeken worden hoe verschillende variaties van de parameters reageren op verschillende testsets. De optimale waarde van de parameters zullen hierbij ook bepaald worden. Pas als er bekend is wat de optimale parameters zijn, kan er uitgesloten worden dat de parameters een negatieve invloed hebben op de

clustering. Hierbij is het ook mogelijk dat de verschillende parameters afhankelijk van elkaar zijn. Om dit te testen zal elke parameter afgezet worden tegen alle combinaties van de overige parameters.
In het oorspronkelijke algoritme is er gebruik gemaakt van vier verschillende parameters. In dit onderzoek is er echter nog een bijgekomen. Bij het implementeren van het oorspronkelijke algoritme bleek namelijk al snel dat het erg langzaam ging en dat er erg snel verschillende categorieën bij elkaar werden gevoegd. Hierdoor is er besloten om alleen naar de belangrijkste woorden te kijken. Dit heeft geresulteerd in een extra parameter genaamd 'top %' (zie paragraaf 4.3.2). Om aan te tonen dat dit inderdaad nodig is, zal als eerst een paar experimenten plaatsvinden met het oorspronkelijk algoritme. Hierna zullen de parameters lambda, mergefactor, minfactor, top % en het aantal clusters verder onderzocht worden.

5.2.1 Mankementen van het originele algoritme

In paragraaf 4.3 is aan een aantal voorbeelden te zien dat het originele algoritme niet optimaal werkt. Er zijn ook een aantal parameters gevarieerd om te kijken hoe goed het originele algoritme het nu daadwerkelijk doet. Het betreft hier echter niet veel variaties, aangezien het veel langer heeft geduurd om het originele algoritme te testen. De waardes die wel gebruikt zijn, zijn te zien in tabel 6. Dit is ook maar getest op een testset van 500 tweets.

Parameter	Verschillende waardes
Mergefactor	0.4 0.7
Lambda	0.4 0.7
Aantal clusters	40 100
Minfactor	0.000125 0.00025

Tabel 6. Het aantal verschillende waardes, dat gebruikt is voor het testen van het OCTSM in zijn originele vorm.

De resultaten van de parametercombinaties die het beste uitvielen zijn te zien in tabellen 7 en 8. Als verwacht is het aantal ontdekte trends met hashtag niet erg hoog. Een hashtag in de testset zou het erg makkelijk voor het algoritme moeten maken. Het algoritme vindt er gemiddeld slechts iets minder dan de helft. Zonder hashtag worden onder maximale omstandigheden gemiddeld slechts 3.6 trends gevonden. Deze resultaten kunnen wellicht hoger uitvallen als er meer verschillende parameterwaardes zouden worden getest, maar aangezien er beperkte tijd is en het aangepaste algoritme meer belovende resultaten produceert, zal er niet verder getest worden met dit originele algoritme.

Lambda	Aantal clusters	Minfactor	Mergefactor	Gemiddeld aantal verschillende topics ontdekt	Aantal geclusterde tweets * F-measure
0.7	**100**	**0.00025**	**0.4**	9.986	58.725
0.7	**100**	**0.000125**	**0.4**	9.986	58.725
0.7	**100**	**0.00025**	**0.7**	9.986	58.725
0.7	**100**	**0.000125**	**0.7**	9.986	58.725
0.4	**100**	**0.000125**	**0.7**	9.605	232.833
0.4	**100**	**0.00025**	**0.7**	6.279	56.680

Tabel 7. De zes best presterende parametercombinaties van de testset met hashtag.

Lambda	Aantal clusters	Minfactor	Mergefactor	Gemiddeld aantal verschillende topics ontdekt	Aantal geclusterde tweets * F-measure
0.4	**100**	**0.000125**	**0.7**	3.617	82.411
0.7	**100**	**0.00025**	**0.4**	1.409	28.918
0.7	**100**	**0.000125**	**0.4**	1.409	28.918
0.7	**100**	**0.00025**	**0.7**	1.395	28.887
0.7	**100**	**0.000125**	**0.7**	1.395	28.887
0.4	**100**	**0.00025**	**0.7**	1.250	71.913

Tabel 8. De zes best presterende parametercombinaties van de testset zonder hashtag.

5.2.2 Parameters van de nieuwe versie

De datasets zullen per parameter worden bekeken. Telkens als er een optimale parameter is berekend zal deze een vaste waarde krijgen voor het verloop van de tests. Het aantal parameters dat is gebruikt, is beperkt omdat hiermee veel rekentijd bespaard wordt. Tabel 9 toont de waardes die uiteindelijk gebruikt zijn. De verschillende waardes uit deze tabel resulteren in (4 x 4 x 3 x 4 x 3 =) 576 runs per dataset (1.500 tweets met of zonder hashtag). Er zijn dus heel wat combinaties van parameters mogelijk. Het zou erg veel werk kosten om al deze combinaties afzonderlijk te bekijken. Daarom is er voor gekozen om een parameter tegenover de rest van de combinaties te zetten. Als vervolgens de verschillende waarden van een bepaalde parameter alleen verschillen in hoogte, maar in fluctuaties hetzelfde zijn dan kan er gezegd worden dat deze waarde constant beter presteert. Dit geeft echter geen goed overzicht van wat er nu daadwerkelijk gebeurt. Op de x-as komen namelijk de nummers van de verschillende runs die verwijzen naar een bepaalde combinatie van parameters. Om deze reden wordt per parameter alleen gekeken naar het gemiddelde van alle andere combinaties.

Parameter	Verschillende waardes
Mergefactor	0.25 0.45 0.65 0.85
Lambda	0.25 0.45 0.65 0.85
Aantal clusters	40 80 100
Top %	0.04 0.07 0.15 0.3
Minfactor	0.001 0.003 0.006

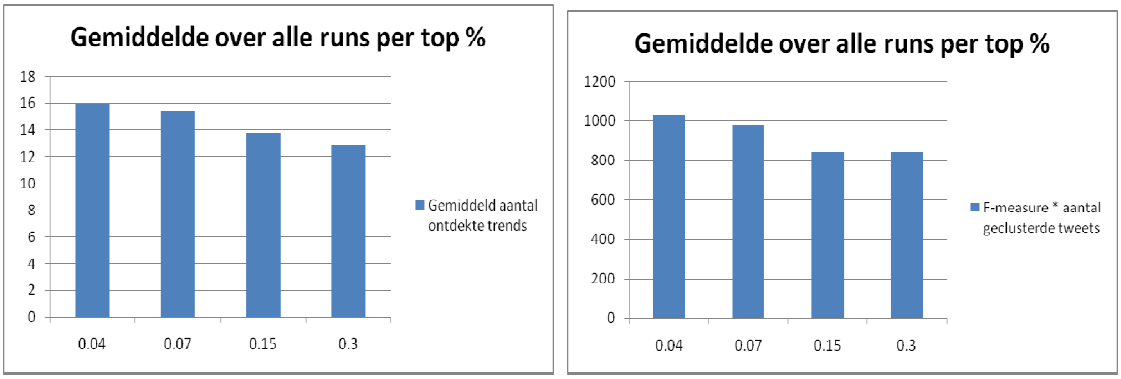
Tabel 9. Het aantal verschillende waardes, dat gebruikt is voor het testen van het OCTSM in zijn aangepaste vorm.

Top %

Het top percentage zal voor de testsets met en zonder hashtag apart bekeken worden. In de figuren 10 en 11 zijn de resultaten te zien van de hierboven beschreven procedure.

Met hashtag

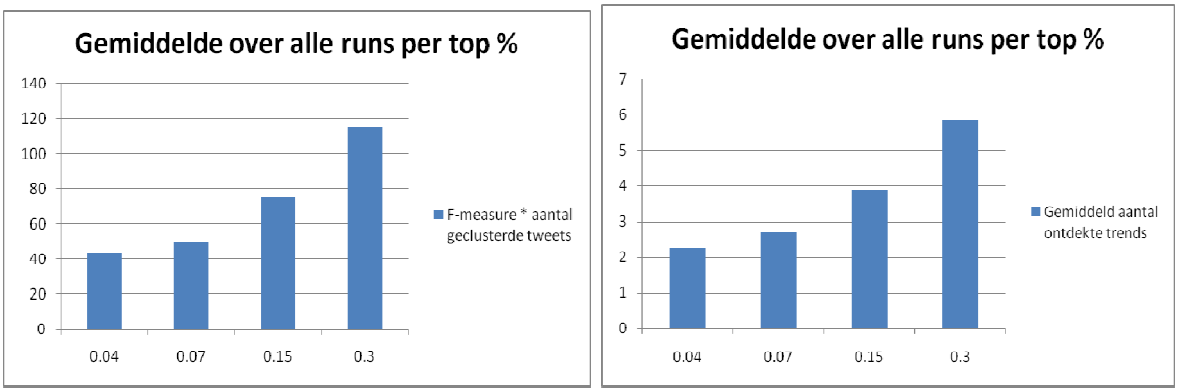
Het top % van 7% en 4% gaat beter dan dat van 30% en 15%. Dit komt doordat er bij 30% en 15% niet alleen gefocust wordt op de hashtag, maar ook op andere woorden. Hierdoor komt er dus 'ruis' in de dataset. Bij 7% en 4% is de hashtag altijd het belangrijkste woord, waardoor in de beginfase (cluster is nog klein) er alleen tweets bij mogen die ook deze hashtag bevatten. Bij een top % van 4% mag er namelijk naast het belangrijkste woord pas naar andere woorden worden gekeken als er meer dan 25 woorden (voor berekening zie paragraaf 4.3.2) in het cluster zitten. Hierna kwamen er ook andere woorden bij, waardoor er weer een beetje ruis in voorkwam. Als het % heel erg klein wordt dan wordt perfectie bereikt, omdat er dan alleen nog maar gekeken wordt naar het belangrijkste woord in het cluster. Dit is altijd de hashtag van de gespecificeerde categorieën, omdat deze in de offline fase met veel frases in combinatie is gebracht.



