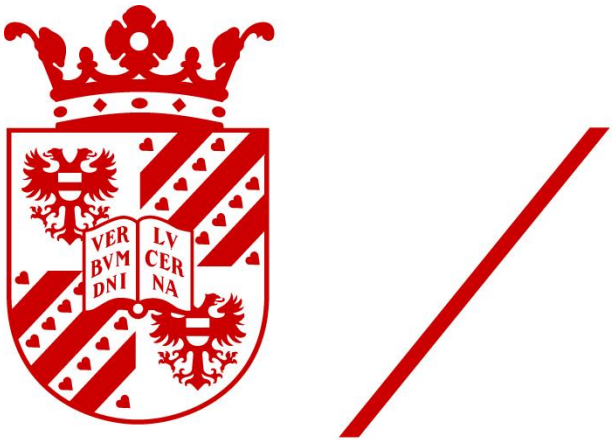
P2000 locatiedata als classifier voor tweets

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. Voor de bijbehorende programmacode en scripts verwijs ik u door naar https://github.com/obipls/scriptie

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 (Twittermania, 2012). 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 geografische voorselectie (dus tweets uit een bepaald gebied) op basis van de locatie van een noodmelding voor hulpdiensten. Deze selectie zou dan bestaan uit een klein en hopelijk relevant deel van de dagelijkse ongeordende en onoverzichtelijke hoeveelheid tweets. Door vervolgens alleen naar deze groep en de centraal staande noodmelding te kijken en met elkaar te vergelijken, kan een computer vrij eenvoudig leren om relevante en niet relevante tweets te leren herkennen, en dit vervolgens zelf toepassen op nieuwe meldingen en tweets. Het proces van leren en relevantie toekennen is eenvoudiger en sneller op een kleine hoeveelheid data met daarin zowel een aanzienlijk deel relevante als irrelevante voorbeelden om beide te kunnen herkennen. Als er bijvoorbeeld 50 tweets relevant zijn bij een incident, is het veel lastiger om deze 50 uit de verzameling van 5 miljoen te halen dan uit een verzameling van 1000 tweets. Natuurlijk zullen er ook irrelevante tweets op dezelfde locatie verschijnen, maar door alleen in de buurt te selecteren zal het allergrootste deel zeker afvallen.

Als bron van de meldingen voor hulpdiensten zullen berichten worden gebruikt die zijn verstuurd via het *P2000* netwerk. Dit is het communicatiesysteem dat door de verschillende (Nederlandse) hulpdiensten wordt gebruikt om melding te maken van incidenten en hulpverleners aan te sturen. De communicatie via het P2000 systeem is niet versleuteld en kan door iedereen worden opgevangen en gepubliceerd op bijvoorbeeld een website. 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, toch als relevant bestempeld worden. Ook kan het interessant zijn om te zien of er voorafgaand of juist na afloop van een melding er tweets opduiken die context kunnen schetsen. Naast het openbare karakter is een tweede argument om deze bron te gebruiken de structuur van de data. Voor de onderlinge communicatie is een standaard opgesteld waardoor elke melding een vast stramien heeft met constante parameters zoals locatievermelding en, indien van toepassing, wat er aan de hand is. Dit laatste ontbreek echter vaak, en dan ziet een gebruiker alleen een melding met de strekking: “Brandweer met grote spoed naar adres X”. Het combineren met tweets kan dit verbeteren, doordat buurtbewoners vertellen wat ze zien. Hoewel gebruikers van hulpdiensten zelf natuurlijk wel precies weten wat er aan de hand is kunnen ook zij hun voordeel doen met informatie uit nabije tweets.

In een studie uit 2012 naar de combinatie van noodmeldingen en tweets waar het framework *Twitcident*[1] uit voortkomt, *Twitcident: Fighting Fire with Information from Social Web Streams*

[1] http://wis.ewi.tudelft.nl/twitcident

(Abel, et al., 2012), is gekeken naar de inhoud en betekenis van tweets die betrekking hebben op een incident. Uit dit onderzoek kan vooral veel informatie gehaald worden met betrekking tot de uiteindelijke implementatie van dit project, aangezien het eenvoudiger is om te werken met bestaande data. Als de resultaten naar wens zijn kan het systeem online gezet worden, waar het te maken krijgt met binnenstromende tweets en noodmeldingen in plaats van vaste datasets. Abel et al. beschrijven de manieren waarop zij de juiste berichten afvangen, en door welke services en API's ze vervolgens de juiste tweets verkrijgen.
De werkwijze en het doel van Abel et al. zijn vergelijkbaar met dit onderzoek, alleen is de geo-locatie van tweets is destijds niet meegenomen als criteria, waarschijnlijk omdat slechts 0,77% van de tweets toen een locatie meekreeg van de gebruiker (Semiocast, 2012). Hoewel het percentage tweets met geo-locatie nog altijd laag is, rond de 3% (Pool, 2015), is het wel flink gestegen in de loop der jaren en is het absolute aantal tweets met locatie zeer groot.

