

BACHELOR INFORMATICA

UvA  UNIVERSITEIT VAN AMSTERDAM

Grafische geografisch-gebaseerde selectie van nieuws

Thomas van Ophem

17 juni 2015

Supervisor(s): Raphael 'kena' Poss (UvA)

Signed:

Samenvatting

Nieuws is normaal gesproken gerepresenteerd rondom een bepaald onderwerp, de lezer haalt de locatie uit de tekst. Zoekmachines zijn onderwerp of keyword gebaseerd. Er is op dit moment een beperkte mogelijkheid om nieuws te zoeken aan de hand van een geografische locatie of een gebied. Dit terwijl een applicatie waarbij het mogelijk is om nieuws te zoeken aan de hand van een locatie of gebied bij kan dragen aan het begrip van nieuws. In deze scriptie bespreken we de ontwerp keuzes die gemaakt zijn voor deze applicatie, vervolgens wordt er gekeken naar de implementatie en worden er twee experimenten uitgevoerd. Ten eerste een experiment om te bepalen hoeveel steden er nodig zijn om een gebied te kunnen representeren, en om te kijken hoe dit aantal af hangt van de grootte van het gebied. Het tweede experiment kijkt naar de toegevoegde waarde van de applicatie. Uit deze experimenten blijkt dat de grootte van het gebied (bijna) geen invloed heeft op het aantal steden dat nodig is om dit gebied te representeren en dat een applicatie waarbij nieuws gezocht kan worden aan de hand van een gebied van toegevoegde waarde is.

Inhoudsopgave

1	Inleiding	5
1.1	Context	5
1.2	Bestaande middelen	5
1.3	Probleem stelling	5
1.4	Strategie en onderzoeksvragen	6
1.5	Organisatie	6
2	Gerelateerd onderzoek	7
2.1	STEWARD: Architecture of a Spatio-Textual Search Engine [2]	7
2.2	NewsStand: A New View on News [1]	8
2.3	Reading News with Maps by Exploiting Spatial Synonyms [3]	8
3	Ontwerp	9
3.1	Web applicatie of standalone/desktop applicatie?	9
3.1.1	Web applicatie	9
3.1.2	Standalone/desktop applicatie	9
3.1.3	Web applicatie vs. standalone/desktop applicatie	9
3.2	Server	10
3.3	Python, PHP of Ruby on Rails	10
3.3.1	PHP	10
3.3.2	Ruby	10
3.3.3	Python	11
3.3.4	CherryPy	12
3.3.5	Cheetah	13
3.3.6	JSON	14
3.3.7	AJAX	14
3.3.8	JQuery	14
3.4	Database	15
3.5	Geografische informatie	15
3.6	Nieuwsbronnen	16
3.6.1	Google News	16
3.6.2	BING News	16
3.6.3	Overig	16
3.7	Zoeken naar nieuws	16
3.7.1	Client side of server side?	17
3.8	Ideaal ontwerp	17
3.9	Samenvatting	17
4	Implementatie	19
4.1	Server en database	19
4.1.1	Server	19
4.1.2	Database	20
4.1.3	geo_data.py	21

4.2	Welke steden liggen in het geselecteerde gebied?	21
4.3	Gebiedsrepresentatie	25
4.3.1	Algoritme 1 - gebiedsrepresentatie	25
4.3.2	Algoritme 2 - gebiedsrepresentatie	28
4.3.3	Andere opties	28
4.4	User Interface	29
4.5	Samenvatting	29
5	Experimenten	31
5.1	Gebiedsrepresentatie	31
5.1.1	Opzet	31
5.1.2	Resultaten	32
5.2	Toegevoegde waarde	33
5.2.1	Opzet	33
5.2.2	Resultaten	34
6	Discussie en toekomstig onderzoek	39
6.1	Discussie	39
6.2	Toekomstig onderzoek	40
6.2.1	Steden met onbekende populatie	40
6.2.2	Dubbele steden	40
6.2.3	Lokaal nieuws - voorstel	40
7	Conclusie	41
A	Installatie	45
A.1	Systeem vereisten	45
A.2	Toevoegen extensies aan SQLite	45
A.3	Server installatie	45
A.4	Server starten	45
B	Resultaten onderzoek gebiedsrepresentatie	47
C	Resultaten onderzoek toegevoegde waarde	49

Inleiding

1.1 Context

Nieuws is normaal gesproken gerepresenteerd rondom een bepaald onderwerp, de lezer haalt de locatie uit de tekst. Zoekmachines zijn onderwerp of keyword gebaseerd, er is op dit moment geen mogelijkheid om nieuws te zoeken aan de hand van een geografische locatie of een gebied.

1.2 Bestaande middelen

Op dit moment kan er naar nieuws gezocht worden via nieuws websites zoals bijvoorbeeld Reuters¹, Associated Press², NU³. Daarnaast is het mogelijk om nieuws te zoeken via zoekmachines zoals Google News⁴ en Bing News⁵.

Aan *The University of Maryland* wordt er bij het *computer science department* al 30 jaar onderzoek gedaan naar de zogenaamde "*spatial browsers*". Dit zijn browsers/zoekmachines waarmee de geografische informatie uit de resultaten inzichtelijker gemaakt wordt voor de gebruiker, maar waarmee ook gezocht kan worden aan de hand van geografische informatie. Het meeste recente systeem wat door het *computer science department* van *The University of Maryland* ontwikkeld is is NewsStand, meer hier over in hoofdstuk 2.

1.3 Probleem stelling

Wie op dit moment naar nieuws wil zoeken doet dit aan de hand van zoektermen gericht op een onderwerp of enkele locatie. Voor het nieuws over een brand in Amsterdam kan gezocht worden met bijvoorbeeld de zoekterm "brand Amsterdam". Dit is voor relatief eenvoudige zoektermen, waar er alleen gekeken wordt naar een stad geen probleem. Maar stel dat we willen weten wat er op dit moment gaande is in Noord Holland. Welke zoektermen moeten dan gebruikt worden?

Het probleem wordt nog groter als we kijken naar grotere (internationale) gebieden, stel we willen weten wat er gaande is rondom de Zwarte Zee. Om hier al het nieuws voor te vinden moet gezocht worden naar het nieuws in alle omliggende landen. Dit resulteert in een groot aantal zoektermen, met als gevolg dat we veel tijd kwijt zijn om een duidelijk beeld te krijgen van wat er in het betreffende gebied gaande is, of dat er helemaal niet gezocht wordt. Gevolg hiervan is dat mensen, burgers, een beperkt/beperkter begrip hebben van het nieuws. Hierdoor

¹<http://www.reuters.com/>

²<http://www.ap.org/>

³<http://www.nu.nl/>

⁴<https://news.google.nl/>

⁵<https://www.bing.com/?scope=news>

krijgen burgers geen of geen goede informatie over wat er gaande is in de wereld, worden zij zich niet bewust van politieke processen met als risico dat zij niet de juiste beslissingen nemen, en belanghebbenden een (mogelijk verkeerde) invloed uit kunnen oefenen.

1.4 Strategie en onderzoeksvragen

Een mogelijke oplossing hiervoor is een applicatie waarmee het mogelijk is om op gebied te zoeken, bijvoorbeeld door een gebied te selecteren op een kaart waar vervolgens het nieuws bijgezoekt wordt. Hiermee krijgen burgers een duidelijker beeld van politieke processen, en zaken die hun leven kunnen beïnvloeden. Daarnaast wordt het mogelijk om de kern en de oorzaken van problemen in de wereld beter te begrijpen.

Dit onderzoek willen we kijken wat de toegevoegde waarde voor de gebruiker van een dergelijke applicatie is. De hoofdvraag in deze scriptie is:

Wat is de toegevoegde waarde van het zoeken van nieuwsberichten aan de hand van een locatie?

Omdat dit een brede vraag is zullen we een aantal deelvragen formuleren om naar het antwoord op deze vraag toe te kunnen werken.

Wat is het beste ontwerp voor een dergelijke applicatie?

Kunnen we een gebied representeren door een beperkt aantal steden? En zo ja, hoe?

Hoeveel steden zijn er nodig om een gebied te representeren?

Hoe kunnen we aan de hand van deze steden naar nieuws zoeken?

Welke nieuwsbronnen kunnen we gebruiken?

Hoe implementeren we een dergelijke applicatie?

1.5 Organisatie

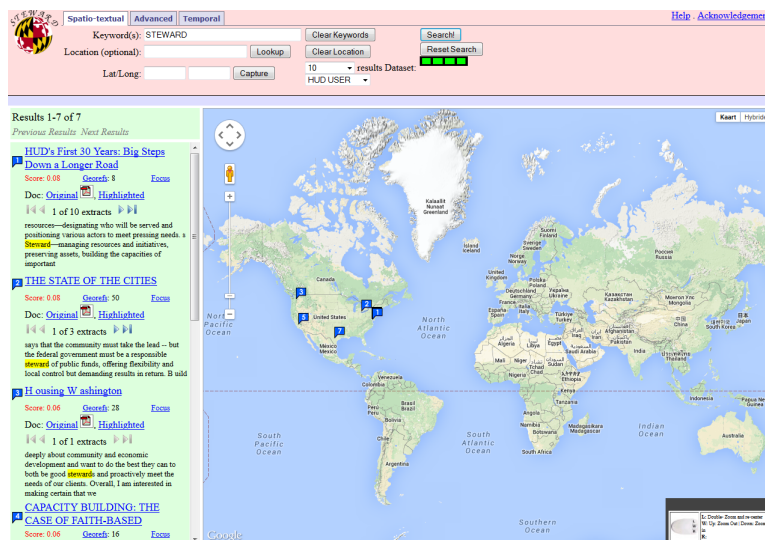
In hoofdstuk 2 zullen we eerst kijken naar gerelateerd onderzoek, zodat we ons een goed beeld kunnen vormen wat er op dit moment mogelijk is. Hoofdstukken 3 en 4 bespreken het ontwerp en de implementatie van de applicatie, met deze applicatie zullen we vervolgens gaan onderzoeken wat de toegevoegde waarde van een dergelijke tool is. De experimenten die hiervoor gedaan zijn zullen besproken worden in hoofdstuk 5.