Figuur 10. De gemiddelde waarde over alle runs ten opzichte van parameter top %. De testset die hierbij gebruikt is bevat wel een hashtag. Links is het gemiddelde aantal ontdekte trends weergegeven en rechts het gemiddeld aantal correct geclusterde tweets.

Zonder hashtag

De dataset zonder hashtags heeft het tegenovergestelde resultaat ten opzichte van de testset met hashtags. Hier ontbreken de hashtags, waardoor het algoritme niet alleen naar het meest kenmerkende woord moet kijken. Deze is immers verwijderd. In deze situatie moet het algoritme juist zijn informatie halen uit de overige woorden die met het onderwerp te maken hebben. Het grootste percentage waarmee getest is (30%) geeft ook het beste resultaat. Het zou best kunnen dat een groter percentage een nog beter resultaat geeft.



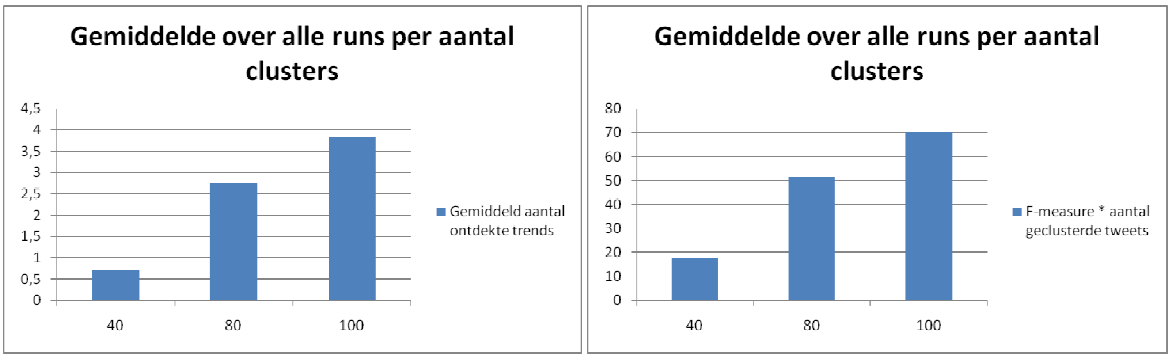
Figuur 11. De gemiddelde waarde over alle runs ten opzichte van parameter top %. De testset die hierbij gebruikt is bevat geen hashtag. Links is het gemiddelde aantal ontdekte trends weergegeven en rechts het gemiddeld aantal correct geclusterde tweets.

Aantal clusters

Uit de analyse met het top % is gebleken dat de dataset met een hashtag het best presteert met een top van 4% en de testset zonder hashtag het best presteert met een top % van 30%. Deze waarden zullen nu als vaste waarde dienen in de rest van de analyse van de parameters. Nu zal de variatie in het aantal clusters bekeken worden. De resultaten hiervan zijn te zien in de figuren 12 en 13. Eerst zal de testset zonder hashtag besproken worden en daarna met.

<u>Zonder hashtag</u>

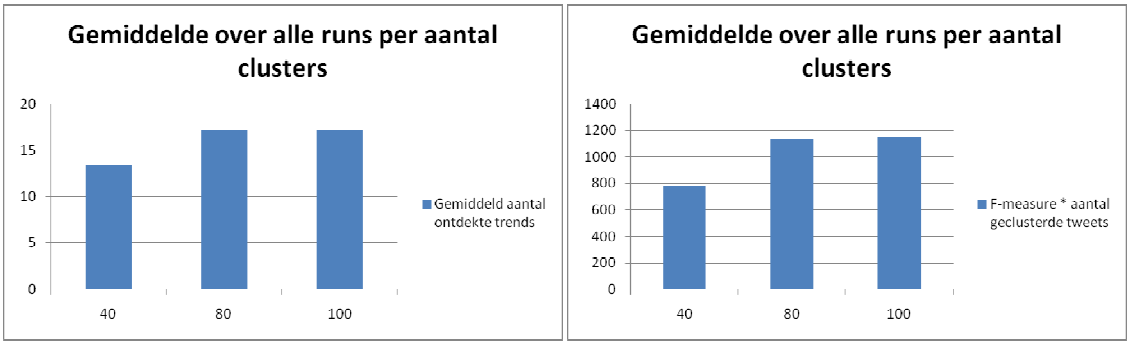
Bij een groot aantal clusters heeft een net nieuw gecreëerd cluster (bestaande uit een net toegevoegde tweet) genoeg tijd om zich aan een ander cluster toe te voegen. Hoe kleiner het aantal clusters is, des te minder tijd een cluster heeft om zich met een ander cluster samen te voegen. Er geldt immers: het cluster dat het langst bestaat zonder dat er een tweet is bijgevoegd, wordt verwijderd. Hierdoor wordt de tendens van figuur 12, hoe meer clusters er zijn hoe beter het algoritme presteert, verklaard.



Figuur 12. De gemiddelde waarde over alle runs ten opzichte van parameter *aantal clusters* met een top % van 30%. De testset die hierbij gebruikt is bevat geen hashtag. Links is het gemiddelde aantal ontdekte trends weergegeven en rechts het gemiddeld aantal correct geclusterde tweets.

Met hashtag

Uit figuur 13 is op te maken dat er bijna geen verschil is tussen 80 en 100 clusters en dat het algoritme minder presteert bij het gebruik van 40 clusters. Hierbij is een top % gebruikt van 4%, waardoor de clusters snel met elkaar worden samengevoegd. Het algoritme wordt het op deze manier namelijk makkelijk gemaakt omdat alleen naar de hashtag gekeken hoeft te worden. Hierdoor is er minder opvang (in een nieuw cluster) nodig voor tweets die later eventueel ergens bijgevoegd hoeven te worden. Op een gegeven moment is er een limiet aan het aantal clusters dat nodig is (het resultaat wordt niet meer beter). Deze grens is te zien als naar het verschil in 80 en 100 clusters wordt gekeken.



Figuur 13. De gemiddelde waarde over alle runs ten opzichte van parameter *aantal clusters* met een top % van 4%. De testset die hierbij gebruikt is bevat wel een hashtag. Links is het gemiddelde aantal ontdekte trends weergegeven en rechts het gemiddeld aantal correct geclusterde tweets.

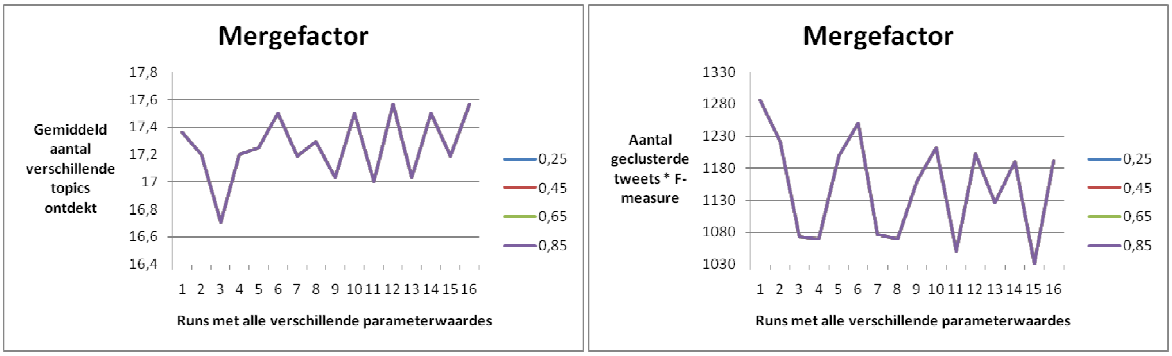
Mergefactor

Uit de analyse met het aantal clusters is gebleken dat de datasets met en zonder hashtag beide bij het gebruik van 100 clusters het best presteren. Deze waarden zullen nu als vaste waarde dienen in de rest van de analyse van de parameters. Nu zal de variatie bij de mergefactor bekeken worden. De resultaten hiervan zijn te zien in de figuren 14 en 15. Eerst zal de testset met hashtag besproken worden en daarna zonder.

