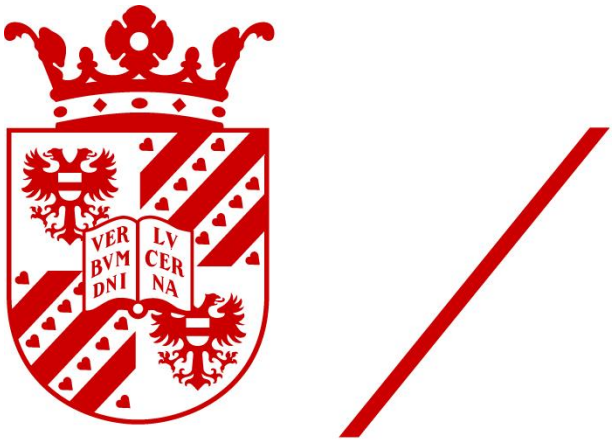
P2000 locatiedata als classifier voor tweets

Gericht zoeken naar verbanden en randzaken rondom een event.



rijksuniversiteit groningen

Olivier Louwaars

S2814714

24 mei 2015

P2000 locatiedata als classifier voor tweets

Gericht zoeken naar verbanden en randzaken rondom een event.

Groningen, mei 2015

Auteur	Olivier Louwaars
Studentnummer	2814714
Afstudeerscriptie in het kader van	Informatiekunde Faculteit der Letteren Rijksuniversiteit Groningen
Begeleiders onderwijsinstelling	mw. M. Nissim Faculteit der Letteren Rijksuniversiteit Groningen
	dhr. J.Bos Faculteit der Letteren Rijksuniversiteit Groningen

Voorwoord

Hoewel het als Pre-Masterstudent een aardige omslag is van het HBO naar het WO, komen de twee samen bij het maken van een scriptie aan het eind van de opleiding. Dankzij een hoge collegedichtheid, is het gelukt om binnen een jaar op te klimmen naar het niveau van een Bachelor student Informatiekunde.

Inhoud

Samenvatting

Als aanleiding voor het schrijven van deze scriptie heeft het framework *Twitcident* (Abel, Hauff et al. 2012) model gestaan. Met behulp van deze software kan een gebruiker zien hoe er in sociale media gereageerd wordt op een gebeurtenis. Door middel van bepaalde kernwoorden en zoektermen kunnen de berichten bij elkaar worden gezocht om zo een ruimer beeld te scheppen dan slechts een noodmelding. Opvallend hierbij is echter dat Abel et al. geen gebruik maakten van bekende geografische data van beide informatiestromen. In dit onderzoek is daarom ingegaan op precies die informatie, om berichten die van dezelfde locatie komen, sneller aan elkaar te linken. Als data voor het onderzoek, zijn 450.000 tweets en 100.000 *P2000* meldingen gebruikt. *P2000* is het nationale waarschuwingssysteem van Nederlandse hulpdiensten, waarmee ze onderling communiceren nadat een melding bij de meldkamer binnen is gekomen. Van deze meldingen was van ongeveer de helft een locatie te bepalen, waarna de juiste tweets bij elke meldingen konden worden gezocht op basis van GPS coördinaten. Door deze coördinaten volgens het principe van *Geohash* (Beatty 2005) om te rekenen naar een code voor een bepaalde plaats, kunnen eenvoudig de omliggende codes gevonden worden. De lengte van de *Geohash* bepaalt de straal van cirkel waarin gezocht wordt. Door een hash van zeven tekens te berekenen, ontstaat er een straal van ~100 meter rond het punt waar de noodoproep over gaat. Alle tweets in deze cirkel werden vervolgens bij de oproep geplaatst. Op deze manier ontstond er een lijst van 39.000 meldingen waarbij een of meerdere tweets in de buurt werden gepubliceerd. Van deze 39.000 waren er rond de 3.800 tweets op dezelfde dag als de oproep geplaatst. Door vervolgens alleen de tweets op te nemen die in de twee uur voor of de drie uur na een gebeurtenis zijn geplaatst (~1500), had ongeveer een kwart van de meldingen een bijbehorende, relevante tweet. Dit betekent dat een *baseline* algoritme dat altijd zegt dat een tweet géén betrekking heeft op een melding, in 75% van de gevallen goed zit. De te bouwen classifier moet dus in ieder geval significant beter presteren dan deze basis om een toevoeging te zijn. Door te leren van 600 geannoteerde tweets, presteerde een *Naive Bayes classifier* inderdaad significant beter, met een accuracy van 92% (p=0,000). Ook de *F-score* (resp. 0,86 en 0,93) en de *precision* (0,75 en 0,92) stegen significant terwijl de *recall* daalde (1,0 en 0,93). Het is dus zeker aan te raden om niet alleen op geografische informatie, maar ook op inhoud van tweets te selecteren bij het koppelen aan meldingen.