Gerelateerd onderzoek

In november 2008 is het artikel, *NewsStand: A New View on News* [1], verschenen. Dit onderzoek gaat verder op een artikel gepubliceerd in november 2007, *STEWARD: Architecture of a Spatio-Textual Search Engine* [2]. Eind 2014 is er van de zelfde groep onderzoekers, verbonden aan *The University of Maryland*, nog een artikel verschenen: *Reading News with maps by Exploiting Spatial Synonyms* [3]. In deze onderzoeken wordt gekeken of het mogelijk is om gebruik te maken van de geografische informatie die (vaak) in nieuwsberichten verwerkt is, en hoe dit dan gebruikt zou kunnen worden. Er wordt hier echter nog niet gekeken naar de mogelijkheid om op een kaart een gebied te selecteren en daar het nieuws bij te zoeken.

2.1 STEWARD: Architecture of a Spatio-Textual Search Engine [2]

STEWARD ("*Spatio-Textual Extraction on the Web Aiding Retrieval of Documents*")¹ is een systeem voor het extraheren, opvragen en visualiseren van referenties naar geografische locaties in tekst. In dit artikel wordt besproken hoe het systeem tot stand is gekomen en wat de mogelijkheden van het systeem zijn.



Figuur 2.1: STEWARD User Interface

Het STEWARD systeem kan gebruikt worden voor een aantal verschillende toepassingen. Bij-

¹<http://steward.umiaccs.umd.edu/>

voorbeeld als zoekmachine voor het "hidden web"², waar een normale zoekmachine, welke gebruik maakt van een pagerank algoritme³, niet zal werken door het gebrek aan links naar de documenten. Ook kan er gezocht worden op nieuws, dit gaat nog wel aan de hand van een onderwerp (met eventueel een locatie). De resultaten worden vervolgens op een kaart weer gegeven zodat de gebruiker gemakkelijk kan zien wat de geografische locatie van het artikel is. Daarnaast kan het systeem gebruikt worden als een monitoring systeem voor ziektes, en het verzamelen van touristische, historische en recreatieve informatie over een stad of gebied.

2.2 NewsStand: A New View on News [1]

Nieuws artikelen bevatten heel veel (implicite) geografische informatie, die niet altijd duidelijk is voor de lezer. Door dit inzichtelijk te maken wordt het begrip van nieuws vergroot. NewsStand doet dit door RSS feeds [22] in de gaten te houden. Voor elk artikel wordt de geografische inhoud geëxtraheerd door gebruik te maken van een geotagger, de artikelen worden vervolgens geclusterd. Door in te zoomen op de kaart kunnen gebruikers nieuwsberichten vinden, afhankelijk van hoe ver een gebruiker ingezoomd is worden verschillende nieuwsberichten getoond.

In het artikel wordt besproken waarom een dergelijk systeem nuttig is, en hoe dit systeem is opgebouwd.

2.3 Reading News with Maps by Exploiting Spatial Synonyms [3]

Dit artikel uit oktober 2014 is het vervolg op *NewsStand: A New View on News* uit november 2008. De architectuur van NewsStand⁴ wordt ook in dit artikel weer besproken, net als het proces van het geotaggen en de problemen waar hier rekening mee gehouden moeten worden.

In de toekomst willen zij deze tool uitbreiden zodat er naast tekst ook gezocht kan worden op foto's, videos en audio. Ook willen ze andere soorten nieuwsbronnen gaan gebruiken zoals Twitter.



Figuur 2.2: NewsStand User Interface

²http://en.wikipedia.org/wiki/Deep_Web

³<http://en.wikipedia.org/wiki/PageRank>

⁴<http://newsstand.umiaccs.umd.edu/web/>

Ontwerp

In dit hoofdstuk zullen de gemaakte ontwerpkeuzes besproken worden. Ook zullen er twee voorstellen gedaan worden voor de implementatie van een applicatie om nieuws te zoeken aan de hand van een geografische locatie.

3.1 Web applicatie of standalone/desktop applicatie?

Een dergelijke applicatie kan geïmplementeerd worden als web applicatie of standalone applicatie. In deze sectie zullen we beide opties bespreken en een keuze tussen deze twee maken.

3.1.1 Web applicatie

Een web applicatie is een applicatie die voor gebruikers beschikbaar is via een webserver. Deze server is vaak bereikbaar via het internet maar het is ook mogelijk om dit via intranet te doen. De gebruiker kan via een client programma, zoals een webbrowser, gebruik maken van een dergelijke applicatie.

3.1.2 Standalone/desktop applicatie

Een standalone of desktop applicatie is een programma waarbij vooraf bepaald is welke taken uitgevoerd kunnen worden. Vaak zijn dergelijke programma's beschikbaar vanaf een lokale schijf in de computer van een gebruiker en hebben deze programma's in principe geen internet/netwerk nodig om te kunnen functioneren.

3.1.3 Web applicatie vs. standalone/desktop applicatie

Om te kunnen beslissen of we beter een web applicatie of een standalone/desktop applicatie kunnen gebruiken moeten we eerst weten welke aspecten van belang zijn bij een dergelijke applicatie. Dat zijn:

- Beschikbaar voor iedereen
- Portabiliteit, niet afhankelijk zijn van een besturingssysteem
- Privacy
- Toegang tot recent nieuws

Nu we weten welke aspecten belangrijk zijn kunnen we naar de voordelen van zowel web applicaties als standalone/desktop applicaties kijken om vervolgens een keuze te maken.

Voordelen web applicatie

- Snel voor iedereen toegankelijk
- Overal beschikbaar (waar internet is)
- Geen onderhoud (updates installeren) voor de gebruiker
- Niet afhankelijk van besturingssysteem

Voordelen standalone/desktop applicatie

- Veiliger, er wordt geen privacy gevoelige data over het netwerk verstuurd
- Lagere kosten voor de gebruiker op de lange termijn
- Geen internet verbinding nodig
- Sneller
- Mogelijkheid om backups van data te maken

Hier uit blijkt dat een web applicatie het beste aansluit bij de aspecten die voor ons van belang zijn. Wel moet er goed gekeken worden naar de privacy bij de ontwikkeling van een dergelijke applicatie, dit omdat we gegevens over een netwerk versturen. Om te voorkomen dat andere mensen mee kijken met ons netwerk verkeer kunnen we gebruik maken van beveiligde protocollen.

3.2 Server

Zoals in de voorgaande sectie beschreven is er gekozen voor een web applicatie. Hiervoor moet een server geïmplementeerd worden, in deze sectie worden de ontwerp keuzes besproken die betrekking hebben op deze webserver.

3.3 Python, PHP of Ruby on Rails

Drie mogelijke talen om een web applicatie mee te ontwikkelen zijn PHP¹, Ruby² en Python³, van deze talen is PHP de meest gebruikte. In figuur 3.1 wordt een vergelijking gemaakt tussen de drie talen. In figuur 3.2 is te zien dat PHP verreweg de meest gebruikte taal is.

3.3.1 PHP

PHP, *PHP: Hypertext Preprocessor*, is een scripttaal waarmee dynamische webpagina's ontwikkeld kunnen worden. De taal is in 1994 ontworpen door Rasmus Lerdorf, en geïnspireerd door Perl⁴. Tegenwoordig is dit de meest gebruikte taal om web applicaties mee te ontwikkelen.

3.3.2 Ruby


















Ruby is een programmeertaal waarmee eenvoudig en snel objectgeëoriëteerd geprogrammeerd kan worden, de taal is in 1995 gepubliceerd door Yukihiro Matsumoto. Sinds 2004 is Ruby on Rails beschikbaar, dit is een open source web applicatie-framework geschreven in Ruby.

¹<http://php.net/>

²<https://www.ruby-lang.org/en/>

³<https://www.python.org/>

⁴<https://www.perl.org/>

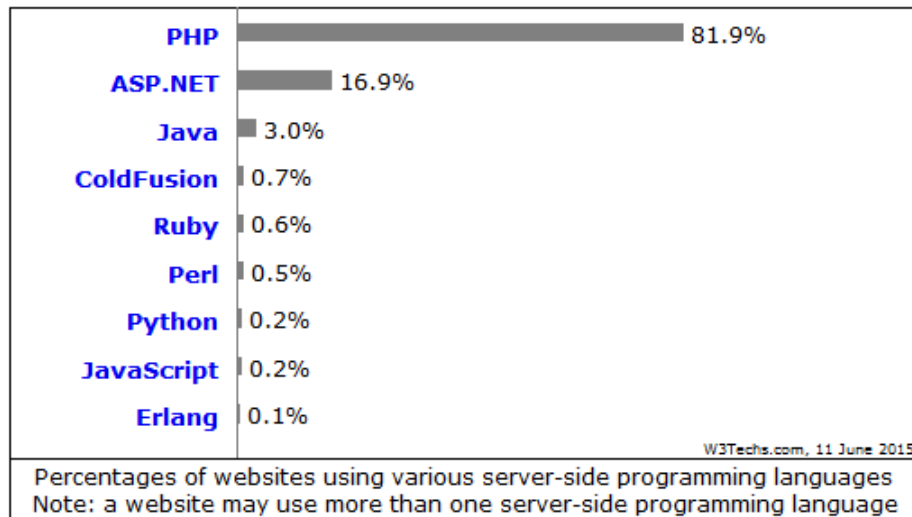
	PHP	RUBY	PYTHON
CURRENT VERSION	PHP: 5.3.8 AUGUST 23, 2011	RUBY: 1.9.3 OCTOBER 31, 2011	PYTHON: 3.2.2 SEPTEMBER 4, 2011
PURPOSE	 PHP was designed for web development to produce dynamic web pages.	 Ruby was designed to make programming fun and flexible for the programmer.	 Python was designed to emphasize productivity and code readability.
CREATOR & YEAR RELEASED	1995 RASMUS LERDORF	1995 YUKIHIRO "MATZ" MATSUMOTO	1991 GUIDO VAN ROSSUM
INFLUENCED BY	<ul style="list-style-type: none"> C PERL JAVA C++ TCL 	<ul style="list-style-type: none"> ADA C++ CLU DYLAN EIFFEL LISP PERL PYTHON 	<ul style="list-style-type: none"> ABC ALGOL 68 C C++ ICON JAVA LISP PERL
SITES BUILT USING IT	 WIKIPEDIA  UDEMY  FACEBOOK	 TWITTER  HULU  GROUPON	 YOUTUBE  GOOGLE
USABILITY	 PHP follows a classic approach and is extensively documented.	 Programmers describe Ruby code as elegant, powerful, and expressive. It is highly usable because of its principle of least astonishment, enforced to minimize confusion for users.	 Python uses strict indentation enforcements. Python is arguably the most readable programming language.
EASE OF LEARNING	 PHP is easy to learn for former C programmers.	 Ruby is better for a programmer who already knows a language or two.	 Python is great for beginners, often recommended by programmers due to the simplicity of its syntax.