Met hashtag

In tegenstelling tot de vorige grafieken in dit hoofdstuk, is in figuur 14 de mergefactor afgezet tegen het totale aantal combinaties van de overige parameters. In deze figuur is te zien dat alle verschillende waarden van de mergefactor precies over elkaar heen vallen. Ook als er een grafiek was geweest van de gemiddelden dan was er geen verschil geweest. Dit betekent dat er bij variatie van deze parameter geen verschil in de resultaten optreedt. Dit komt doordat de nieuwe parameter top % de functie van deze parameter zowat overneemt. Bij het samenvoegen van clusters wordt er namelijk eerst gekeken of de woorden die de meeste waarde hebben, bij beide clusters, overeenkomen. Op deze gevonden matches

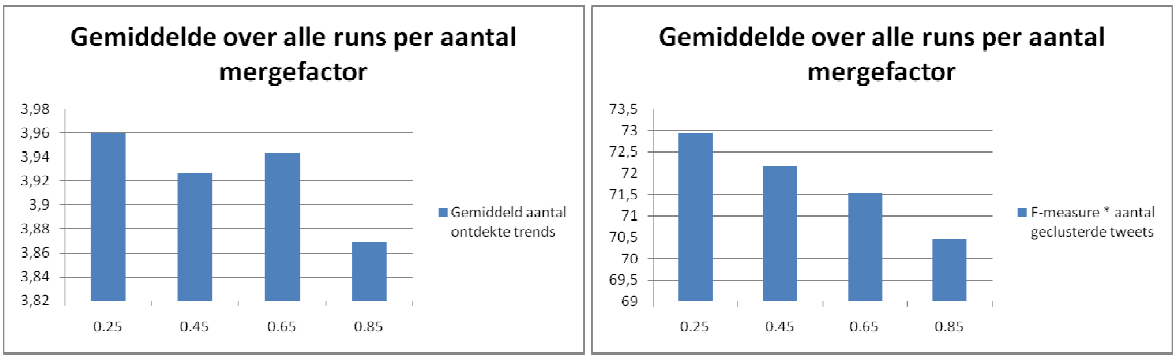
wordt pas de vergelijking gedaan waar mergefactor een rol in speelt. Aangezien de woorden die nu vergeleken worden allemaal in de top van het cluster zitten, betekent dit dat ze allemaal een grote waarden hebben waardoor er altijd een match ontstaat. Hierdoor kan de mergefactor niet goed getest worden. Wellicht kan er in een later stadium zonder top % getest worden zodat ook voor deze parameter de optimale waarde bepaald kan worden.



Figuur 14. De waardes over alle runs ten opzichte van parameter *mergefactor* met een top % van 4% en het *aantal cluster* op 100. De testset die hierbij gebruikt is bevat wel een hashtag. Links is het gemiddelde aantal ontdekte trends weergegeven en rechts het gemiddeld aantal correct geclusterde tweets.

<u>Zonder hashtag</u>

In de testset zonder hashtag speelt de mergefactor wel een rol. Bij deze testset is er namelijk voor gekozen naar een hoger toppercentage te kijken (30%). Hierdoor krijgt de mergefactor weer een grotere rol. De verschillen zijn nog wel erg klein, maar uit de grafieken is af te lezen dat mergefactor met waarde 0.25 een iets beter resultaat geeft. Er kan dus geconcludeerd worden dat hoe hoger het toppercentage is, des te meer invloed de mergefactor krijgt. Het is mogelijk dat het algoritme nog beter presteert op deze testset als er een toppercentage hoger dan 30% wordt gebruikt. Hierdoor zou de mergefactor weer iets meer invloed krijgen.



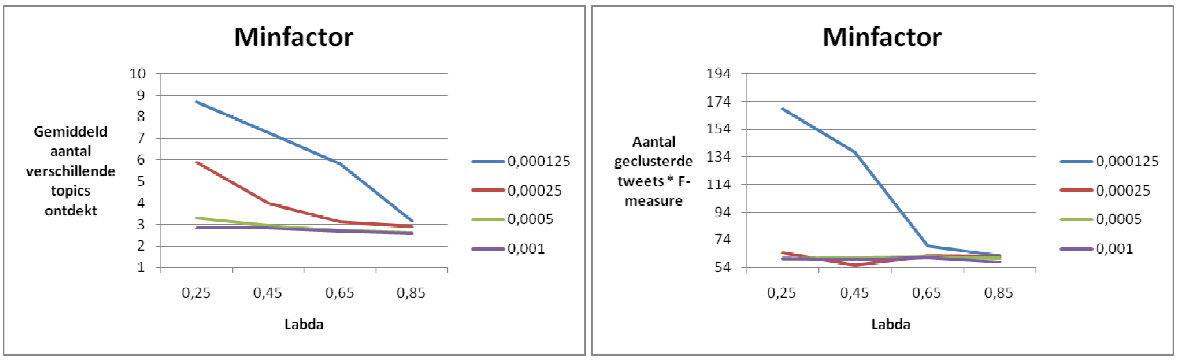
Figuur 15. De gemiddelde waarde over alle runs ten opzichte van parameter *mergefactor* met een top % van 30% en het *aantal cluster* op 100. De testset die hierbij gebruikt is bevat geen hashtag. Links is het gemiddelde aantal ontdekte trends weergegeven en rechts het gemiddeld aantal correct geclusterde tweets.

Lambda en minfactor

Uit de analyse van de mergefactor is gebleken dat deze parameter niet veel invloed heeft, maar aangezien 0.25 bij de testset zonder hashtag een iets beter resultaat gaf is deze waarde gekozen als vaste parameter. Dezelfde waarde is gekozen bij de testset met hashtag. Dit had echter net zo goed een andere waarde kunnen zijn, aangezien het resultaat toch hetzelfde blijft. Er zijn nu nog maar twee waardes over. Deze zullen tegelijk behandeld worden. De resultaten hiervan zijn te zien in de figuren 16 en 17. Eerst zal de testset zonder hashtag besproken worden en daarna met.

Zonder hashtag

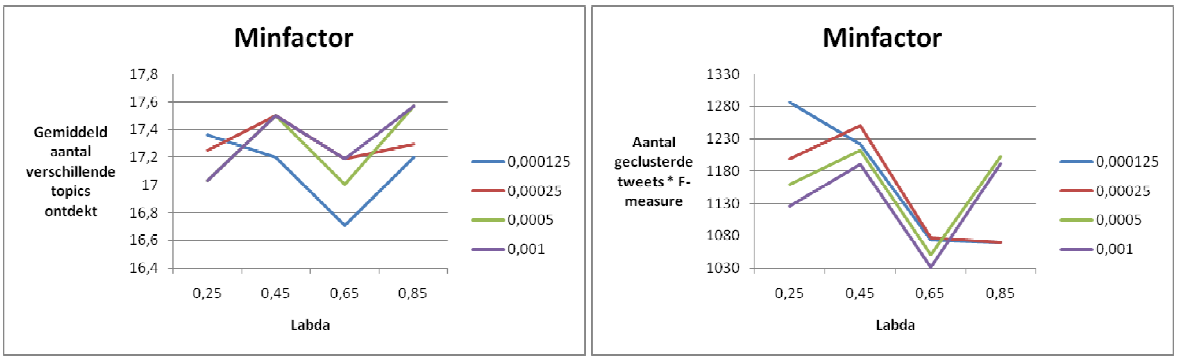
Er zijn nu nog maar twee verschillende parameters over. Hierdoor kunnen ze nu tegenover elkaar gezet worden. Voor alle twee de maten geldt dat bij elke waarde van lambda, 0.000125 de best presterende minfactor is. Deze waarde gecombineerd met een lambda van 0.25 geeft de maximale waarde. Een nog lagere waarde van minfactor geeft wellicht een nog beter resultaat. Hoe hoger lambda hoe zwaarder de frases worden meegeteld ('topic signature translation model'). Deze frases moeten, kijkend naar de resultaten, niet al te veel worden meegerekend.



Figuur 16. Parameter *minfactor* ten opzichte van parameter *minfactor* met een top % van 30%, het *aantal cluster* van 100 en een *mergefactor* van 0.25. De testset die hierbij gebruikt is bevat geen hashtag. Links is het gemiddelde aantal ontdekte trends weergegeven en rechts het gemiddeld aantal correct geclusterde tweets.