1. Inleiding

Elke dag worden er vanuit Nederland meer dan 5 miljoen tweets verstuurd over de meest uiteenlopende onderwerpen. Soms alleen interessant voor de schrijver, soms voor een klein groepje lezers, maar soms ook voor heel Nederland, zoals bij een ramp. In deze scriptie zal worden gekeken naar het effect van een voorselectie op basis van *P2000* informatie, die de parameters levert voor het zoeken van geografisch aanverwante tweets. *P2000* is het communicatiesysteem dat door de verschillende (Nederlandse) hulpdiensten wordt gebruikt om meldingen door te geven en hulpverleners aan te sturen. Door op basis van deze berichten een voorselectie van tweets te maken, kunnen tweets die er tekstueel wellicht niets mee te maken hebben, maar wel uit de buurt komen, er toch bijgevoegd worden. Ook is het interessant om te zien of er voorafgaand of juist na afloop van de melding er interessante tweets opduiken die context kunnen schetsen. Aanleiding voor dit onderzoek is het framework *Twitcident* uit 2012, waarin onderzoekers wel naar semantiek kijken, maar niet naar de geo-locatie van tweets en noodmeldingen. De onderzoeksvraag hierbij is: “Is het toepassen van een geografische voorselectie als filter voldoende voor het vinden van relevante tweets in een klein geografisch gebied binnen Nederland? En zo niet, wat moet er dan nog meer gebeuren? ”

Centraal in dit onderzoek staat de nog altijd uitdijende stroom tweets. Dit immer groeiend aantal wordt steeds lastiger te doorzoeken, en dus zou het veel tijd schelen als er slechts een beperkt deel van alle gepubliceerde tweets hoeft te worden doorzocht. De keuze om hiervoor gebruik te maken van het *P2000* communicatiesysteem voor hulpdiensten is tweeledig: Voor de onderlinge communicatie is een standaard opgesteld waardoor elke melding een vast stramien heeft met constante parameters zoals locatievermelding. Ook wordt er, indien van toepassing, op dezelfde manier gespecificeerd om wat voor incident het precies gaat, wat ook weer *keywords* oplevert om later tweets mee te kunnen zoeken. Het tweede argument voor de keuze van deze data heeft een maatschappelijke reden: Iedereen wil tegenwoordig zo goed mogelijk op de hoogte zijn van wat er om hem heen gebeurt, en als deze scriptie daar verbetering in kan brengen levert dat altijd direct winst op.

Er is reeds onderzoek gedaan naar dit onderwerp, en zoals gezegd is het artikel *Twitcident: Fighting Fire with Information from Social Web Streams* (Abel, et al., 2012) de directe aanleiding geweest. In dit artikel wordt beschreven hoe beschikbare *P2000* data wordt omgezet in een soort opsporingsbericht, waarna de best bijpassende tweets hiervoor worden verzameld. Opvallend hierbij is dat de locatie van zowel de tweets als van de melding hierbij geen belangrijke rol lijkt te spelen. Dat is dan ook direct het grote verschil met, of de aanvulling op die dit onderzoek vormt. Verder is, vrijwel gelijktijdig, het onderzoek *TEDAS: A Twitter-based Event Detection and Analysis System* (Li, et al., 2012) gepubliceerd. Ook dit onderzoek richt zich op het detecteren en groeperen van events, maar maakt veel gebruik van de ingebouwde geo-locaties die tweets

steeds vaker meekrijgen. Samen kunnen deze artikelen dan ook de basis leggen voor het nieuwe product en onderzoek.

Voor de te realiseren applicatie is het van belang dat een goede inventarisatie van de bestaande software wordt gemaakt, om dubbel werk te voorkomen en goed werkende koppelingen te kunnen maken. In het hoofdstuk Methode zal verder aan de orde komen op welke manier de software precies toegepast zal worden. De resultaten die geboekt worden zullen vervolgens ook gerapporteerd worden, waarna er geconcludeerd wordt in hoeverre deze resultaten daadwerkelijk beter of slechter zijn dan van toeval verwacht mag worden.