Verder is, vrijwel gelijktijdig, het onderzoek *TEDAS: A Twitter-based Event Detection and Analysis System* (Li, et al., 2012) gepubliceerd. Ook hier richt men zich op het detecteren en groeperen van events, maar wordt wel veel gebruik gemaakt van de ingebouwde geo-locaties van tweets (indien aanwezig). Li et al. zien twitteraars als journalisten die overal verspreid zijn en voortdurend nieuwsberichten van 140 tekens maken. De bedoeling is om de eerste die over een gebeurtenis (CDE , *Crime and Disaster related Event*) schrijft op te sporen, en van hieruit gerelateerde tweets te vinden en zo geografische verbanden te kunnen leggen en deze inzichtelijk te maken aan gebruikers. Het systeem is bedoeld om achteraf verbanden te kunnen zien, bijvoorbeeld buurten in een stad waar veel branden zijn, en dus niet echt om incidenten live te volgen. Naast de geo-locatie die Twitter automatisch aan tweets geeft, hebben ze ook tips voor als deze informatie ontbreekt. Zo kan er gekeken worden naar een patroon in tweets van een gebruiker, en als er bij vergelijkbare tweets wel een locatie staat kan deze dan gebruikt worden. Ook het netwerk van een twitteraar is van belang, zo zal iemand die hij volgt en vaak in zijn tweets noemt waarschijnlijk in de buurt wonen, waardoor de locatie van de twitteraar 'voorspeld' kan worden. Deze informatie is wederom pas voor de implementatie interessant, aangezien er in de statische twitterdata niet naar volgers en hun locatie gezocht kan worden.

Hoewel deze twee onderzoeken op het eerste oog dus vooral relevant lijken voor een later stadium, zijn ze samen juist de aanleiding geweest om dit onderzoek op te starten. Ze beantwoorden de vraag of het überhaupt zin heeft om een koppeling tussen tweets en noodmeldingen te maken, en wat een dergelijke koppeling toe zou voegen.

In deze scriptie zal het volgende onderzocht worden: "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? "

3

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 tussen de verschillende onderdelen te kunnen maken. Verderop in deze scriptie, in het hoofdstuk Methode, zal 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 zoals het opsplitsen van zinnen.

2.1. Data verzamelen

Voor dit onderzoek zijn twee bronnen nodig: de P2000 data met alle noodmeldingen van Nederlandse hulpdiensten, en de Twitter database van de Rijksuniversiteit Groningen met daarin alle tweets in het Nederlands van de afgelopen vijf jaar. Na een aantal kleine experimentjes en een pilotstudy op ~20.000 P2000 meldingen bleek begin mei dat er aan de ene kant erg veel data uit beide bronnen nodig was om een fatsoenlijk aantal ´matches' tussen tweets en meldingen te kunnen maken, terwijl aan de andere kant het verzamelen en verwerken van data niet bijzonder snel ging. Om die reden is er besloten tot een middenweg, namelijk de data van één maand. Deze arbitraire keuze bleek genoeg data op te leveren, terwijl de hoeveelheid nog wel te overzien bleef.

2.1.1. P2000 data

Hoewel P2000 data ongecodeerd wordt verstuurd, is er nog wel speciale apparatuur nodig om mee te kunnen luisteren. Gelukkig wordt dit al gedaan en worden de berichten vervolgens in een database opgeslagen. Helaas is de toegang tot deze database niet gratis, en kan er via de sites die hier hun data uithalen niet direct in de database gezocht worden. Live data is eenvoudiger te krijgen, door een abonnement op een RSS feed die automatisch updates doorgeeft, maar voor oude berichten was een script nodig dat ze automatisch kon downloaden.
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[2]. Deze site geeft de meldingen per 15 weer, met zo min mogelijk metadata er omheen. Aangezien de website asynchroon loopt blijft de 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*[3] 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.