Met hashtag

Bij deze testset is wel te zien dat het zwaarder meewegen van het 'topic signature translation model' een positieve uitwerking kan hebben. Als lambda de waarde 0.85 heeft dan worden de meeste trends gevonden. Een hoger aantal trends ontdekken is meer waard dan het aantal tweets clusteren. Dit is de reden dat voor de combinatie voor lambda de waarde 0.85 is gekozen en voor minfactor de waarde 0.001 is gekozen.



Figuur 17. Parameter *minfactor* ten opzichte van parameter *minfactor* met een top % van 30%, het *aantal cluster* van 100 en een *mergefactor* van 0.25. De testset die hierbij gebruikt is bevat wel een hashtag. Links is het gemiddelde aantal ontdekte trends weergegeven en rechts het gemiddeld aantal correct geclusterde tweets.

In tabel 10 zijn de optimale parameters in een overzicht geplaatst. Nu deze bekend zijn kan in de volgende paragraaf specifieker naar de resultaten gekeken worden.

	Limiet van 1.500, met hashtag	**Limiet van 1.500, zonder hashtag**
Top %	4	30
Aantal clusters	100	100
Mergefactor	0.25	0.25
Minfactor	0.001	0.000125
Lambda	0.85	0.25

Tabel 10. Een overzicht van de gevonden optimale parameters voor beide testsets.

5.3 Aantal geclusterde tweets per categorie

De experimenten die in deze paragraaf besproken gaan worden zullen gebruikmaken van de optimale parameters, gevonden in de vorige paragraaf. Als de resultaten van deze experimenten vervolgens tegenvallen dan kan het in ieder geval niet aan de parameterinstellingen liggen. In deze paragraaf zullen deze parameters worden toegepast op een testset van 1.500 en 4.600 tweets. De resultaten van deze twee testsets komen echter redelijk met elkaar overeen. Daarom zullen de resultaten van de testset met 1.500 tweets getoond worden en zal er alleen gerefereerd worden naar de resultaten van de andere testset. De daadwerkelijke tabellen waarin deze resultaten worden getoond, zijn te vinden in bijlage II voor de dataset met hashtag en in bijlage III voor de dataset zonder hashtag. De resultaten van de twee verschillende datasets zullen nu apart bekeken worden.

5.3.1 De dataset met hashtag

In tabel 11 is voor de testset met hashtag per categorie aangegeven hoeveel procent van de tweets goed geclusterd zijn. Een tweet is hierbij goed geclusterd als deze zich in een cluster bevindt, die voor meer dan 50% bestaat uit tweets van dezelfde klasse en als er zich in totaal meer dan 3 tweets in dit cluster bevinden. Hierbij wordt ook aangegeven over hoeveel verschillende clusters dit goede aantal geclusterde tweets verdeeld is.

Alle categorieën worden bij dit experiment gevonden en 79.3% van alle tweets zijn aan een correct cluster toegevoegd. In tabel 12 is per categorie aangegeven hoeveel tweets er van elke categorie in een cluster zijn gekomen dat gedomineerd wordt door een andere klasse (aantal tweets van deze andere klasse is meer dan 50%). Bij deze testset zijn het aantal incorrect geclassificeerde tweets erg klein. Hier is namelijk maar 0.9% van de gehele testset in het verkeerde cluster terechtgekomen. Er is ook maar 19.8% (100 - 79.3 - 0.9) van de tweets niet in de uiteindelijke clusters terecht gekomen. Deze tweets hebben in een cluster gezeten die op een gegeven moment is verwijderd. Dit percentage is niet het belangrijkst. Het is veel belangrijker dat het aantal correct geclusterde tweets goed over de verschillende categorieën verspreid zijn en dat het aantal 'false positives' tot een minimum is gebracht. Zolang er namelijk van elke categorie een aantal tweets bij elkaar in een cluster zijn gestopt kan er geconstateerd worden dat er een trend is gevonden. Tot slot is er ook nog tabel 13. Als er nog clusters waren geweest die wel meer dan 3 tweets bevatten, maar niet in het bezit waren van een dominante categorie dan zou dit hier getoond worden. Mochten hier wel een aantal tweets vermeld worden, dan is dit een teken dat meerdere categorieën bij elkaar zijn samengevoegd.

Categorie	Input	Aantal correct geclusterd	Verdeeld over aantal clusters	%goed geclusterd
#ns	98	97	1	99.0
#blijdorp	36	35	1	97.2
#tmobile	98	93	1	94.9
#pvv	96	88	1	91.7
#tnt	97	87	1	89.7
#sinterklaas	97	80	1	82.5
#nkafstanden	31	25	1	80.6
#feyenoord	93	75	1	80.6
#lagerhuis	40	32	1	80.0
#ipad	98	78	1	79.6
#moerdijk	93	74	1	79.6
#museumnacht	39	31	1	79.5
#f1	95	74	1	77.9
#tvoh	96	74	2	77.1
#penoza	24	18	1	75.0
#vuurwerk	94	70	1	74.5
#ohohcherso	96	61	2	63.5
#gtst	85	48	1	56.5
#mulisch	78	44	1	56.4
#obama	17	6	1	35.3
Totaal	**1501**	**1190**		**79.3**

Tabel 11. Een overzicht van de resultaten van de correct geclusterde tweets bij de testset met hashtag.

Categorie	Input	% in verkeerde cluster	Aantal false positives
#gtst	85	3.5	3
#ohohcherso	96	2.1	2
#ns	98	1.0	1
#tmobile	98	1.0	1
#tnt	97	1.0	1
#pvv	96	1.0	1
#tvoh	96	1.0	1
#f1	95	1.1	1
#penoza	24	4.2	1
#feyenoord	93	1.1	1
#vuurwerk	94	1.1	1
Totaal		**0.9**	**14**

Tabel 12. Een overzicht van de resultaten van de incorrect geclusterde tweets bij de testset met hashtag.

Aantal tweets in een cluster zonder dominante klasse	Aantal clusters
0	0

Tabel 13. Een overzicht van het aantal clusters dat wel meer dan 3 tweets heeft maar geen dominante categorie bevat.

Bij de experimenten met de testset van 4.600 tweets worden niet alle categorieën gevonden. De categorieën '#tnt', '#vuurwerk', '#tvoh', '#mulisch' en '#lagerhuis' worden hierbij zowat niet gevonden. Alleen de categorie '#tvoh' en '#mulisch' hebben nog een cluster met ieder 4 tweets. Dit is echter te weinig om te stellen dat er een echt goede trend gevonden is. De meeste tweets van deze categorieën zijn waarschijnlijk samengevoegd in een cluster. Er bestaat namelijk een cluster dat bestaat uit 733 tweets die geen dominante waarde heeft. Door de misclassificatie van deze categorieën valt het percentage correct geclassificeerde tweets wat minder uit dan de testset met 1.500 tweets. Dit percentage bedraagt namelijk 58.3%, het aantal incorrect geclassificeerde tweets bedraagt 22.2% en het aantal tweets dat niet geclassificeerd is bedraagt 19.5%. De 22.2% klinkt erger dan het is. Het grootste gedeelte hiervan is namelijk bij elkaar in een cluster gevoegd, waardoor de andere categorieën er minder onder lijden.

5.3.2 De dataset zonder hashtag

Het algoritme wist de categorieën bij de dataset met hashtag nog redelijk te onderscheiden in de vorige paragraaf. Bij de dataset zonder hashtag is dit echter niet het geval. In deze situatie zijn er slechts zeven categorieën die onderscheiden zijn (zie tabel 14). De aantallen waarmee dit gebeurt is ook nog eens beperkt. Het percentage dat het aantal verkeerd geclusterde tweets aangeeft is het dubbele van het percentage goed geclusterde tweets (zie tabel 14 en 15). Hierbij zijn er ook nog eens vijf clusters die een foutieve trend aangeven (zie tabel 16). Ook als er gekeken wordt naar de testset van 4.600 tweets, zit er niet veel verbetering in. Hier zijn er wel iets meer trends ontdekt (ongeveer 7 trends), maar hier is het aantal foutieve clusters ook gestegen (naar 9 trends). Het percentage fout geclusterde tweets en goed geclusterde tweets ligt redelijk op hetzelfde niveau als de testset van 1.500 tweets met respectievelijk 14.3% en 27.2% van de tweets.

Tot slot is in tabel 17[12] nog de tijd vermeld dat het algoritme nodig heeft om een tweet te verwerken. Hierbij geldt hoe beter het aantal tweets verspreid zijn over het aantal clusters, des te sneller het gaat. Als er heel grote clusters ontstaan dan betekent dit dat er telkens tweets aan dezelfde clusters worden toegevoegd. Bij elke verandering van deze clusters moeten de gewichten van alle woorden in dit cluster worden bijgewerkt. Grote clusters bevatten namelijk veel woorden, waardoor er veel updates plaats moeten vinden. De dataset met hashtags die uit 1.500 tweets bestaat, is het beste verspreid over de clusters. Er is geen cluster die ver boven de andere clusters uitsteekt qua aantallen tweets. Deze testset verwerkt gemiddeld dan ook de meeste tweets per seconde.