2. Methode

Hoewel de te gebruiken methode voor dit onderzoek uniek is, staat vooraf al vast dat alle te programmeren onderdelen met *Python* zouden worden gerealiseerd. Zowel door de bestaande kennis en uitvoerige documentatie van deze programmeertaal, als het gebruiksgemak voor taal gerelateerde opdrachten en functionaliteiten.

2.1. Data verzamelen

Voor dit onderzoek zijn twee verschillende databases gebruikt; de openbare P2000 database zoals deze op diverse websites te vinden is en de Twitter database van de Rijksuniversiteit Groningen met daarin alle Nederlandse tweets van de afgelopen vijf jaar.

2.1.1. P2000 data

Deze eerste stap van het onderzoek was direct een erg grote, namelijk het schrapen van veranderende data op dezelfde URL van de gekozen P2000 website[1]. Deze site geeft de meldingen per 15 weer, met zo min mogelijk metadata er omheen. Aangezien de website asynchroon loopt blijft URL ongeacht de inhoud hetzelfde, waardoor een *webcrawler* niet door kan naar eerdere pagina's door de URL aan te passen. Het drukken op de 'vorige' knop door de gebruiker moest dus gesimuleerd worden door een script. Uiteindelijk bleek de knop door te verwijzen naar een PHP-script dat wel het paginanummer in de URL heeft, zodat daar de spider uit de *Scrapy*[2] module heen gestuurd kon worden. *Scrapy* heeft als voordeel dat het gebruikers van tevoren laat definiëren welke informatie of HTML-tags hij wel en niet wil downloaden, zodat het strippen van webpagina's veel sneller kan verlopen dan als de volledige pagina gedownload zou moeten worden. Zo heeft *Scrapy* op 10.250 pagina's alleen de meldingen voor de volledige maand april gedownload waar passende tweets bij gezocht moesten worden.
Voor deze koppeling was het van belang dat iedere melding een GPS-coördinaat kreeg toegewezen in plaats van de plaats en straatnaam die er nu in stond. Door te kijken of er een straatnaam met bijbehorende stad in de melding stond die ook voor komt in een los bestand met alle plaats- en straatnamen van Nederland, kon er voor ~50.000 meldingen een adres met eventueel huisnummer gevonden worden. Al deze adressen werden door de module *Geopy*[3]gehaald, die er een lengte- en breedtegraad voor terug gaf.

2.1.2. Twitterdata

Voor het downloaden van de tweets was een veel simpeler script afdoende, er hoefde voor iedere tweet uit april alleen gekeken te worden of deze een GPS-coördinaat had, waarna hij aan

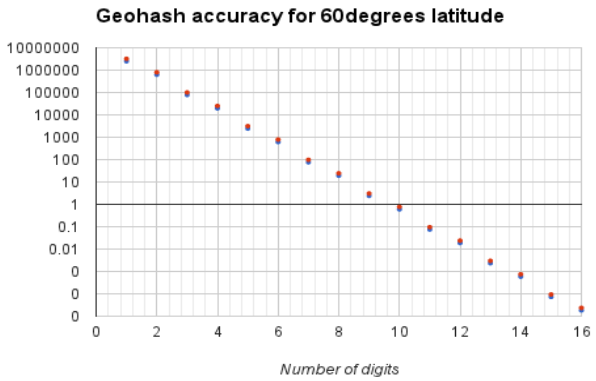
---

[1] http://p2000-online.net
[2] http://scrapy.org
[3] https://geopy.readthedocs.org

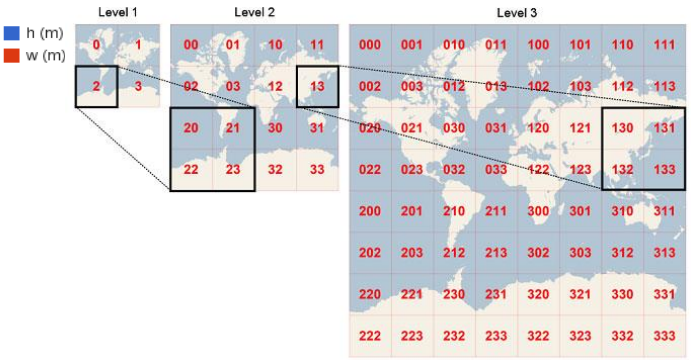
de dataset toegevoegd kon worden.
Met behulp van het programma Tweet2Tab dat voor studenten beschikbaar is, kunnen de tweets volgens een vaste volgorde met de gewenste kolommen worden opgeslagen. Voor dit onderzoek waren de tijd, datum, locatie, gebruikersnaam en tekst van belang.