Figuur 3.1: Vergelijking PHP, Ruby en Python, bron: Udemy

3.3.3 Python

Python is een in 1991 verschenen programmeertaal, de taal is door Guido van Rossum ontwikkeld. Het is een van de eenvoudigste talen om te leren en omdat het erg op psuedo code lijkt is het meestal ook goed te lezen en te begrijpen voor mensen die zelf niet kunnen programmeren.

Voor dit project is gekozen om gebruik te maken van Python, in combinatie met CherryPy en Cheetah. We hebben hier voor gekozen omdat het een stuk sneller is dan PHP wat betreft run time en het aantal regels code nagenoeg gelijk is aan het aantal dat bij PHP nodig zou zijn, zie



Figuur 3.2: Vergelijking PHP, Ruby en Python, bron: <http://w3techs.com>

figuur 3.3. Hierbij is er ook rekening mee gehouden dat met Python ook standalone applicaties gemaakt kunnen worden, iets wat met PHP (bijna) niet mogelijk is. Op deze manier kan een groot deel van de code direct gebruikt worden als er behoefte is aan een standalone/desktop applicatie.

3.3.4 CherryPy

CherryPy [16] is een open-source (BSD license⁵) minimalistisch Python Web Framework, hiermee is het mogelijk om web applicaties te maken op (bijna) de zelfde manier als een objectgeëoriënteerd Python programma. Het resultaat hiervan is dat er minder code nodig is en het ontwikkelen van een web applicatie minder tijd kost.

Inmiddels bestaat CherryPy al meer dan tien jaar en heeft het bewezen zeer snel en stabiel te zijn. Het wordt gebruikt bij veel websites, van kleine simpele sites tot zeer veel eisende applicaties.

Een voorbeeld van een simpele website met CherryPy is het volgende:

```
import cherrypy

class HelloWorld(object):
    @cherrypy.expose
    def index(self):
        return "Hello world!"

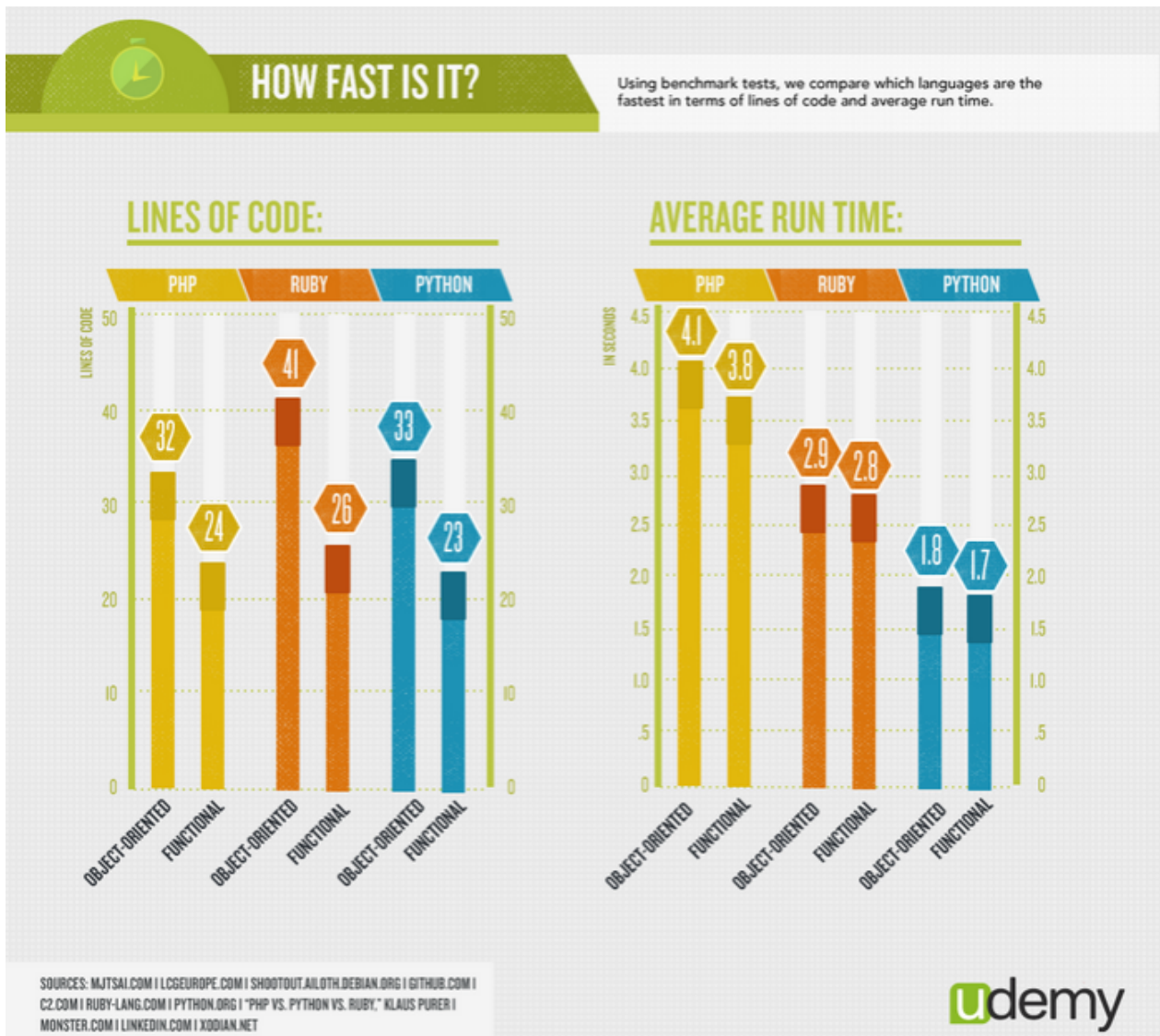
if __name__ == '__main__':
    cherrypy.quickstart(HelloWorld())
```

Deze code start een webserver met de standaard configuratie en print de tekst “Hello World!” op de pagina. Het is ook mogelijk om een eigen configuratie voor de server te gebruiken, zoals hier onder te zien is, hierin is *cherry_conf* een Python dictionary⁶. Op deze manier kan er bijvoorbeeld handmatig bepaald worden op welk IP adres de server te vinden is en welke poort, standaard 8080, er gebruikt wordt.

```
cherrypy.quickstart(HelloWorld(), config = cherry_conf)
```

⁵https://en.wikipedia.org/wiki/BSD_licenses

⁶<https://docs.python.org/2/tutorial/datastructures.html#dictionaries>



Figuur 3.3: Vergelijking snelheid PHP, Ruby en Python, bron: Udemy

In plaats van een simpele tekst op het scherm willen we graag een complete HTML pagina terug geven, dit doen we door gebruik te maken van Cheetah.

3.3.5 Cheetah

Cheetah [17] of Cheetah Template is een template engine die gebruik maakt van Python. Het kan apart gebruikt worden of in combinatie met andere tools en frameworks, zoals bijvoorbeeld CherryPy. Over het algemeen wordt het gebruikt voor server-side scripting en het genereren van dynamische webpagina's. Een simpel voorbeeld is hier onder te zien, deze Cheetah template genereert een HTML pagina met daarop een tabel waarin voor elke client een regel wordt gemaakt met de achternaam, voornaam en een link naar het emailadres.

```
<html>
  <head>
    <title>
      $title
    </title>
  </head>
```

```

<body>
  <table>
    #for $client in $clients
      <tr>
        <td>
          $client.surname, $client.firstname
        </td>
        <td>
          <a href="mailto:$client.email">$client.email</a>
        </td>
      </tr>
    #end for
  </table>
</body>
</html>

```

Voor dit project gebruiken we Cheetah om een template te maken van de User Interface.

3.3.6 JSON

JSON [18] staat voor Javascript Object Notation, het is een simpel formaat voor gegevens uitwisseling. Het is gemakkelijk door computers te verwerken en goed leesbaar voor mensen. Nagenoeg alle programmeertalen ondersteunen JSON, zo ook Python. Hier onder volgt een voorbeeld van JSON.

```

[
  {
    "Naam": "JSON",
    "Type": "Gegevensuitwisselingsformaat",
    "isProgrammeertaal": false,
    "Zie ook": [ "XML", "ASN.1" ]
  },
  {
    "Naam": "JavaScript",
    "Type": "Programmeertaal",
    "isProgrammeertaal": true,
    "Jaar": 1995
  }
]

```

Wij gebruiken JSON om de lijst met steden en landen terug te sturen naar de client en ook de nieuwsberichten worden als JSON object opgevraagd.

3.3.7 AJAX

AJAX [19], *Asynchronous JavaScript And XML*, wordt gebruikt bij de ontwikkeling van interactieve webpagina's, waarbij asynchroon gegevens van de server worden opgehaald. Dat wil zeggen dat we niet de hele pagina hoeven te vernieuwen als we bijvoorbeeld een andere afbeelding willen laden. Bij dit project maken we gebruik van AJAX om de coördinaten van het middelpunt en de straal van het geselecteerde gebied naar de server te verstruven. Ook het uitvoeren van zoekopdrachten voor nieuwsberichten gaat via AJAX. Hiervoor gebruiken we de standaard functies die voor AJAX in JQuery [21] geïmplementeerd zijn.