5.4 Conclusies die aan de resultaten verbonden kunnen worden

De hypothese die opgesteld is in paragraaf 5.3.1 kan voor het grootste gedeelte verworpen worden. Het OCTSM-algoritme kan namelijk het grootste gedeelte van de categorieën goed onderscheiden als de tweets een hashtag bevatten, maar dit is niet perfect. Ook het aantal tweets dat correct is geclassificeerd

[12] De snelheden die in deze tabel worden vermeld zijn afhankelijk van de snelheid van de gebruikte computer. Als dit algoritme later in gebruik zal worden genomen, zal het op dezelfde computer worden gedraaid als deze snelheden zijn gemeten.

is verre van perfect. De 79.3% van de testset van 1.500 tweets komt redelijk in de buurt, maar de 58.3% is hier al een stuk verder van verwijderd. Als de hashtags eenmaal uit de tweets zijn weggehaald, is het algoritme bijna niet meer in staat om de verschillende categorieën te onderscheiden. Hoewel het weglaten van de hashtag een testomgeving creëert die niet met de werkelijkheid overeenkomt, is het toch wel een signaal dat dit algoritme zoveel mogelijk informatie nodig heeft om de tweets correct te clusteren.

Categorie	Input	Aantal correct geclusterd	Verdeeld over aantal clusters	%goed geclusterd
#tnt	97	52	2	53.6
#ns	98	50	1	51.0
#f1	95	29	1	30.5
#sinterklaas	97	27	3	27.8
#ohohcherso	96	22	1	22.9
#tmobile	98	10	2	10.2
#mulisch	78	9	1	11.5
#gtst	85	3	1	3.5
#nkafstanden	31	0	0	0
#moerdijk	93	0	0	0
#obama	17	0	0	0
#blijdorp	36	0	0	0
#pvv	96	0	0	0
#ipad	98	0	0	0
#feyenoord	93	0	0	0
#penoza	24	0	0	0
#vuurwerk	94	0	0	0
#museumnacht	39	0	0	0
#tvoh	96	0	0	0
#lagerhuis	40	0	0	0
Totaal	**1501**	**202**		**13.5**

Tabel 14. Een overzicht van de resultaten van de correct geclusterde tweets.

Categorie	Input	% in verkeerde cluster	Aantal false positives
#pvv	96	63.5	61
#ipad	98	50.0	49
#moerdijk	93	45.2	42
#feyenoord	93	34.4	32
#blijdorp	36	77.8	28
#tmobile	98	26.5	26
#mulisch	78	28.2	22
#penoza	24	75.0	18
#vuurwerk	94	18.1	17
#ns	98	16.3	16
#gtst	85	18.8	16
#f1	95	14.7	14
#sinterklaas	97	14.4	14
#lagerhuis	40	25.0	10
#nkafstanden	31	32.3	10
#tvoh	96	7.3	7
#obama	17	41.2	7
#tnt	97	7.2	7
#museumnacht	39	12.8	5
#ohohcherso	96	4.2	4
Totaal		**27.0**	**405**

Tabel 15. Een overzicht van de resultaten van de incorrect geclusterde tweets bij een testset zonder hashtag.

Aantal tweets in een cluster zonder dominante klasse	Aantal clusters
315	5

Tabel 16. Een overzicht van het aantal clusters dat wel meer dan 3 tweets heeft maar geen dominante categorie bevat, bij een testset zonder hashtag.

Met of zonder hashtag	Aantal verwerkte tweets	Verstreken tijd (s)	Aantal Tweets/seconde
met	**1.500**	280	5.36
zonder	**1.500**	400	3.75
met	**4.600**	3.288	1.40
zonder	**4.600**	1.206	3.81

Tabel 17. Een overzicht van de snelheid van het OCTSM-algoritme op de verschillende testsets.

Een echte stroom van tweets bestaat voor een groot deel uit tweets die niet tot een duidelijke trend behoren. Hierdoor kan de werkelijkheid in zekere mate vergeleken worden met de testset zonder hashtags. Ook hier is namelijk weinig informatie over het achterliggende onderwerp beschikbaar. In paragraaf 5.2.2 wordt onderbouwd waarom er bij deze dataset een groot aantal clusters nodig is om het optimale resultaat te behalen. Nu is er echter nog een beperkte dataset gebruikt. Als het om een echte stroom met data zou gaan dan spelen er nog veel meer categorieën een rol, waardoor het aantal clusters alleen maar zou moeten toenemen. Het nadeel hiervan is dat er bij elke toevoeging van een tweet, alle clusters met het veranderde cluster vergeleken moeten worden, waardoor er bij grote aantallen clusters de rekentijd erg snel zou toenemen. Merk op dat niet alle clusters opnieuw met elkaar vergeleken hoeven worden. De kans tussen clusters die niet veranderd zijn, blijft namelijk hetzelfde.

Het feit dat er wel een goede clustering plaatsvindt als de hashtags zich wel in de tweets bevinden, kan betekenen dat als de echte stroom wordt beperkt, dit algoritme wel toepasbaar zou worden. Het 'burst-algoritme' uit paragraaf 3.3.8 zou bijvoorbeeld voor deze beperking van de dataset kunnen dienen. Dit algoritme ontdekt alleen de woorden die ten opzichte van het verleden opeens veel voorkomen. Als deze woorden eenmaal gedetecteerd zijn dan zou OCTSM per trend kunnen vaststellen welke woorden belangrijk zijn voor deze trend. Het nadeel van het 'burst-algoritme' is namelijk het feit dat niet meteen gezien kan worden of bepaalde woorden bij elkaar horen, of dat ze twee aparte trends zijn. Een mogelijke oplossing die hieruit voortkomt, is dat het OCTSM-algortime alleen de subcategorieën binnen een bepaalde hashtag ontdekt.

6 Conclusies

Het doel van de stage is primair het ontwikkelen van een online clusteringmethode; daarnaast dient aandacht besteed te worden aan verzameling van posts en aan de online presentatie van resultaten. Er is voor gekozen om voor dit onderzoek alleen tweets als input te gebruiken. Hierbij gaat het alleen om de analyse van tweets uit Nederland en moet de applicatie in de programmeertaal Perl worden gemaakt. Dit hoofdstuk zal de conclusies van deze stage beschrijven en de volgende onderzoeksvraag samen met de vijf bijhorende deelvragen beantwoorden:

Hoe kunnen aan de hand van een clusteringmethode de meest voorkomende onderwerpen, op Twitter of andere bronnen van een constante informatiestroom, in kaart worden gebracht?

1. Hoe worden de posts van de constante datastroom verzameld?
2. Zijn er al bestaande technieken op het gebied van topicdetectie d.m.v. clustering en zijn er al methodes voor het clusteren van tweets en reacties op forums, nieuwswebsites en blogs?
3. Hoe wordt een eventueel gevonden clusteringtechniek toegepast?
4. Hoe wordt de gevonden clusteringtechniek geëvalueerd?
5. Hoe worden de gevonden onderwerpen gerepresenteerd?

1. Hoe worden de posts van de constante datastroom verzameld?

De eerste deelvraag is in hoofdstuk 2 beantwoord. Bij Twitter is een account opgevraagd, waardoor de tweets van 100.000 gebruikers onderschept konden worden. Voor het onderzoek zijn alleen Nederlandse tweets relevant. Om deze tweets op te sporen is de selectie van gebruikers gebaseerd op de aanhang van de Nederlandse politieke partijen en hun politici. Hierbij is er vanuit gegaan dat de aanhang van de Nederlandse politiek veelal bestaat uit Nederlandse twitteraars, die op hun beurt weer Nederlandse tweets produceren. Op deze manier zijn er ook veel Nederlandse accounts gevonden, maar er kwamen ook nog veel accounts in voor die buitenlandse talige tweets posten. Voor deze reden is er per account gekeken in welke taal er getwitterd wordt. Als deze persoon dan regelmatig Nederlandse tweets twittert, dan wordt deze persoon geclassificeerd als een Nederlandse twitteraar en schaarde zich op deze manier bij de selectie van de vaste volgers. Bij het grotendeels posten van buitenlandse tweets wordt de gebruiker juist van de groep van vaste volgers verwijderd.

2. Zijn er al bestaande technieken op het gebied van topicdetectie d.m.v. clustering en zijn er al methodes voor het clusteren van tweets en reacties op forums, nieuwswebsites en blogs?