2.2. Data koppelen

Aangezien het coördinaat dat elke tweet en melding nu had erg precies is, zou de kans dat er precies op die plek een overeenkomst is verwaarloosbaar zijn. Om die reden is er gekozen om alle coördinaten te versleutelen volgens het *geohash* (Beatty, 2005) principe. Een *geohash* is een reeks letters die een coördinaat representeert, waarbij de lengte van de hash gelijk is aan de precisie van het coördinaat. Zoals in Figuur 1 te zien is, heeft een hash van lengte 7 een precisie van ~100 meter. Dit is voldoende voor dit onderzoek, aangezien iemand op deze afstand nog wel iets mee krijgt van een incident. Een ander voordeel van een *geohash*, is dat van elk punt altijd alle 'buren' bekend zijn. De eerste zes letters zijn gelijk, en de zevende geeft de positie weer ten opzichte van het eerste punt. Als een punt tegen een grensvlak aan ligt, en dus een buurman heeft die met andere letters begint (Figuur 2), houdt de functie die alle buren berekent hier rekening mee en zal dit punt ondanks een andere code toch meenemen. Op deze manier worden er acht vierkanten verkregen rondom een punt, waarbij er in dit geval wordt gezocht naar een tweet met dezelfde hash als één van deze negen vierkanten (Figuur 3). In de praktijk betekent dit dat er voor elk van 450.000 tweets moet worden gekeken of deze voor komt in een van de negen mogelijkheden van 50.000 meldingen, waardoor dit script het meest tijdrovend was van het hele onderzoek. Aangezien de indeling van geografische coördinaten traditioneel gebeurt met (breedtegraad, lengtegraad), was daar ook vanuit gegaan bij dit onderzoek. Na veel puzzelen en wachten op de berekeningen, bleek echter dat Twitter haar coördinaten andersom gebruikt waardoor er telkens slechts één toevallige match opdook. Door dit om te draaien ontstond er een lijst van 39.000 meldingen met bijbehorende tweets.



Figuur 2: De precisie van een geohash met x aantal tekens. Bron: http://goo.gl/W9dVMI



Figuur 2: Versimpelde uitleg van het opbouwen van een geohash Bron: https://goo.gl/yxRNBT

```
def lochasher(lines):
      hashDict = {}
      for line in lines:
              if type(line[1]) != tuple:
                      hashed = Geohash.encode(round(float(line[0][0]),6)
                      ,round(float(line[0][1]),6),7)
                      hashDict.setdefault(hashed,[]).append(line)
              else:
                      hashed = Geohash.encode(round(float(line[0][1]),6)
                      ,round(float(line[0][0]),6),7)
                      hashedneighbors = geohash.neighbors(hashed)
                      hashedneighbors.append(hashed)
                      hashDict.setdefault((' '.join(hashedneighbors)
                      ,line[1]),[]).append(line)

      return hashDict

def locmatcher(tweets,alerts):
      matchDict={}
      pbar=ProgressBar()
      x=0
      for hashedtw in pbar(tweets):
              for hashedal in alerts:
                      if  hashedtw in hashedal[0].split():
                      matchDict.setdefault(hashedal[0]
                      ,[]).append(hashedtw)
                      x+=1

      print(x)
      return matchDict
```

Figuur 3: De gebruikte code voor geohashes en het koppelen van meldingen en tweets.

2.3. Data analyseren

Nadat de tweets aan de bijbehorende locatie gekoppeld waren, kon er een eerste analyse van de koppels gemaakt worden. Iedere melding had gemiddeld drie tweets, met uitschieters tot negen. Aangezien deze tweets van de hele maand april konden zijn is er als eerst gefilterd op tweets van dezelfde dag als de melding. De ongeveer 4.000 overblijvers hadden gemiddeld nog maar iets minder dan twee tweets, waarop is besloten om voor 700 oproepen te annoteren of een tweet hier wel of niet relevant voor was. Na annotatie bleek echter dat slechts $^1/_8$ van de tweets nu relevant was, waardoor een systeem dat iedere tweet als niet relevant zou classificeren het in 87,5% goed zou hebben. Door de relevante tweets te analyseren werd duidelijk dat het overgrote deel daarvan binnen twee uur vóór en drie uur ná een incident werden gepost. Door alleen deze tweets door de selectie te laten werd $^1/_4$ van de tweets relevant, waardoor het percentage goed voorspelde antwoorden daalde. De *baseline classifier* bleef echter voorspellen dat geen enkele tweet relevant is, waardoor een systeem dat relevante tweets wél weet te herkennen getraind moest worden.