3.3.8 JQuery

JQuery is een JavaScript framework waarmee dynamische en interactieve websites gemaakt kunnen worden. Hiermee kunnen we de DOM en CSS van websites aanpassen zonder de bron code aan te moeten passen, op deze manier kunnen we bijvoorbeeld een tabel uitbreiden of de lay-out

aanpassen voor één gebruiker. Daarnaast heeft JQuery ook ondersteuning voor AJAX. JQuery is beschikbaar onder de MIT-licentie en de GNU General Public License.

3.4 Database

Voor de database hebben we gekozen voor een SQLite [14] database. SQLite is een software pakket wat een SQL database engine implementeerd, maar in tegenstelling tot bijvoorbeeld MySQL is hier geen client/server systeem voor nodig. Bij SQLite wordt de database in één bestand op de schijf opgeslagen.

Omdat SQLite standaard geen ondersteuning heeft voor veel wiskundige functies zijn deze toegevoegd via een extensie. Hiervoor wordt het bestand *extension-functions.c*⁷ gebruikt, hierin zijn functies als *cos* en *sin* geïmplementeerd.

3.5 Geografische informatie

Voor de geografische informatie, de steden met hun breedtegraad en lengtegraad, populatie en landen is gekeken naar de mogelijkheid om dit via Google Maps [9] te doen. Hoewel dit in theorie mogelijk is, is dit in praktijk geen goede optie. We zouden dan alle steden die in Google Maps te vinden zijn eerst moeten downloaden en opslaan in een database voordat we kunnen gaan berekenen of ze in het geselecteerde gebied liggen.

Een andere mogelijkheid waar naar gekeken is, is om gebruik te maken van Wolfram Alpha [31], hiermee is het mogelijk om de afstand tussen twee steden op aarde te berekenen. Wel moeten we dan zelf de breedtegraad en lengtegraad van het start en eind punt hebben, we kunnen hiermee dus geen steden zoeken.

GeoNames [20] is een database waar (vrijwel) alle steden op de wereld in te vinden zijn. De database bevat heel veel informatie waarvan, voor ons, vooral de naam van de stad, de populatie, de breedtegraad en lengtegraad en het land waar de stad in ligt interessant zijn. Voor landen en steden zijn ook alternatieve namen, in andere talen dan Engels, te vinden in de database. Daarnaast zijn ook zeeën, oceanen, meren, bergen etc. op te zoeken. Voor dit project is alleen gekeken naar steden en landen maar voor een toekomstig onderzoek zijn dit interessante opties.

De database is via een API te benaderen of te downloaden als tekstbestand. Bij dit project is voor het laatste gekozen om het netwerk verkeer zo beperkt mogelijk te houden en om te zorgen dat de data altijd beschikbaar is. Omdat een tekstbestand niet efficiënt te doorzoeken is schrijven we de data naar onze eigen SQLite database.

Tijdens de ontwikkeling van de applicatie bleek dat voor een deel van de steden, 32480 van de 144573, geen populatie beschikbaar is. Omdat we bij het algoritme voor de gebiedsrepresentatie kijken wat de grootste stad in een gebied is en dit niet mogelijk is met twee of meer steden waarvan de populatie gelijk is aan 0 worden deze steden uitgesloten tijdens de selectie. In sectie 6.2.1 wordt een voorstel gedaan hoe dit opgelost zou kunnen worden zodat deze steden wel gebruikt kunnen worden. Een ander probleem wat tijdens de implementatie boven kwam is dat een aantal grote steden, zoals Londen en Parijs, in meerdere delen in de database opgeslagen zijn. Ook dit kan voor problemen zorgen bij het algoritme voor de gebiedsrepresentatie. Een oplossing hiervoor is te vinden in sectie 6.2.2.

⁷<https://www.sqlite.org/contrib>

3.6 Nieuwsbronnen

Voor de ontwikkeling van de applicatie is gekeken naar meerdere nieuwsbronnen. Hier zullen we een aantal van deze bronnen bespreken.

3.6.1 Google News

Google News [10] is een zoekmachine van Google om nieuws berichten te doorzoeken. De titles, eerste alinea's en afbeeldingen van nieuwsberichten op het internet worden verzameld en kunnen aan de hand van keywords worden doorzocht. Voor deze dienst is een API beschikbaar maar deze wordt sinds 26 mei 2011 niet meer ondersteund. Er kan nog wel gebruik van gemaakt worden, maar de API wordt niet meer bijgewerkt. Daarnaast is er geen garantie dat de API goed werkt.

3.6.2 BING News

BING News [11] is een vergelijkbare zoekmachine van Microsoft. Ook hier is een API voor beschikbaar, welke nog wel ondersteund wordt. Per maand kunnen hier gratis 5000 zoekopdrachten mee uitgevoerd worden, mocht er meer nodig zijn dan is dit mogelijk tegen betaling.

3.6.3 Overig

Naast Google News en BING News hebben we nog naar een aantal andere optie gekeken.

Yahoo News Service

Ook deze zoekmachine is te vergelijken met Google News en BING News. Hier voor is ook een API [12] beschikbaar, deze is echter alleen te gebruiken tegen betaling. Voor dit project is dat niet haalbaar maar mogelijk dat dit in de toekomst een interessante optie is.

FAROO

FAROO [13] is een alternatief voor de hierboven genoemde opties. Het is een gratis zoek API waarbij tot 1 miljoen zoekopdrachten per maand gedaan kunnen worden.

Lokaal nieuws

Een hele interessante optie is om lokale nieuws sites en kranten te zoeken bij het geselecteerde gebied. Vooral voor kleinere gebieden, zal dit nuttige resultaten op kunnen leveren. in sectie 6.2.3 gaan we hier dieper op in.

3.7 Zoeken naar nieuws

Het zoeken naar nieuws gaat via de BING API [11]. Hiermee is het mogelijk om gratis 5000 zoekopdrachten per maand uit te voeren, tegen betaling kunnen er meer zoekopdrachten uitgevoerd worden. We kunnen het beste zoeken aan de hand van de naam van een stad en het land waar deze stad in te vinden is. Dit is nodig omdat wanneer we alleen op de naam van een stad zoeken, en deze stad komt in meerdere landen voor, we ook resultaten uit andere landen krijgen die niks met het zoek gebied te maken hebben. Bijvoorbeeld Willemstad in Noord Brabant, Nederland en Willemstad in Curaçao. Voor dit project maken we hier geen gebruik van omdat als we een zoekopdracht uitvoeren met "Willemstad, Nederland" we ook nieuws terug krijgen waar alleen "Nederland" in vermeldt wordt, hier is nog geen goede oplossing voor gevonden.

3.7.1 Client side of server side?

Het zoeken naar nieuws kan zowel client side als server side gebeuren. Het zoeken van nieuws door de server heeft als voordeel dat het mogelijk is om een cache aan te leggen, van nieuws wat bij een stad hoort. Daarbij komt dat de sleutel van de BING API niet beschikbaar is voor de gebruiker en dus niet misbruikt kan worden. Het is echter toch wenselijk om dit client side te doen omdat we op deze manier de netwerk en server belasting kunnen verspreiden.

3.8 Ideaal ontwerp

In deze sectie beschrijven we een voorstel hoe een dergelijke applicatie eigenlijk zou moeten. Helaas is het vooral door gebrek aan tijd en middelen niet mogelijk om het op deze manier te doen.

Zoals eerder gezegd heeft het zoeken naar nieuws door de server een aantal voordelen. Dit zoeken zou dan ook niet meer via bijvoorbeeld BING News gaan, een betere optie is om gebruik te maken van RSS feeds [22].

RSS feeds

RSS staat voor *Rich Site Summary*, maar ook vaak *Really Simple Syndication* genoemd, het is een eenvoudige manier om op de hoogte te blijven van bijvoorbeeld het laatste nieuws. Het is gemaakt zodat we niet zelf alle sites hoeven te bezoeken waarvan we op de hoogte willen blijven van de nieuwste berichten. In plaats daarvan kunnen we de RSS feed in de gaten houden, de nieuwste berichten komen hier automatisch op te staan.

Met een achtergrond proces op de server kunnen deze RSS feeds in de gaten gehouden worden en zodra er een nieuw nieuws bericht beschikbaar is wordt deze gedownload. Eenmaal gedownload moet een bericht geanalyseerd worden om te bepalen bij welke stad het bericht hoort. Nieuwsberichten die geanalyseerd zijn worden opgeslagen in een database, met bijvoorbeeld de stad en het land waar het bericht bij hoort, een link naar het bericht en de tijd dat het nieuwsbericht gepubliceerd is. Deze tijd kan gebruikt worden om te bepalen of het nog een actueel bericht is.

Met de hier beschreven methode verstuurt de client alleen nog het middelpunt en de straal van de op de kaart gemaakte selectie. De server doet de rest van het werk, dat wil zeggen het zoeken naar steden die bij het geselecteerde gebied horen, het zoeken naar nieuws dat bij deze steden horen en vervolgens deze lijst met nieuwsberichten terug sturen naar de client. Op deze manier is het ook eenvoudig om nieuwsbronnen toe te voegen, we hoeven alleen de RSS feed van de bron in de gaten te houden.

3.9 Samenvatting

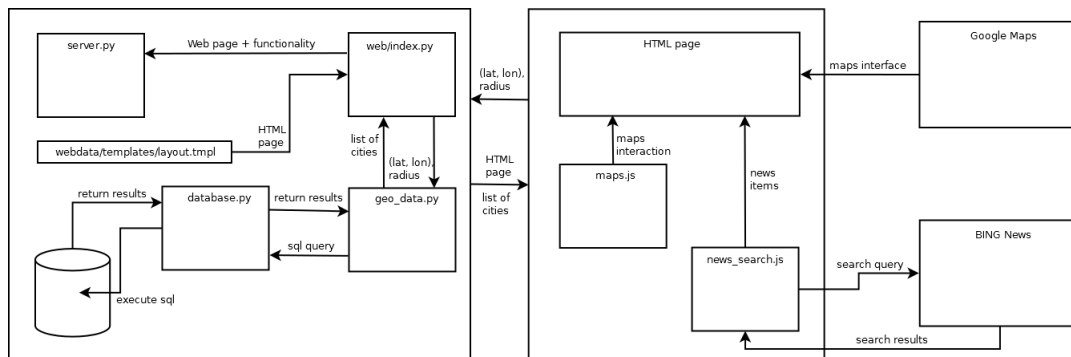
In dit hoofdstuk hebben we gezien hoe we een applicatie om nieuws te zoeken aan de hand van een locatie kunnen opbouwen. Er is gekozen om een web applicatie te ontwikkelen waarbij de server verantwoordelijk is voor de selectie van de steden die in een gebied liggen. De client zoekt vervolgens naar nieuws.