Met de literatuurstudie, die behandeld is in hoofdstuk 3, is de tweede deelvraag beantwoord. Hier zijn acht verschillende technieken besproken die rekening houden met het verwerken van een datastroom. De belangrijkste eigenschappen die het algoritme moet bezitten zijn:

- Algoritme kan met tekst (grote dimensies) omgaan.
- Er kunnen snel en incrementeel nieuwe datapunten verwerkt worden (efficiënt genoeg om elke binnenkomende tweet te verwerken).
- Compacte representatie

- Er wordt rekening gehouden met de tijd (nieuwe bestanden krijgen hogere prioriteit).
- Er is een duidelijke en snelle manier om uitschieters en ruis te identificeren.

De algoritmes Birch, Coweb en Stream zijn als eerst behandeld. Vanuit andere literatuur werd echter duidelijk, dat dit niet de beste algoritmes zijn en dat er betere te vinden zijn. Deze algoritmes vielen af omdat ze niet genoeg van de bovengenoemde eigenschappen bevatten. CluStream kwam al een stuk verder. Hier wordt er van te voren een offline fase gedraaid, zodat er in een online fase sneller punten verwerkt kunnen worden. De algoritmes die veruit als beste naar voren zijn gekomen, zijn de algoritmes Fractioneel clusteren en D-Stream. Aan de meeste bovengenoemde eigenschappen wordt bij deze algoritmes voldaan. Alleen werd in een paar artikelen vermeld dat deze algoritmes niet om zouden kunnen gaan met tekst en omdat dit essentieel is, vielen ook deze twee algoritmes af. Het volgende algoritme waar naar gekeken werd, is het 'burst-algoritme'. In tegenstelling tot de andere algoritmes is dit geen clusteralgoritme. Deze bepaalt alleen welke woorden als trend kunnen worden getypeerd en clustert de woorden vervolgens niet tot een duidelijke trend. Echter is het van groot belang dat de trends wel goed geïdentificeerd kunnen worden. Een algoritme dat gebaseerd is op het CluStream algoritme is OCTSM. Dit algoritme is speciaal gemaakt voor de verwerking van tekst. De representatie van het OCTSM-model neemt de context van de tekst in ogenschouw, waardoor er waarschijnlijk duidelijke trends ontstaan. De sentimenten van gedetecteerde trends worden ook goed vastgelegd aangezien door middel van het 'semantic smoothing model' de relatie van de trend met andere woorden kan worden vastgesteld.

3. Hoe wordt een eventueel gevonden clusteringtechniek toegepast?

In hoofdstuk 4 wordt antwoord gegeven op de derde deelvraag. Hier worden als eerst alle woorden uit de tweets gehaald die een belangrijke waarde kunnen hebben voor het identificeren van trends. Deze woorden worden ook gebruikt voor het maken van frases. Een frase is hierbij een combinatie van twee woorden dat een klein onderwerp representeert. Deze frases dragen bij aan het preciezer clusteren van de woorden uit de tweets. Deze frases zorgen namelijk dat er naar context van de tweets wordt gekeken. Het offline proces haalt informatie uit het begin van de datastroom en probeert hierdoor informatie te winnen om straks te gebruiken bij het online proces. Stel dat in het offline proces er een woord vaak in combinatie met een aantal frases is voorgekomen. Als deze frases zich later dan ook in een cluster bevinden dan kan er met grotere zekerheid gezegd worden dat dit woord meer kans heeft om echt bij dit onderwerp te horen. In het offline proces zijn aan de hand van 'expectation maximisation' de kansen op alle woord-frase combinaties vastgesteld.
Het online proces is de afhandeling van de datastoom in 'real time'. Als een tweet binnenkomt wordt hierbij als eerst gekeken of er clusters samengevoegd kunnen worden. Als dit het geval is dan wordt deze tweet een nieuw cluster, waardoor deze tweet eventueel een nieuwe trend kan gaan vormen. Als clusters niet samengevoegd kunnen worden dan wordt er gekeken of de binnengekomen tweet bij een van de al bestaande clusters past. Als dit het geval is dan wordt deze toegevoegd aan dit cluster en zo niet, dan wordt het cluster verwijderd waarvan de updatedatum het oudst is en vervangen door de nieuwe tweet. Snelle verwerking van de tweets is hierbij mogelijk door een efficiënte representatie van de clusterprofielen.

Het algoritme zoals beschreven in de originele literatuur [32] is niet helemaal aangehouden. De belangrijkste verandering is de manier waarop de clusters met elkaar vergeleken worden. Het vergelijken van de clusters ging bij de implementatie niet alleen erg langzaam, maar ook snel fout. Door het geringe aantal woorden die zich bevinden in de tweets, wordt er te veel waarde gehecht aan de woorden die zich wel in de clusters bevinden. Hierdoor vindt er al een snel een match plaats tussen twee clusters terwijl misschien slechts een woord overeenkomt. Om deze negatieve eigenschap te omzeilen wordt er bij deze vergelijking alleen gekeken naar de woorden die het belangrijkst zijn voor een cluster.

4. **Hoe wordt de gevonden clusteringtechniek geëvalueerd?**

In Hoofdstuk 5 is de vierde deelvraag beantwoord. Het bleek lastig een realistische testomgeving te creëren die een twitterstroom goed genoeg kon nabootsen om het OCTSM algoritme op te testen. Het is namelijk moeilijk om een bepaalde groep van tweets tot een trend te rekenen. Als het om een kleine dataset was gegaan dan zou er handmatig nog verschillende trends geselecteerd kunnen worden, maar in dit geval ging het om een dataset met 11.8 miljoen tweets. Er is daarom gekozen om geautomatiseerd een dataset samen te stellen. Er zijn namelijk twintig verschillende categorieën verzameld aan de hand van een hashtag dat zich in een tweet bevindt. De vragen die nu beantwoord kunnen worden zijn de volgende: kan het algoritme deze trends onderscheiden? En lukt dit nog steeds als de hashtags uit de tweets wordt verwijderd?

De verwachting was dat het algoritme met hashtag geen problemen zou hebben en dat bij de dataset zonder hashtag de onderwerpen er ook nog wel goed uit te halen zouden zijn, omdat het algoritme expliciet rekening houdt met de context van een tweet. Deze verwachtingen bleken niet helemaal gegrond. Van de dataset met hashtag die bestond uit een stroom van 1.500 tweets werd 79.3% van de tweets correct geclusterd en alle categorieën zijn hierbij gevonden. Bij een stroom van 4.600 tweets was dit al een stuk minder. Hier waren er maar 15 categorieën gevonden en diende de overige 5 als ruis voor de andere categorieën (‘false positieves‘). Dit resulteerde in een correcte classificatie van 58.3% van de tweets. Bij de dataset van 1.500 tweets was er slechts 0.9% van de tweets in een klasse ingedeeld die tot een andere trend behoorde (‘false positives’). Bij de dataset van 4.600 was dit al 19.5%.

Het algoritme op de datasets zonder hashtags deden het echter niet goed. Hier werd bij de datastroom van 1.500 tweets slechts 13.5% van de tweets correct geclusterd en was 27.0% van de tweets in een incorrecte cluster gekomen. Er werden hierbij ongeveer 7 trends ontdekt. Bij de dataset van 4.600 tweets zijn de resultaten van gelijke stemming. Hier werd 14.3% van de tweets werd correct geclusterd, 27.2% in een fout cluster geclusterd en er werden ongeveer 9 trends geïdentificeerd. Er blijkt gewoonweg te weinig informatie in een tweet te zitten, als de hashtag verwijderd wordt, om ze bij elkaar te clusteren.

Uiteindelijk kan gesteld worden dat het OCTSM-algoritme alleen toegepast kan worden als er een beperkte dataset beschikbaar is. De slechte resultaten op de datasets zonder hashtag geven aan dat te veel ruis niet bevorderlijk is voor dit algoritme. Hoe er verder gehandeld moet worden zal in de aanbeveling beschreven worden.

5. **Hoe worden de gevonden onderwerpen gerepresenteerd?**

De vijfde deelvraag is niet beantwoord, omdat er niet genoeg tijd was om deze uit te werken.

7 Aanbevelingen

Het feit dat er wel een goede clustering plaatsvindt als de hashtags zich wel in de tweets bevinden, kan betekenen dat als de echte stroom wordt beperkt, dit algoritme wel toepasbaar zou worden. Het 'burst-algoritme' uit paragraaf 3.3.8 zou bijvoorbeeld voor deze beperking van de dataset kunnen dienen. Dit algoritme ontdekt alleen de woorden die ten opzichte van het verleden opeens veel voorkomen. Als deze woorden eenmaal gedetecteerd zijn dan zou OCTSM per trend kunnen vaststellen welke woorden belangrijk zijn voor deze trend. Het nadeel van het 'burst-algoritme' is namelijk het feit dat niet meteen gezien kan worden of bepaalde woorden bij elkaar horen, of dat ze twee aparte trends zijn. Een mogelijke oplossing die hieruit voortkomt, is dat het OCTSM-algortime alleen de subcategorieën binnen een bepaalde hashtag ontdekt.