---

[2] http://p2000-online.net
[3] http://scrapy.org

Zo heeft *Scrapy* op 10.250 pagina's alleen de meldingen voor de volledige maand april gedownload waar passende tweets bij gezocht moesten worden.
De enige overeenkomst tussen de tweets en de meldingen zou de locatie zijn, en dus was de volgende stap dat iedere melding een GPS-coördinaat kreeg toegewezen in plaats van de plaats en straatnaam die er nu (soms) in stond. Een coördinaat is universeel en uniek, wat het vergelijken en samenvoegen mogelijk maakt. Door voor iedere melding te kijken of er een straatnaam met bijbehorende stad in stond die ook voor kwam 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*[4]gehaald, die er een lengte- en breedtegraad voor terug gaf. Geopy helpt gebruikers om adressen en coördinaten bij elkaar te zoeken, en kan via *Open Street Maps* bij bijna alle adressen op aarde. Helaas mag er via de gratis service slechts een beperkt aantal adressen per etmaal opgezocht worden (~35.000), waardoor het een aantal dagen duurde om alle coördinaten te verkrijgen en op te slaan.

2.1.2. Twitterdata

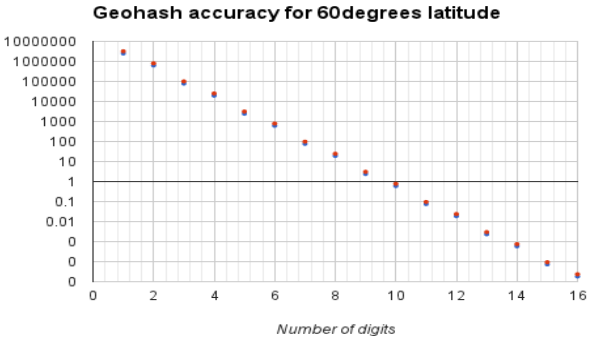
Voor het downloaden van de tweets was een veel simpeler script afdoende, er hoefde voor iedere tweet uit april alleen gekeken te worden of deze een GPS-coördinaat had, waarna hij aan de dataset toegevoegd kon worden. De faculteit Letteren van de Rijksuniversiteit Groningen verzamelt al jaren Nederlandse tweets, waarna deze beschikbaar worden gesteld aan studenten en onderzoekers om mee te werken. Naast de data worden er ook een aantal instrumenten aangeboden waarmee de data geselecteerd of bewerkt kan worden. Iedere tweet wordt opgeslagen in een groot aantal kolommen, die niet altijd allemaal nodig zijn. Met het script *Tweet2Tab* kunnen de gewenst kolommen geselecteerd en opgeslagen worden. Voor dit onderzoek waren de tijd, datum, locatie, gebruikersnaam en tekst van belang.

2.2. Data koppelen

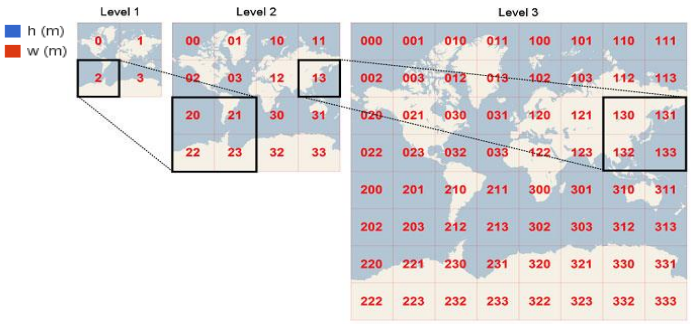
Aangezien het coördinaat dat elke tweet en melding nu had erg precies is, zou de kans dat er precies op die plek een overeenkomst is verwaarloosbaar zijn. Om die reden zijn alle coördinaten versleuteld volgens het *geohash* (Beatty, 2005) principe. Een *geohash* is een reeks letters die een coördinaat representeert, waarbij de lengte van de hash gelijk is aan de precisie van het coordinaat. 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.