Ook hebben we gezien welke nieuwsbron we kunnen gebruiken, namelijk BING News. Daarnaast hebben we gekeken naar andere mogelijke nieuwsbronnen zoals Google News, Faroo en Yahoo News Service.

Met dit hoofdstuk hebben we de vraag *Welke nieuwsbronnen kunnen we gebruiken?* beantwoord. Ook is er een begin gemaakt met het antwoord op de vraag *Hoe implementeren we een dergelijke applicatie?*, hiermee zullen we verder gaan in het volgende hoofdstuk.

Implementatie

Hier onder zullen we de implementatie van de hiervoor gemaakte ontwerpkeuzes beschrijven. De



Figuur 4.1: Architectuur

code en de installatie van de server zijn uitgebreider toegelicht in bijlage A.

4.1 Server en database

4.1.1 Server

De server is geschreven in Python en maakt gebruik van CherryPy en Cheetah. In principe is de server geïmplementeerd als webserver maar zou ook gebruikt kunnen worden als API [33] voor een andere applicatie.

Webserver

Op de webserver draait één pagina, *web/index.py*, de template met daarin alle HTML voor deze pagina is te vinden in *webdata/templats/layout.tmpl*.

Wanneer *web/index.py* een HTTP GET request [32] krijgt wordt de HTML uit deze template terug gestuurd naar de gebruiker.

Zodra een gebruiker een gebied op de kaart selecteerd wordt er een HTTP POST request door de client naar de server gestuurd. De door de client opgestuurde data, de breedtegraad en lengtegraad van het middelpunt en straal van het gebied, wordt gecontroleerd en de steden die bij dit gebied horen worden opgezocht. Deze steden worden in een JSON object terug gestuurd naar de client.

API

Om de server te gebruiken als API voor een andere applicatie maken we alleen gebruik van de HTTP POST request [32], hierin sturen we de breedtegraad en lengtegraad van het middelpunt en de straal van het gebied. Vervolgens krijgen we een JSON object terug met de steden die dit gebied representeren welke we verder kunnen verwerken voor onze applicatie.

```
POST / HTTP/1.1
Host: 192.168.0.18:8080
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:38.0) Gecko/20100101 Firefox/38.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: nl,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://192.168.0.18:8080/
Content-Length: 68
Cookie: session_id=5c6f93c64ae57b5c0433122dac235099a1dcb4d5
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
```

```
lat=52.3740300&lon=4.8896900&radius=50.0
```

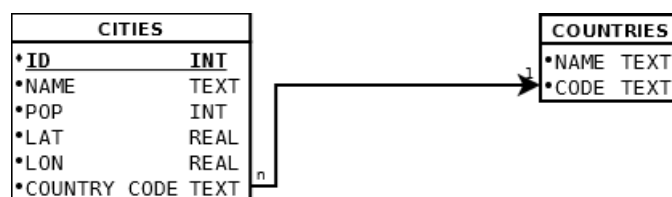
Als we het hier bovenstaande HTTP POST request naar de server sturen die op IP adres 192.168.0.18 : 8080 draait krijgen we een lijst met steden terug die binnen 50km van Amsterdam liggen.

4.1.2 Database

Zoals uitgelegd in sectie 3.4 maken we gebruik van een SQLite database, Python heeft hier ondersteuning voor via de *sqlite3* [23] module.

De database, figuur 4.1.2 bevat twee tabellen, *cities* en *countries*. De *cities* tabel bevat alle steden, elke stad in de tabel heeft een uniek ID, een naam, een aantal inwoners, een code voor het land waarin deze stad ligt en de coördinaten, breedtegraad en lengtegraad, van de stad. De landcode verwijst naar de tabel *countries*, hiermee kan de naam van het land waarin de stad ligt opgezocht worden.

De tweede tabel in de database is de *countries* tabel, met velden voor de naam en de code van het land.



Figuur 4.2: Database

database.py

Bij een SQL query is een groot deel van de query vast, bij bijvoorbeeld een SELECT query veranderen alleen de naam van de tabel en de velden die we willen selecteren.

```
SELECT x, y, z FROM table;
```

Om niet elke keer een hele query te moeten maken staan er in dit bestand functies waarbij we aan kunnen geven welke velden van welke tabel we willen selecteren, de functie maakt de SQL query en voert deze ook direct uit. Er zijn functies voor het selecteren, toevoegen, updaten en verwijderen van entries in de database. Daarnaast zijn er ook functies voor het maken en verwijderen van tabellen.

4.1.3 geo_data.py

In dit Python bestand staan functies voor het downloaden van de geografische informatie van *geonames.org*. Deze informatie is beschikbaar in een tekst bestand, omdat dit niet efficiënt is als er gekeken moet worden welke steden er in een gebied liggen is er een functie om deze data naar de database te schrijven. In principe wordt dit alleen de eerste keer gedaan, maar mocht het nodig zijn kan de database opnieuw opgebouwd worden.

Behalve de functies om de database te downloaden en op te bouwen zijn er ook functies om de steden te zoeken en een algoritme om te bepalen welke steden het gebied representeren hier geïmplementeerd.

4.2 Welke steden liggen in het geselecteerde gebied?

Een door de gebruiker op de kaart geselecteerd gebied wordt weergegeven door de coördinaten van het middelpunt (breedtegraad¹ en lengtegraad²) en de straal van de cirkel (het gebied) in kilometers.

Om te berekenen welke steden binnen dit gebied liggen moet de afstand van het middelpunt tot een stad berekend worden. Is deze afstand kleiner dan of gelijk aan de straal van het geselecteerde gebied, dan ligt de stad in het gebied.

Voor het berekenen van de afstand tussen twee punten $P_1(lat_1, lon_1)$ en $P_2(lat_2, lon_2)$ op een bol, zoals de aarde, kan geen gebruik gemaakt worden van de stelling van Pythagoras [35] omdat deze stelling uit gaat van een plat vlak. Om de afstand tussen twee punten op een bol te berekenen kan gebruik gemaakt worden van bijvoorbeeld de *spherical law of cosines* [26] of de *haversine formule* [27]. De *haversine formule* is vooral voor kleine afstanden nauwkeuriger, met deze formule kunnen afstanden tot ongeveer één meter nauwkeurig berekend worden. Bij de *spherical law of cosines* is de maximale fout een paar meter. Om deze reden wordt de *haversine formule* meer gebruikt bij navigatie, voor dit project is een fout van een paar meter echter niet belangrijk. Er is daarom gekozen voor de *spherical law of cosines*.

$$distance = \arccos(\sin(lat_1) \cdot \sin(lat_2) + \cos(lat_1) \cdot \cos(lat_2) \cdot (\lon_1 - \lon_2)) \cdot R$$

Hierin zijn (lat_1, lon_1) de coördinaten van het startpunt, in dit geval het middelpunt van het geselecteerde gebied, en (lat_2, lon_2) de coördinaten van het tweede punt waarvoor we de afstand tot het start punt willen berekenen. Deze coördinaten worden door de client in graden door gestuurd en moeten eerst omgezet worden naar radialen om in deze formule gebruikt te kunnen worden. R is de straal van de bol waarop beide punten liggen, in dit geval de aarde $\Rightarrow R = 6371km$.

Met deze formule zou meteen een *SQL [34] query* geconstrueerd kunnen worden.

```
SELECT
    cities.NAME, cities.LAT, cities.LON, cities.POP
FROM
    cities
```

¹<http://en.wikipedia.org/wiki/Latitude>

²<http://en.wikipedia.org/wiki/Longitude>

WHERE

```
acos(sin(lat) * sin(cities.lat) + cos(lat) * cos(cities.lat) * cos(lon - cities.lon)) *  
6371 <= radius;
```

Deze query geeft als resultaat een lijst met steden die binnen het door de gebruiker geselecteerde gebied liggen. Voor elke stad wordt de naam, populatie, breedtegraad en lengtegraad terug gegeven.

Nadeel van deze query is echter dat voor alle steden in de database berekend moet worden of de afstand tot het middelpunt van het geselecteerde gebied kleiner dan of gelijk is aan de straal van het gebied. Dit kost veel tijd, om dit te versnellen selecteren we alleen kandidaat steden waar we vervolgens de berekening op uit gaan voeren. Door de minimale en maximale breedtegraad en lengtegraad te berekenen kunnen we al een heleboel steden uitsluiten die sowieso buiten het geselecteerde gebied liggen.

Berekenen minimale en maximale breedtegraad

Als je van een punt A naar een punt B op een cirkel gaat kun je de hoek, r , tussen deze twee punten berekenen. Lengtecirkels (meridianen)³ zijn dergelijke cirkles op aarde met een straal van $R = 6371km$. Je kan dus over een meridiaan verplaatsen, dat wil zeggen de lengtegraad blijft constant, en simpelweg r aftrekken/optellen bij de breedtegraad van het middelpunt van het geselecteerde gebied om de minimale en maximale breedtegraad te krijgen.

$$lat_{min} = lat - r$$

$$lat_{max} = lat + r$$

r hangt af van de afstand tussen de twee punten en de straal R van de aarde. De afstand tussen de twee punten stellen we gelijk aan de maximale afstand dat een stad nog in het geselecteerde gebied ligt, dat is de straal van het gebied.

$$r = \frac{\max.distance}{R} = \frac{radius}{R}$$

Berekenen minimale en maximale lengtegraad (methode 1)

Een door veel applicatie gebruikte benadering om de minimale en maximale lengtegraad te berekenen is een vaste breedtegraad kiezen en de lengtegraad aanpassen. dat betekent dat er over een breedtecirkel [28] bewogen wordt. In deze sectie zullen we aan de hand van een voorbeeld laten zien dat dit geen nauwkeurige resultaten geeft.