In paragraaf 4.1 is zelf een procedure ontwikkeld die de relevantere woorden uit de tweets filtert. Dit was nodig omdat het algoritme dat hier in [32] voor gebruikt is, Xtract [34], niet voor de Nederlandse taal beschikbaar is. Deze procedure is een onderzoek op zich waard. Nu is de Brill tagger gebruikt om dit op te lossen, dit is niet noodzakelijkerwijs de beste oplossing. Er zijn namelijk een hoop woorden die nog niet herkend worden. Ook zou er nog meer gelet kunnen worden op woorden met spelfouten erin. Nu wordt er alleen naar de stam gekeken van het woord, waardoor hier al enigszins rekening mee wordt gehouden. Google gaat echter nog iets verder. Zij hebben een applicatie [W20] ontwikkeld die spelfouten uit woorden kan halen. Hoe goed dit product werkt is nog niet duidelijk maar zou misschien in deze situatie een aanvulling zijn.

Bij het trainen van het 'topic signature translation model' aan de hand van 'expectation maximisation' op 20.000 tweets viel het op dat er erg veel werkgeheugen nodig was. Bij ongeveer 30.000 tweets was er niet genoeg geheugen over om al deze tweets te trainen. Dit is afhankelijk van de computer die er gebruikt wordt en van de wijze van implementatie, maar als er meer tweets moeten worden getraind bij de huidige implementatie dan zal dit in stapjes moeten gebeuren.

Als er in een later stadium dit algoritme op een echte stroom van tweets moet worden toegepast, dan is het 'topic signature translation model' op een gegeven moment niet meer up to date. Bij Twitter worden er zowat elke dag weer nieuwe hashtags verzonnen. Van de tweets die deze nieuwe hashtags bevatten is bij het maken van het model nog helemaal geen informatie beschikbaar. Dit geldt niet alleen voor hashtags, maar ook voor 'normale' woorden. Onderwerpen die de ene week erg veel besproken worden zouden de volgende week helemaal niet meer aan bod kunnen komen, maar vervangen zijn door weer totaal nieuwe onderwerpen.

Een oplossing voor dit probleem zou het dagelijks (of misschien wekelijks) updaten van de modellen gemaakt in de offline fase. Hierdoor ontstaat er op een gegeven moment een erg groot model. Het voordeel hiervan is dat er over veel onderwerpen informatie beschikbaar komt. Een nadeel is dat het raadplegen van dit model steeds langzamer gaat. Een kans opzoeken in een kleinere database gaat namelijk sneller dan dat deze vol zit met data van het afgelopen jaar. Artikel [32] heeft hier zelf al een oplossing voor gegeven: alleen de waardes van $p(w_i|t_k)$ die het meest recent zijn opgevraagd worden in het werkgeheugen opgeslagen. Mocht een kans op een gegeven moment niet meer wordt opgevraagd dan wordt deze kans uit het werkgeheugen gehaald en zou bij de volgende opvraag uit de database moeten worden gehaald. Voor een preciezere beschrijving moet het artikel zelf geraadpleegd worden.

De oplossing die is gegeven voor het te snel samenvoegen van clusters in paragraaf 4.3.2, is een mogelijke oplossing. Het zou ook nog kunnen dat door het aanpassen van de formule getoond in figuur 5, er wel een efficiëntere samenvoeging kan ontstaan. Misschien zou er een manier bedacht worden waardoor minder groter waardes per woord worden gegeven als een cluster maar uit een paar woorden bestaat.

8 Literatuurlijst

Wetenschappelijke artikelen

[1] Barbará D. Requirements for Clustering Data Streams, Vol. 3, issue 2, pages 23-27, januari 2002.

[2] Chernoff, H. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations. Annals of Mathematical Statistics, Vol. 23, pages 493-509, 1952.

[3] Barbará D., and Chen, P. Tracking Clusters in Evolving Data Sets. Proceedings of FLAIRS'2001, Special Track on Knowledge Discovery and Data Mining , Key West, mei 2001.

[4] Zhang T., Ramakrishnan R., and Livny M. "BIRCH: A Efficient Data Clustering Method for Very Large Databases. Proceedings of the ACM SIGMOD Conference on Management of Data, Montreal, Canada, 1996.

[5] Gluck M.A., and Corter J.E. Information, uncertainty, and the utility of categories. Proceedings of the Seventh Annual Conference of the Cognitive Science Society, lrvine, CA,1985.

[6] J. B. MacQueen: "Some Methods for classification and Analysis of Multivariate Observations, Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability", Berkeley, University of California Press, 1:281-297, 1967.

[7] Hartigan, J. and Wong, M. , “Algorithm AS136: A k-means clustering algorithm.” Applied Statistics, 28, 100-108, 1979.

[8] Barbará D., and Chen, P. Using the Fractal Dimension to Cluster Datasets. Proceedings of the ACM-SIGKDD International Conference on Knowledge and Data Mining, Boston, augustus 2000.

[9] Schroeder M. Fractal, Chaos, Power Laws: Minutes from an Infinite Paradise. W.H. Freeman and Company, 1991.

[10] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[11] Fisher D.H. Iterative Optimization and Simplification of Hierarchical Clusterings. Journal of AI Research, Vol. 4, pages 147-180, 1996.

[12] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In Proc. VLDB, pages 81–92, 2003.

[13] A. Jain, R. Dubes. Algorithms for Clustering Data, Prentice Hall, New Jersey, 1998.

[14] Y. Chen, L. Tu. Density-Based Clustering for Real-Time Stream Data, 2007.

[15] F. Cao, M. Ester, W. Qian, A. Zhou. Density-Based Clustering over an Evolving Data Stream with Noise, 2006.

[16] J. Kleinberg. “Bursty and Hierarchical Structure in Streams”. Proc. 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, 2002.

[17] P. Fekkes. A Web Application for Visualization of the Blogspave and Trend Detection within the Blogspace, Master’s thesis, oktober 2005.

[18] L. Dalton, V. Ballarin and M. Brun. Clustering Algorithms: On Learning, Validation, Performance, and Applications to Genomics, 2009.

[19] M. Halkidi, Y. Batistakis, M. Vazirgiannis. On Clustering Validation Techniques, 2001.

[20] N. Grira, M. Crucianu, N Boujeaa. Unsupervised and Semi-supervised Clustering: a Brief Survey, augustus 2005.
[21] Marggie D. González Toledo. A comparison in cluster validation techniques, december 2005.
[22] Jess Jiangsheng Shen. Using Cluster Analysis, Cluster Validation, and Consensus Clustering to Identify Subtypes of Pervasive Developmental Disorders, november 2007.
[23] L. Kaufman and P. Rousseeuw. Finding groups in data: An introduction to cluster analysis. John Wiley & Sons, 1990.
[24] Roth, V.; Braun, M.; Lange, T.; Buhmann, J.M. Stability-based model order selection in clustering with applications to gene expression data. In Lecture Notes in Computer Science; Dorronsoro, J.R., Ed.; Springer: Heidelberg, 2002, Vol. 2415, pp. 607-612.
[25] Yeung, K.Y.; Haynor, D.R.; Ruzzo, W.L. Validating clustering for gene expression data. Bioinformatics, 2001, 17, 309-318.
[26] Cover, T.; Hart, P. Nearest neighbor pattern classification. IEEE Trans. Inf. Theory, 1967, 13, 21-27.
[27] Roth, V.; Braun, M.; Lange, T.; Buhmann, J.M. Stability-based model order selection in clustering with applications to gene expression data. In Lecture Notes in Computer Science; Dorronsoro, J.R., Ed.; Springer: Heidelberg, 2002, Vol. 2415, pp. 607-612.
[28] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: Theory and practice. 2003.
[29] L. O'Callaghan et al. Streaming-Data Algorithms For High-Quality Clustering. ICDE Conference, 2002.
[30] J, Oberst Efficient Data Clustering and How to Groom Fast-Growing Trees, 03-03-2009.
[31] M. Khalilian, N. Mustapha. Data Stream Clustering: Challenges and Issues, 17-03-2010.
[32] Y. B. Liu, J. R. Cai, J. Yin, “Clustering text data streams”, Journal of Computer Science and Technology, vol. 23(1), Jan. 2008, pp. 112-128.
[33] Xiaodan Zhang, Xiaohua Zhou, Xiaohua Hu. Semantic smoothing for model-based document clustering. In Proc. ICDM06, Hong Kong, december 18-22, pp.1193-1198.
[34] Smadja, F. Retrieving collocations from text: Xtract. Computational Linguistics, 1993, 19(1), pp. 143--177.
[35] S. C. Johnson, “Hierarchical Clustering Schemes" Psychometrika, 2: 241-254,1967.
[36] Watanabe, O. Simple Sampling Techniques for Discovery Science. IEICE Transactions on Inf. & Syst., Vol. E83-D, No. 1, January, 2000.
[37] Zhou X, Hu X, Zhang X, Lin X, Song I Y. Context-sensitive semantic smoothing for the language modeling approach to genomic IR. In Proc. ACM SIGIR, Seattle, Washington, August 6-11, 2006, pp.170-177.
[38] Zhong, S. and Ghosh, J. Generative model-based document clustering: a comparative study. Knowledge and Information Systems, 8(3): 374-384, 2005.
[39] Sheets genaamd ‘Naive Bayes for Text Classification (2)’ van Steve Renals: een professor op de universiteit van Edinburgh.