[4] https://geopy.readthedocs.org

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 500.000 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 1: 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

```
1 def lochasher(lines):
2         hashDict = {}
3         for line in lines:
4                 if type(line[1]) != tuple:
5                         hashed = Geohash.encode(round(float(line[0][0]),6)
6                         ,round(float(line[0][1]),6),7)
7                         hashDict.setdefault(hashed,[]).append(line)
8                 else:
9                         hashed = Geohash.encode(round(float(line[0][1]),6)
10                        ,round(float(line[0][0]),6),7)
11                        hashedneighbors = geohash.neighbors(hashed)
12                        hashedneighbors.append(hashed)
13                        hashDict.setdefault((' '.join(hashedneighbors)
14                        ,line[1]),[]).append(line)
15        return hashDict
16
17 def locmatcher(tweets,alerts):
18        matchDict={}
19        pbar=ProgressBar()
20        x=0
21        for hashedtw in pbar(tweets):
22                for hashedal in alerts:
23                        if  hashedtw in hashedal[0].split():
24                        matchDict.setdefault(hashedal[0]
25                        ,[]).append(hashedtw)
26                        x+=1
27        print(x)
28        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 meldingen (~1.400 tweets) 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 (~1800) 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.

Van de 25% relevante tweets bleek een groot deel gebaseerd op P2000 berichten, en verwezen deze daar soms ook naar (Tabel 1). Vaak gaf de tweet echter nog wel een aanvulling op de melding, waardoor aan een schijnbaar nietszeggende melding toch wat context kan worden toegevoegd. Ook veel kranten, omroepen en de hulpdiensten zelf zijn actief op twitter en vertellen na afloop of ter plaatse wat er aan de hand is. 'Normale' twitteraars die toevallig ergens passeerden waren zeldzaam in de data, en kwamen vaker voor naarmate het incident groter werd.

Tabel 1: Een aantal gematchte meldingen en tweets

A1 AMBU OG902 Botersloot 3011HE Rotterdam rt bon 38075
#ROTTERDAM #RRM | Bedrijf / instelling 1 Botersloot rt daadwerkelijke overval . politie doet onderzoek | http://t.co/GcjLac8A5r #p2000
PRIO 1 Ongeval Letsel Buiten : : Oude Raadhuisstraat : Didam (OVDB Alarm , rv) (GRIP 1)
@rachidfinge attractie is nu leeg !! Brandweer heeft de inzittende gered
PRIO 1 Buitenbrand : : Rolklaver : Kampen 041092 042095 042086 (BR: middel) (AGS Alarm) (GRIP 1)
Brandweer Kampen : geen asbest in woonwijk gevonden na brand volkstuinencomplex http://t.co/2uwutbUByA #112overijssel #rtvoost
Prio 2 assistentie kleine ontsmetting (inmelden: BNH-INCI-30) (-) (OPS: zeer grote brand) (GRIP: 2) Noordergeestkerk Noordergeeststraat 9 Heiloo Brand Bijeenkomst 4333
Grote brand vannacht bij leegstaande kerk in #Heiloo . Alleen de toren staat nog overeind . @RTVNH http://t.co/K6DiS3McoP
A1 5011LN X : HAG Tilburg Beethovenlaan X Beethovenlaan X HAG Tilburg Beethovenlaan Tilburg 32848
Tilburg - Verwarde Tilburger slaat voorruit auto kapot : De politie heeft dinsdag 28 april 2015 omstreeks 16.35... http://t.co/bYz1twXaZ2

3. Resultaten

Per melding bleek uiteindelijk gemiddeld iets meer dan één tweet binnen het gestelde tijdskader geplaatst, waarop het aantal te annoteren tweets op 600 is gesteld zodat opnieuw $^1/_3$ van het totaal geannoteerd werd. 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 *probabilities*, dit betekent dat het voortdurend kansen berekent voor elk woord dat er in wordt gestopt: de kans dat het woord in een relevante tweet staat, de kans dat het in een irrelevante tweet staat, de kans dat een tweet relevant is en de kans dat het woord überhaupt voorkomt. Door deze kansen vervolgens weer door elkaar te delen of met elkaar te vermenigvuldigen, komt het algoritme tot een conclusie: wel of niet relevant. In Figuur 4 is in woorden te zien welke kansen het algoritme gebruikt. Posterior is de kans dat het woord in een relevante tweet staat. Prior is het aantal keer dat het woord eerder al voorkwam in een relevante tweet gedeeld door het totaal aantal tweets. Likelihood is de kans dat het woord voorkomt in alle tweets en Evidence is de kans dat een tweet positief is.

$$Posterior = \frac{\Pr ior \times Likelihood}{Evidence}$$

Vergelijking 1: Naive Bayes in woorden

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 trainingsdata 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[5] (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 cate-

[5] http://nltk.org

gorie 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 (Tabel 1). 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 2: 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 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 3: 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 4: 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?