Als we over een breedtecirkel bewegen, bewegen we over een zogenaamde kleine cirkel [29]. Een breedtecirkel op breedtegraad $lat = 1.3963$ heeft een straal $R_S = R \cdot \cos(lat) = 6371 \cdot \cos(1.3963) = 1106km$. Een afstand van $1000km$ op deze breedtecirkel komt nu overeen met een hoek $r_S = \frac{d}{R_S} = \frac{1000}{1106} = 0.9039$ tussen twee punten op deze cirkel. Bij een lengtegraad van -0.6981 van het beginpunt komen de minimale en maximale lengtegraad nu op

$$lon_{min} = -0.6981 - 0.9039 = -1.6020$$

$$lon_{max} = -0.6981 + 0.9039 = 0.2058$$

Dit is echter **niet** de minimale en maximale lengtegraad die we kunnen bereiken door $1000km$ in een willekeurige richting te bewegen vanaf het beginpunt $M = (1.3969, -0.6981)$. Dat komt omdat we ons verplaatsen over een kleine cirkel, afstanden over een kleine cirkel zijn groter dan over een grote cirkel [30]. Hoewel we dus $1000km$ afgelegd hebben op een kleine cirkel om van $M = (1.3969, -0.6981)$ naar $P_S = (1.3969, -1.6020)$ kunnen we “afsnijden” op een grote cirkel. Deze afstand is

³[http://en.wikipedia.org/wiki/Meridian_\(geography\)](http://en.wikipedia.org/wiki/Meridian_(geography))

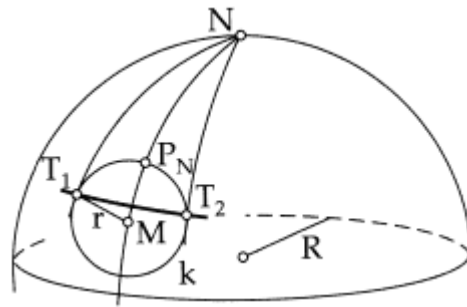
$$\begin{aligned} dist &= \arccos(\sin(lat_1) \cdot \sin(lat_2) + \cos(lat_1) \cdot \cos(lat_2) \cdot \cos(lon_1 - lon_2)) \cdot R \\ &= \arccos(\sin(1.3963) \cdot \sin(1.3963) + \cos(1.3963) \cdot \cos(1.3963) \cdot \cos(-0.6981 - (-1.6020))) \cdot 6371 \\ &= 967km \end{aligned}$$

We kunnen dus nog $33km$ verder verplaatsen en zitten dan nog steeds binnen het geselecteerde gebied, dat wil zeggen dat we een kleinere en grotere minimale en maximale lengtegraad kunnen bereiken.

Als we dus gebruik maken van $lon \pm \frac{d}{\cos(lat)}$ als grenzen voor de lengtegraad, sluiten we een aantal plaatsen uit die in werkelijkheid wel in het gebied liggen. Een punt $P = (1.4618, -1.6021)$ ligt buiten de berekende grenzen voor een gebied met middelpunt $M = (1.3969, -0.6981)$ en straal $d = 1000km$ terwijl de afstand van M maar $872km$ is, een fout van meer dan 12%.

Berekenen minimale en maximale lengtegraad (methode 2)

De methode omschreven in de vorige sectie om de minimale en maximale lengtegraad te berekenen werkt dus niet goed. Dit is te zien in figuur 4.3, de punten op de cirkel met de minimale en maximale lengtegraad T_1 en T_2 liggen niet op de zelfde breedtecirkel [28] als M , het middelpunt van het geselecteerde gebied maar dichtbij de pool. In plaats daarvan moeten de volgende



Figuur 4.3: Tangent meridians to the query circle

formules gebruikt worden

$$\begin{aligned} lat_T &= \arcsin\left(\frac{\sin(lat)}{\cos(r)}\right) \\ \Delta lon &= \arccos\left(\frac{\cos(r) - \sin(lat_T) \cdot \sin(lat)}{\cos(lat_T) \cdot \cos(lat)}\right) \\ &= \arcsin\left(\frac{\sin(r)}{\cos(lat)}\right) \end{aligned}$$

$$\begin{aligned}lon_{min} &= lon_{T_1} = lon - \Delta lon \\lon_{max} &= lon_{T_2} = lon + \Delta lon\end{aligned}$$

SQL query

Aan de hand van de hierboven berekende minimale en maximale breedtegraad en lengtegraad kunnen we nu een SQL query maken die de berekening, de spherical law of cosines, alleen hoeft uit te voeren op een veel kleiner aantal kandidaat plaatsen.

```
SELECT
    cities.NAME, cities.LAT, cities.LON, cities.POP
FROM
    cities
WHERE
    (cities.LAT >= lat_min AND cities.LAT <= lat_max)
    AND
    (cities.LON >= lon_min AND cities.LON <= lon_max)
```

```

GROUP BY
    cities.NAME
HAVING
    acos(sin(lat) * sin(cities.LAT) + cos(lat) * cos(cities.LAT) *
        cos(cities.LON - (lon))) <= radius;

```

Deze query selecteert de naam, breedtegraad, lengtegraad en populatie voor elke stad die binnen het geselecteerde gebied valt. Eerst worden alle steden geselecteerd waarvoor de breedtegraad en de lengtegraad binnen de minimale en maximale waardes vallen. Vervolgens wordt voor elk van deze kandidaat plaatsen uitgerekend of deze ook echt binnen het geselecteerde gebied liggen. De query is vervolgens aangepast om ook de landen waarin de steden liggen te selecteren. Dit is nodig omdat sommige steden in meerdere landen voorkomen. Doe je dit niet kan het voorkomen dat je nieuws uit een heel ander wereld deel krijgt dan het gebied wat de gebruiker geselecteerd heeft.

```

SELECT
    cities.NAME, cities.LAT, cities.LON, cities.POP, countries.NAME
FROM
    cities, countries
WHERE
    cities.COUNTRY_CODE = countries.CODE
    AND
    (cities.LAT >= lat_min AND cities.LAT <= lat_max)
    AND
    (cities.LON >= lon_min AND cities.LON <= lon_max)
GROUP BY
    cities.NAME
HAVING
    acos(sin(lat) * sin(cities.LAT) + cos(lat) * cos(cities.LAT) *
        cos(cities.LON - (lon))) <= radius;

```

Omdat een deel van de steden in de database een populatie van 0 heeft (32480 van de 144573 steden), en het algoritme voor de gebiedsrepresentatie de grootste stad in een gebied moet vinden selecteren we alleen de steden waarvan de populatie **niet** 0 is.

```

SELECT
    cities.NAME, cities.LAT, cities.LON, cities.POP, countries.NAME
FROM
    cities, countries
WHERE
    cities.COUNTRY_CODE = countries.CODE
    AND
    (cities.LAT >= lat_min AND cities.LAT <= lat_max)
    AND
    (cities.LON >= lon_min AND cities.LON <= lon_max)
    AND
    cities.pop != 0
GROUP BY
    cities.NAME
HAVING
    acos(sin(lat) * sin(cities.LAT) + cos(lat) * cos(cities.LAT) *
        cos(cities.LON - (lon))) <= radius;

```

Een voorbeeld van het resultaat als deze SQL query uitgevoerd wordt is het volgende:

```

[(u'Alkmaar', 0.918595932323124, 0.08287887939312794, 94853, u'Netherlands'),
(u'Amsterdam', 0.9140992660382857, 0.08534118990184153, 741636, u'Netherlands'),
(u'Beverwijk', 0.9160069109107156, 0.08127893606782473, 37585, u'Netherlands'),
(u'Broek in Waterland', 0.9151489070504352,

```

```
...
(u'Zaanstad', 0.9154798214766133, 0.08401247074229824, 140085, u'Netherlands')]
```

4.3 Gebiedsrepresentatie

Bij het geselecteerde gebied worden vaak honderd of meer, en bij een groot gebied duizenden, steden gevonden die in dit gebied liggen, een voorbeeld is te zien in figuur 4.4. Om voor elke stad naar nieuws te zoeken levert evenveel zoekopdrachten als steden op. Dit gaat erg veel tijd in beslag nemen, ongeveer *300ms* per zoekopdracht, en levert te veel nieuws op. Een dergelijke hoeveelheid nieuws, in de orde van duizenden artikelen, is voor de gebruiker niet te verwerken en draagt niet bij aan een beter begrip van het nieuws en wat er in het gebied gaande is.



Figuur 4.4: Alle 12485 steden in een gebied

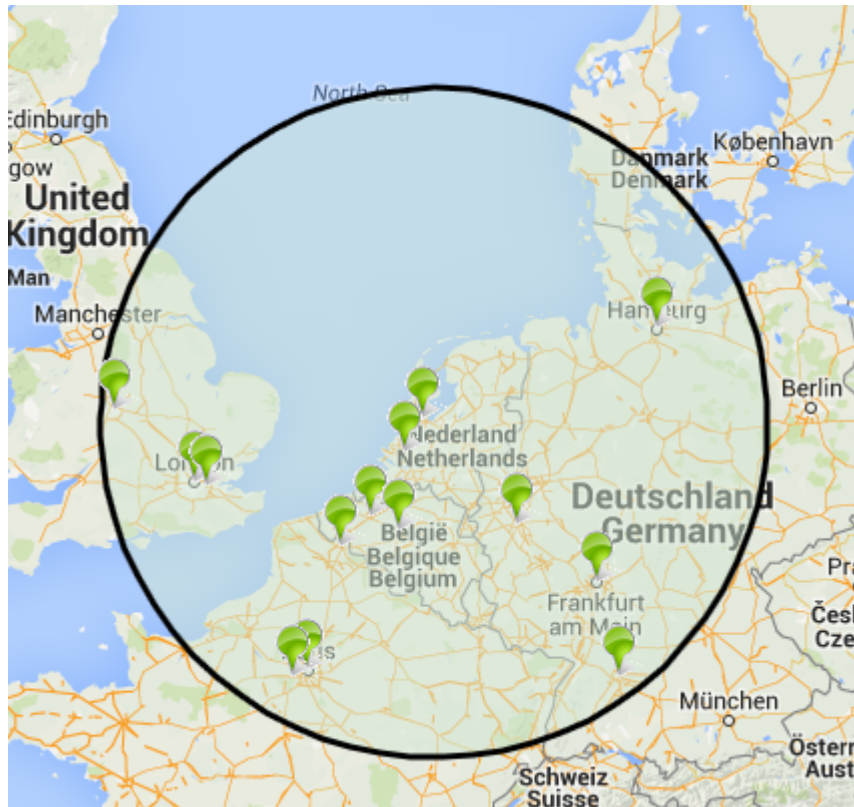
Een gebied moet dus door een kleiner aantal steden gerepresenteerd worden zodat het aantal zoekopdrachten beperkt blijft en de gebruiker de hoeveelheid nieuws goed kan verwerken. Hiervoor zijn verschillende opties die hieronder besproken zullen worden. Een voorbeeld hiervan is in figuur 4.5 te zien, dit is het zelfde gebied maar nu met een beperkt aantal steden.