3. Resultaten

Per melding was gemiddeld iets meer dan één tweet binnen het gestelde tijdskader geplaatst, waarop het aantal te annoteren tweets op 600 is gesteld. Voor deze tweets is handmatig aangegeven of ze relevant(1) of niet relevant(0) waren voor de betreffende noodoproep, waarna er met behulp van het *Naive Bayes* (Manning, et al., 2008) algoritme door het systeem kon worden geleerd waar een (ir)relevante tweet aan te herkennen is. *Naive Bayes* gaat uit van een *bag of words* principe, wat betekent dat alle woorden van alle tweets bij elkaar worden verzameld waarbij de woordvolgorde, woordpositie of combinatie van woorden er niet toe doen. Het doel is om het systeem te leren documenten te classificeren. Dit is een vorm van *supervised machine learning*. Hierbij zijn de mogelijke klassen op voorhand gegeven en is van de documenten in de trainingsdata door de handmatige annotatie bekend tot welke klasse zij behoren. Het categoriseren wordt gedaan door het voorspellen van de meest waarschijnlijke klasse op basis van de woordfrequentie. Het systeem leert met behulp van de training-set van woorden, welke woorden bij welke klasse het meeste voorkomen. In een resterende test-set van documenten kan vervolgens worden getest hoe goed de *Naive Bayes classifier* werkt. In *Python* kan middels de NLTK[4] (Natural Language Toolkit) module op deze manier geclassificeerd worden. Door van 80% van de data te leren, en het geleerde op 20% ongeziene data toe te passen, was het model in staat om 92% van de tweets aan de juiste categorie toe te wijzen. Een op het oog significant verschil dat ook wordt ondersteund door een gepaarde t-test (p=0,000).
Aangezien het basisalgoritme iedere tweet kwalificeerde als niet-relevant, heeft het alle niet-relevante tweets juist (recall is 100%). Dat het vervolgens een groter aantal tweets in deze categorie heeft dan er in zouden moeten zitten heeft alleen invloed op de precision (75%). Het gewogen gemiddelde van deze twee levert een F-score op van 0,86, terwijl het nieuwe model 0,92 voor de relevante en zelfs 0,94 voor de irrelevante tweets scoort. Hoewel dit verschil een stuk kleiner is, betekent dit dus wel een forse toename in precision (van 0,75 naar respectievelijk 0,95 en 0,90). Daarentegen daalt de recall (van 1,0 naar 0,93), maar met een kleiner percentage.

Tabel 1: Scores van het nieuwe model.

	PRECISION	RECALL	F-SCORE
RELEVANT	0.958333	0.884615	0.92
IRRELEVANT	0.909091	0.967742	0.9375

Naast deze totaalscores is het ook interessant om te zien op welke punten het goed ging en op welke minder goed. Tabel 2 laat zien welke keuzes gemaakt zijn door het getrainde algoritme, waarbij de r voor referentie staan (wat het zou moeten zijn) en de v voor voorspeld (wat het systeem denkt dat het is). Opvallend hier aan is het relatief grote percentage dat eigenlijk relevant is, maar niet zo beoordeeld wordt. Hieruit blijkt dat er bepaalde signaalwoorden die een mens

[4] http://nltk.org

wel aan een gebeurtenis zou koppelen, niet als zodanig door de machine herkend worden. Dit kan verholpen worden door extra nadruk op bepaalde woorden te leggen, die de doorslag geven naar positief of negatief als ze aangetroffen worden. Aangezien het aantal tweets waarop uiteindelijk het getrainde algoritme losgelaten wordt slechts beperkt is (~300, dus 75 relevant) is het gevaarlijk om bepaalde woorden aan te wijzen die nu relatief vaak voorkomen, omdat dit misschien toeval is en ze over het geheel gezien juist niet belangrijk zijn. Voor dit principe, *overfitting* genaamd, wordt gewaarschuwd in het artikel *Domain Adaptation: Overfitting and Small Sample Statistics* (Kakade, et al., 2011). Om deze reden is er ook gekozen om zeer spaarzaam voorkomende woorden te negeren, en alleen de 3000 vaakst voorkomende woorden als trainingsdata te gebruiken. Als een woord immers slechts eenmaal voorkomt koppelt de classifier dit direct als belangrijk woord aan een bepaalde keuze.