Webadressen

[W1] Chun Wei, Clustering Data Streams, afdeling Computer & Information Technology
http://www.cis.uab.edu/kddm/seminar_home/Chun_ClusteringDataStreams.ppt (19-10-2010)
[W2] Jiong Yang, Clustering II, Case Western Reserve University

http://vorlon.case.edu/~jiong/cs435_2008/jy_clustering2.ppt (19-10-2010)
[W3] http://techcrunch.com/2010/09/14/twitter-seeing-90-million-tweets-per-day/ (20-01-2011)
[W4] http://blog.sysomos.com/2010/01/14/exploring-the-use-of-twitter-around-the-world/ (26-10-2010)
[W5] http://twitter.com/PvdA/tweede-kamerleden (26-10-2010)
[W6] www.vox-pop.nl (14-12-2010)
[W7] http://static.businessinsider.com/image/4c07a5357f8b9aae79740a00-590/people-use-twitter-to-link-chat-and-say-what-they-are-doing-at-the-moment.jpg (14-12-2010)
[W8] http://www.alexa.com/topsites (14-12-2010)
[W9] http://nl.wikipedia.org/wiki/Social_media (14-12-2010)
[W10] http://nl.wikipedia.org/wiki/Twitter (14-01-2011)
[W11] http://www.techradar.com/news/internet/10-news-stories-that-broke-on-twitter-first-719532 (14-01-2011)
[W12] http://bits.blogs.nytimes.com/2010/06/18/sports-fans-break-records-on-twitter/ (14-01-2011)
[W13] http://mashable.com/2010/06/25/tps-record/ (14-01-2011)
[W14] http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html (07-02-2011)
[W15] http://mathworld.wolfram.com/Entropy.html (07-02-2011)
[W16] http://dragon.ischool.drexel.edu/xtract.asp (07-02-2011)
[W17] http://en.wikipedia.org/wiki/Brill_tagger (07-02-2011)
[W18] http://cosmion.net/jeroen/software/brill_pos/ (07-02-2011)
[W19] http://search.cpan.org/~chm/PDL-2.4.7/Basic/PDL.pm (19-02-1011)
[W20] http://code.google.com/p/google-refine/ (20-02-2011)

Bijlage I

Deze bijlage geeft een SQL-query waar naar verwezen wordt in hoofdstuk 2.

Met de onderstaande query wordt bepaald welke twitteraars er gevolgd worden:

```
SELECT distinct followed AS followers, follower_count, tweet_count FROM
follows, author where a_id = followed and (preferred_language = "-" or
preferred_language = "nl") and timestamp = (SELECT max(timestamp) FROM
follows) and ((tweet_count>50 and follower_count>10) or (tweet_count<=50 and
follower_count>1000))
UNION
SELECT distinct follower AS followers, follower_count, tweet_count FROM
follows, author where a_id = follower and (preferred_language = "-" or
preferred_language = "nl") and timestamp = (SELECT max(timestamp) FROM
follows) and ((tweet_count>50 and follower_count>10) or (tweet_count<=50 and
follower_count>1000))
order by follower_count desc, tweet_count desc
```

Bijlage II

Deze bijlage geeft de resultaten weer van de testset van 4.600 tweets met een hashtag. Dezelfde resultaten voor een testset van 1.500 tweets is gegeven in paragraaf 5.3.1.

Categorie	Input	Aantal correct geclusterd	Verdeeld over aantal clusters	%goed geclusterd
#ns	292	271	1	92.8
#pvv	296	267	1	90.2
#tmobile	295	266	1	90.2
#penoza	68	60	1	88.2
#blijdorp	114	97	1	85.1
#ipad	295	248	1	84.1
#moerdijk	296	248	1	83.8
#nkafstanden	91	74	1	81.3
#feyenoord	283	227	1	80.2
#f1	296	229	1	77.4
#sinterklaas	293	225	1	76.8
#gtst	273	192	1	70.3
#museumnacht	118	76	1	64.4
#obama	55	32	1	58.2
#ohohcherso	297	161	2	54.2
#mulisch	233	4	1	1.7
#tvoh	296	4	1	1.4
#tnt	293	0	0	0
#vuurwerk	297	0	0	0
#lagerhuis	120	0	0	0
Totaal	**4601**	**2681**		**58.3**

Tabel a. Een overzicht van de resultaten van de correct geclusterde tweets bij de testset met hashtag.

Categorie	Input	% in verkeerde cluster	Aantal false positives
#tnt	293	86.0	252
#vuurwerk	297	83.5	248
#tvoh	296	73.6	218
#mulisch	233	73.8	172
#lagerhuis	120	78.3	94
#gtst	273	2.9	8
#ohohcherso	297	1.7	5
#sinterklaas	293	1.4	4
#obama	55	5.5	3
#tmobile	295	1.0	3
#moerdijk	296	1.0	3
#f1	296	0.7	2
#feyenoord	283	0.7	2
#blijdorp	114	1.8	2
#ipad	295	0.7	2
#penoza	68	1.5	1
#ns	292	0.3	1
#nkafstanden	91	0	0
#pvv	296	0	0
#museumnacht	118	0	0
Totaal		**22.2**	**1020**

Tabel b. Een overzicht van de resultaten van de incorrect geclusterde tweets bij een testset met hashtag.

Aantal tweets in een cluster zonder dominante klasse	Aantal clusters
733	1

Tabel c. Een overzicht van het aantal clusters dat wel meer dan 3 tweets heeft maar geen dominante categorie bevat, bij een testset met hashtag.

Bijlage III

Deze bijlage geeft de resultaten weer van de testset van 4.600 tweets zonder een hashtag. Dezelfde resultaten voor een testset van 1.500 tweets is gegeven in paragraaf 5.3.2.

Categorie	Input	Aantal correct geclusterd	Verdeeld over aantal clusters	%goed geclusterd
#ns	292	131	1	44.9
#pvv	296	124	2	41.9
#tnt	293	122	1	41.6
#tmobile	295	81	4	27.5
#f1	296	78	1	26.4
#feyenoord	283	44	1	15.5
#moerdijk	296	39	1	13.2
#obama	55	6	1	10.9
#ohohcherso	297	11	2	3.7
#lagerhuis	120	4	1	3.3
#tvoh	296	8	1	2.7
#gtst	273	4	1	1.5
#ipad	295	4	1	1.4
#sinterklaas	293	0	0	0
#mulisch	233	0	0	0
#nkafstanden	91	0	0	0
#blijdorp	114	0	0	0
#penoza	68	0	0	0
#vuurwerk	297	0	0	0
#museumnacht	118	0	0	0
Totaal	**4601**	**656**		**14.3**

Tabel a. Een overzicht van de resultaten van de correct geclusterde tweets bij de testset zonder hashtag.

Categorie	Input	% in verkeerde cluster	Aantal false positives
#ipad	295	48.8	144
#feyenoord	283	43.5	123
#moerdijk	296	36.5	108
#sinterklaas	293	33.8	99
#mulisch	233	42.5	99
#tmobile	295	27.5	81
#gtst	273	28.9	79
#ohohcherso	297	20.2	60
#ns	292	18.5	54
#tvoh	296	17.6	52
#blijdorp	114	45.6	52
#pvv	296	16.9	50
#f1	296	13.5	40
#vuurwerk	297	13.1	39
#penoza	68	54.4	37
#lagerhuis	120	29.2	35
#tnt	293	10.9	32
#nkafstanden	91	29.7	27
#museumnacht	118	19.5	23
#obama	55	29.1	16
Totaal		**27.2**	**1250**

Tabel b. Een overzicht van de resultaten van de incorrect geclusterde tweets bij een testset zonder hashtag.

Aantal tweets in een cluster zonder dominante klasse	Aantal clusters
902	9

Tabel c. Een overzicht van het aantal clusters dat wel meer dan 3 tweets heeft maar geen dominante categorie bevat, bij een testset zonder hashtag.