Aantal steden om een gebied te representeren

Het aantal steden om een gebied te representeren is bepaald aan de hand van een gebruikersonderzoek. De opzet en resultaten van dit onderzoek worden besproken in sectie 5.1.

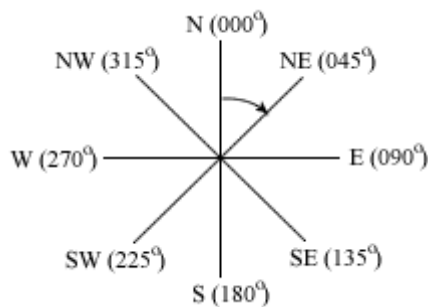
4.3.1 Algoritme 1 - gebiedsrepresentatie

Dit algoritme reduceert een lijst steden tot een maximum aantal zodat een gebied goed gerepresenteerd wordt. Dit gaat op de volgende manier:



Figuur 4.5: Het zelfde gebied maar nu met het algoritme voor gebiedsrepresentatie

Dit recursieve algoritme heeft als input een lijst met steden, dit zijn in eerste instantie alle steden die in het door de gebruiker op de kaart geselecteerde gebied liggen. De grootste stad in dit gebied wordt gezocht en aan het eindresultaat toegevoegd. Vervolgens wordt het gebied in vier delen opgedeeld: Noord-Oost, Zuid-Oost, Zuid-West en Noord-West, zie figuur 4.6. Om deze opdeling te maken wordt voor elke stad de richting, ten opzichte van de grootste stad, berekend (zie algoritme 2) en aan het juiste deel gebied toegevoegd. Als het aantal steden in een deel



Figuur 4.6: Deelgebieden en hun richting

gebied kleiner dan of gelijk is aan 5 wordt de grootste stad in het deelgebied gezocht en aan het eindresultaat toegevoegd. Is het aantal steden echter groter dan wordt het hier boven beschreven proces herhaald voor het deelgebied.

Aan het einde geeft dit algoritme een lijst van steden die het gebied representeren. Als deze lijst nog steeds meer steden bevat dan het maximale aantal steden om een gebied te representeren wordt dit proces herhaald.

Split():

Input: list of cities

Result: Reduced list of cities representing the area

```
while number of result < number of cities to represent the area do
    find biggest city;
    add to result;

    foreach city do
        calculate bearing;
        add to corresponding sublist (nw/ne/se/sw);
    end

    foreach sublist (nw/ne/se/sw) do
        if number of cities <= 5 then
            find biggest city;
            add to result;
        else
            Split(sublist);
            add to result;
        end
    end
end
```

Algorithm 1: Algortime 1 voor gebiedsrepresentatie

GetBearing():

Input: $(lat_1, lon_1), (lat_2, lon_2)$

Output: Bearing in degrees

convert $(lat_1, lon_1), (lat_2, lon_2)$ to radians;

$\Delta lon = lon_2 - lon_1$;

$x = \sin(\Delta lon) \cdot \cos(lat_2)$;

$y = \cos(lat_1) \cdot \sin(lat_2) - (\sin(lat_1) \cdot \cos(lat_2) \cdot \cos(\Delta lon))$;

$bearing = \arctan(\frac{x}{y})$;

convert bearing to degrees;

$bearing = \frac{bearing + 360.0}{360.0}$;

Algorithm 2: Berekenen van de richting

4.3.2 Algoritme 2 - gebiedsrepresentatie

Een andere mogelijkheid om een aantal steden die een gebied vertegenwoordigen te selecteren is weer de grootste stad kiezen maar nu het gebied in twee delen splitsen in plaats van vier zoals bij het eerder beschreven algoritme.

Split():

Input: list of cities

Result: Reduced list of cities representing the area

t: list of list of cities;

```
while number of result < number of cities to represent the area do
  foreach list in t do
    if direction = n/s then
      biggest, north, south = split_north(list);
      append biggest to results;
      append north, south to t;
      remove list from t;
      if last list in t then
        | set direction to e/w;
      end
    else
      biggest, east, west = split_east(list);
      append biggest to results;
      append north, south to t;
      remove list from t;
      if last list in t then
        | set direction to n/s;
      end
    end
  end
end
```

Algorithm 3: Algoritme 2 voor gebiedsrepresentatie

Het algoritme kiest de grootste stad in het gebied, vervolgens wordt er gekeken welke steden ten noorden liggen en welke steden ten zuiden. Voor elk van deze twee deel gebieden wordt weer de grootste stad gezocht, daarna worden deze deel gebieden opnieuw gesplitst maar nu in oost en west.

Voor elk deel gebied wordt dus telkens de grootste stad gezocht en deze wordt aan het eindresultaat toegevoegd. het splitsen van een deelgebied gaat om en om: eerst wordt het gebied in noord en zuid gesplitst, dan wordt elk deel gebied in oost en west gesplitst. Nu hebben we vier deelgebieden die allen weer in noord en zuid worden verdeeld etc.

4.3.3 Andere opties

Naast deze twee algoritmes zijn er nog enkele andere mogelijkheden om steden te kiezen om een gebied te representeren. Deze werken echter minder goed dan de hier boven beschreven algoritmes.

Grootste steden

Een mogelijkheid is om alleen de grootste steden in een gebied te selecteren. Voordeel is dat dit een snel en eenvoudig proces is, nadeel is echter dat de verdeling van de grote steden over het

gebied (vaak) niet uniform is met als gevolg dat de geselecteerde steden geen goede representatie van het gebied geven. Een ander probleem bij het selecteren van alleen de grootste steden is dat grote steden, zoals bijvoorbeeld Londen, in meerdere delen in de database staan. Als je nu de grootste steden kiest worden vaak alle drie de delen gekozen met als gevolg dat er clusters ontstaan en er minder steden in de rest van het gebied gekozen kunnen worden.

Hoofdsteden

In het geval van een selectie van meerdere landen is het een mogelijkheid om alleen de hoofdsteden te gebruiken voor de representatie eventueel aangevuld met de grootste steden in het gebied om voldoende steden te krijgen. Dit is eigenlijk alleen een goede optie als er precies even veel landen als benodigde steden in het gebied liggen. Liggen er minder landen in het gebied, en wordt er aangevuld met de grootste steden, loop je hoogst waarschijnlijk tegen het zelfde probleem aan als wanneer je alleen maar voor de grootste steden kiest. Daarbij komt dat het resultaat vaak nagenoeg gelijk zal zijn omdat hoofdsteden vaak bij de grotere steden horen.

Willekeurig

Uit de lijst met steden die in het gebied liggen kunnen willekeurig een aantal steden gekozen worden welke dit gebied representeren. Omdat het hier mogelijk is dat alleen (of grotendeels) kleine/de kleinste steden gekozen worden is dit een minder geschikte mogelijkheid. Ook is het, net als bij boven genoemde methodes, mogelijk dat de gekozen steden niet verspreid liggen maar zich concentreren in een klein deel van het gebied. Ander groot probleem is dat elke keer dat de gebruiker een gebied selecteert een andere lijst aan steden terug gegeven wordt, hierdoor zullen er elke keer andere nieuws artikelen gezocht worden. Het wordt hierdoor moeilijker (onmogelijk) om een artikel opnieuw op te zoeken met de tool.

4.4 User Interface

De User Interface is te zien in figuur 4.7. Aan de linker kant is een kaart te zien waar de gebruiker met de muis een gebied, in de vorm van een cirkel, kan selecteren. Het selecteren van een gebied gaat als volgt:

- Linker muisklik op de kaart waar het middelpunt van het gebied komt
- Ingedrukt houden en slepen, er is nu een groene cirkel te zien
- Los laten als het gebied de gewenste grootte heeft

Aan de rechter kant is een grijs veld waarin een lijst met links naar nieuws artikelen voor het geselecteerde gebied weergegeven wordt.