Tabel 2: Verdeling van de scores

	IRRELEVANT (V)	**RELEVANT**
IRRELEVANT (R)	52.6%	1.8%
RELEVANT (R)	5.3%	40.4%

Ook is het mogelijk en aan te raden om (te experimenteren met) het aantal woorden waarmee getraind wordt te beperken. Woorden die in zowel relevante als irrelevante tweets veel voor komen, zoals lidwoorden en voornaamwoorden, kunnen eruit gefilterd worden door ze op te nemen in een *stoplijst*. Deze woorden worden vervolgens uitgesloten van het automatisch leren. In dit experiment voegde een stoplijst echter niets toe en gaat wederom het argument van *overfitting* op. De meest informatieve woorden voor de testset staan in Tabel 3, waarbij de verhouding van de kans dat het woord in de ene klasse voorkomt ten opzichte van de kans dat het in de andere klasse voorkomt wordt genoemd. Een ‘@’ komt bijvoorbeeld 7,8 keer vaker voor bij een niet relevante tweet dan bij een relevante, terwijl ‘Brandweer’ 6,9 keer vaker relevant dan niet relevant is. Door de kleine hoeveelheid (en daardoor wellicht weinig variabele) testdata, kunnen ook woorden die minder logisch lijken en een mens niet zouden helpen juist heel hoog scoren voor de machine.

Tabel 3: Informatieve woorden en de kans van voorkomen in een klasse

WOORD	**WAARDEN**	**VERHOUDING**
GAAT	rel : irrel	13.8 : 1
1	rel : irrel	11.3 : 1
WEER	irrel : rel	8.2 : 1
@	irrel : rel	7.8: 1
!	irrel : rel	7.5 : 1
NIET	irrel : rel	7.0 : 1
BRANDWEER	rel : irrel	6.9 : 1
/	rel : irrel	6.9 : 1
MAAR	irrel : rel	6.4 : 1
(	rel : irrel	6.0 : 1

4. Conclusie

Op basis van de gevonden resultaten, kan er geconcludeerd worden dat het niet afdoende is als een tweet uit hetzelfde tijdsframe en van dezelfde locatie komt als een noodmelding via P2000. Hoewel een aanzienlijk deel van de tweets die hieraan voldoen weliswaar relevant zijn voor de melding, zijn er teveel andere tweets die niet voldoen. Zo kan er niet zonder meer vanuit gegaan worden dat een tweet die in deze selectie wordt aangetroffen relevant is. Wel is aangetoond dat het op tijd en locatie voorselecteren van tweets een bijzonder goede basis is voor het verdere classificeren. Het percentage van 25% relevante tweets bij een speling van vijf uur rond een gebeurtenis werd immers al 12,5% bij een hele dag mét een overeenkomstige locatie, laat staan als er gekeken zou worden naar tweets op dezelfde dag door een hele stad of zelfs heel Nederland.

Hoewel er is begonnen met een ruime hoeveelheid tweets en P2000 meldingen, bleef daar door steeds strengere selecties niet erg veel van over. Belangrijke verfijningen zoals *feature selection* (het benadrukken van bepaalde belangrijke woorden) en het gebruik van een stoplijst waren niet mogelijk door een te gering aantal tweets in de uiteindelijke trainings- en testdata.

Bibliografie

Abel, Fabian, et al. 2012. Twitcident: Fighting Fire with Information from Social Web Streams. *WWW 2012, Proceedings of the 21st World Wide Web Conference 2012.* Lyon, France : Companion Volume, 2012, pp. 305-308.

Beatty, Bryan Kendall (Sammamish, WA, US). 2005. *Compact text encoding of latitude/longitude coordinates. 20050023524* United States, 2005.

Kakade, Sham, Foster, Dean en Salakhutdinov, Ruslan. 2011. Domain Adaptation: Overfitting and Small Sample Statistics. *Cornell University Library.* [Online] 2011. [Citaat van: 16 Mei 2015.] http://arxiv.org/abs/1105.0857v1.

Li, Rui, et al. 2012. TEDAS: A Twitter-based Event Detection and Analysis System. *2012 IEEE 28th International Conference on Data Engineering (ICDE).* Washington, DC : IEEE, 2012, pp. 1273 - 1276.

Manning, Chrisopher D., Raghavan, Prabhakar en Schütze, Hinrich. 2008. 13 Text Classification and Naive Bayes. *Introduction to Information Retrieval.* Cambridge : Cambridge University Press, 2008, pp. 206,207, 234-242.

Bijlagen

Scripts?