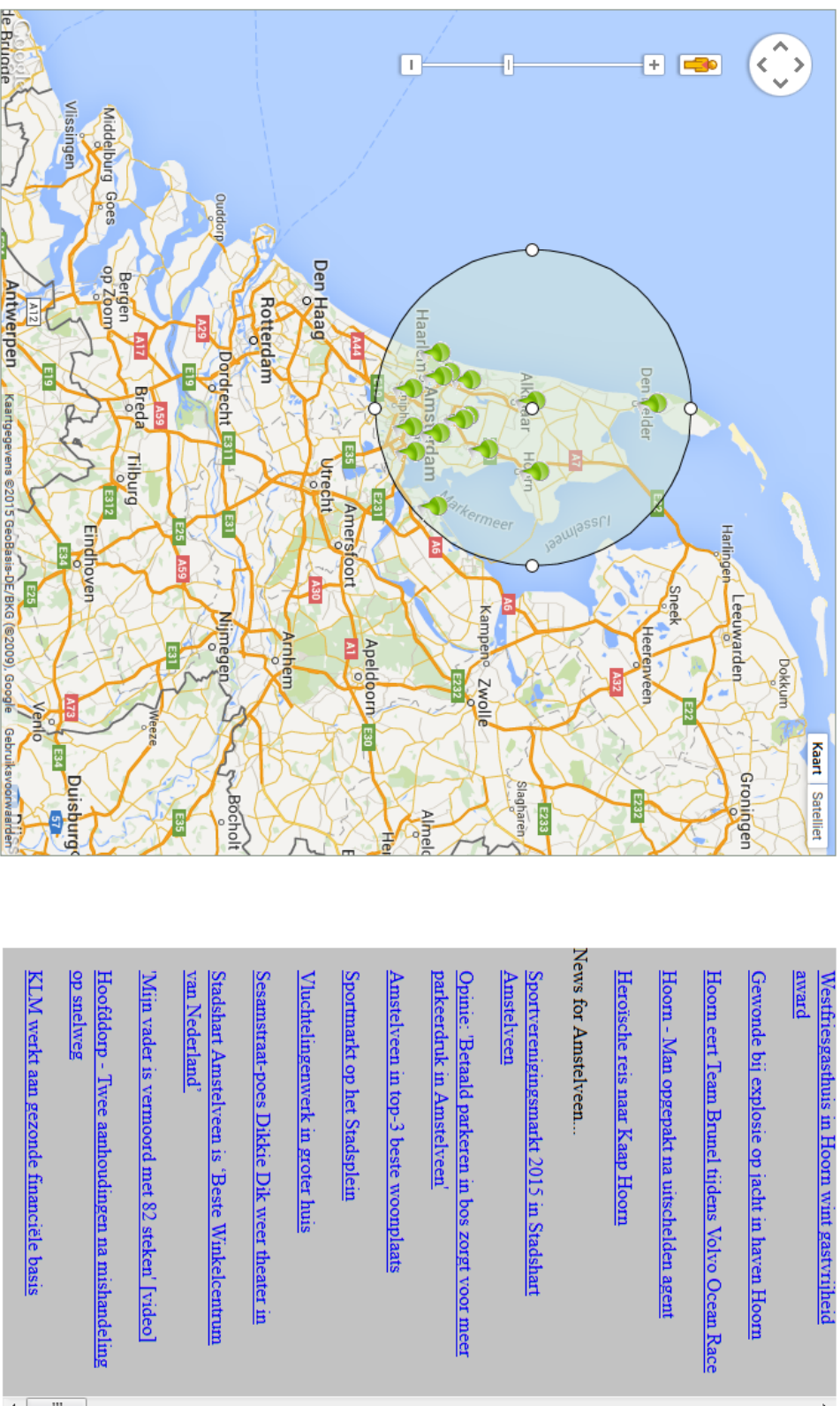
4.5 Samenvatting

In dit hoofdstuk hebben we gekeken hoe we het in het voorgaande hoofdstuk voorgestelde ontwerp kunnen implementeren. We hebben gekeken naar PHP, Ruby en Python uiteindelijk hebben we er voor gekozen om de web applicatie met Python, CherryPy en Cheetah te implementeren.

Ook hebben we gekeken hoe we een gebied kunnen representeren met een beperkt aantal steden. Hiervoor hebben we twee algoritmes besproken.

Met dit hoofdstuk hebben we de vragen *Hoe implementeren we een dergelijke applicatie?* en *Kunnen we een gebied representeren door een beperkt aantal steden? En zo ja, hoe?* beantwoord. In het volgende hoofdstuk zullen we de experimenten die uitgevoerd zijn aan de hand van de hier ontwikkelde applicatie bespreken.

Met deze applicatie kunt u nieuws zoeken door een gebied te selecteren op de kaart. Dit selecteren doet u door een punt te kiezen op de kaart, de linker muisknop ingedrukt te houden en te slepen tot het gebied groot genoeg is. Zodra u de linker muisknop los laat wordt er naar nieuws gezocht. Het is helaas niet mogelijk om de kaart te slepen, als u naar een andere gebied wilt kunt u dit doen door in en uit te zoomen. In en uit zoomen kan door op de kaart naar boven en beneden te scrollen.



Figuur 4.7: De User Interface

Experimenten

Om te weten of de ontwikkelde applicatie echt een toegevoegde waarde heeft moet er een gebruikers onderzoek uitgevoerd worden. Daarnaast is er een onderzoek nodig om een te bepalen hoeveel steden er nodig zijn om een gebied te representeren, en om een keuze te maken uit de twee voorgestelde algoritmes om deze steden te kiezen. Met deze twee onderzoeken kunnen we de volgende twee vragen beantwoorden:

Hoeveel steden zijn er nodig om een gebied te representeren?

Wat is de toegevoegde waarde van het zoeken van nieuwsberichten aan de hand van een locatie?

5.1 Gebiedsrepresentatie

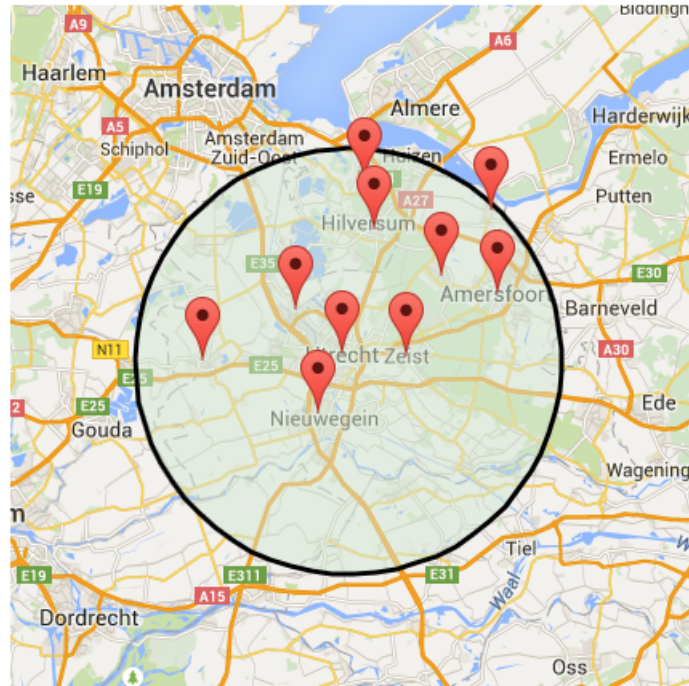
Dit onderzoek is gedaan om te bepalen hoeveel steden er nodig zijn om een gebied goed te representeren. Ook wordt er gekeken welke van de twee voorgestelde algoritmes het beste is om de steden te selecteren die het gebied gaan representeren.

5.1.1 Opzet

Voor dit experiment is een enquête gemaakt waarin de gebruiker een afbeelding van een geselecteerd gebied te zien krijgt. Een voorbeeld hiervan wordt gegeven in figuur 5.1. Bij elke afbeelding krijgt de gebruiker de vraag of dit een goede representatie is van het geselecteerde gebied. Dit is een meerkeuze vraag met als mogelijke antwoorden *ja* en *nee*. Wanneer de gebruiker *nee* kiest krijgt hij of zij een afbeelding van het zelfde gebied te zien maar met meer steden. Er zijn afbeeldingen met tien, vijftien en twintig steden. Zodra er ja gekozen wordt krijgt de gebruiker een vervolg vraag om te bepalen welke van de twee voorgestelde algoritmes beter is. Hiervoor krijgt hij of zij twee afbeeldingen van het zelfde gebied te zien, met het door hem of haar gekozen aantal steden. Een van de afbeeldingen is met algoritme 1 gemaakt terwijl de andere algoritme 2 gebruikt. De vraag is nu welke van de twee representaties beter is, met als mogelijke antwoorden optie 1 of optie 2. Afbeelding 5.2 is hier een voorbeeld van. In totaal zijn er voor vier gebiedsvragen, dit zijn gebieden met een straal van $25km$, $100km$, $500km$ en $1000km$.

Deze enquête is in met Google Docs¹ gemaakt en de resultaten worden in een spreadsheet opgeslagen.

¹<http://docs.google.com/>



Is dit een goede representatie? *

- ☐ Ja
☐ Nee

Figuur 5.1: Voorbeeld vraag gebiedsrepresentatie

5.1.2 Resultaten

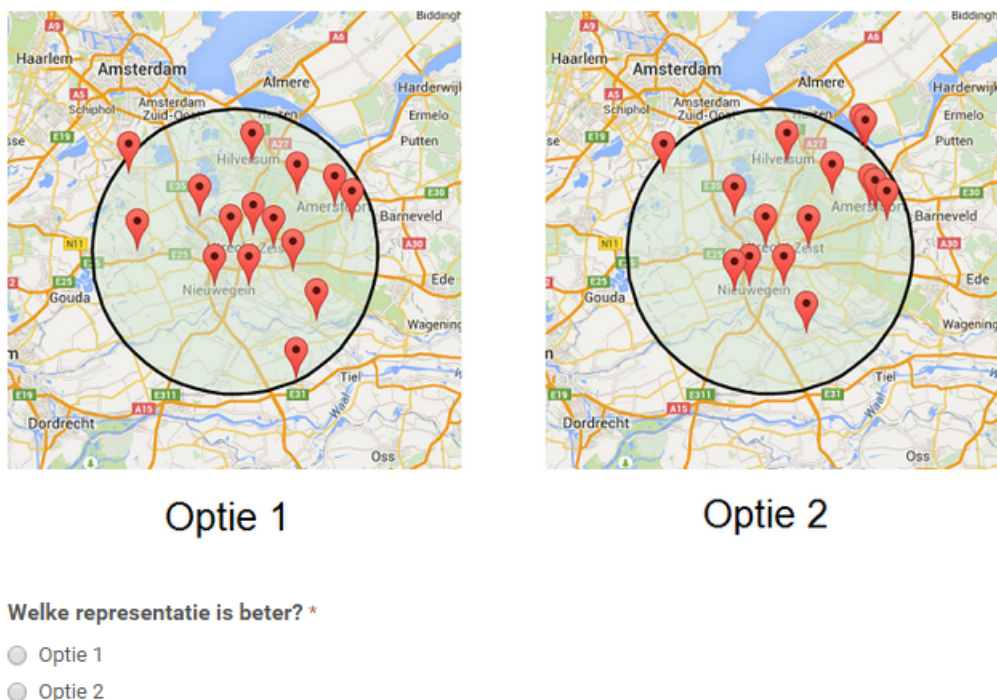
De resultaten zoals deze in het spreadsheet opgeslagen zijn zijn te vinden in bijlage B. Hier onder worden de resultaten per gebied besproken. De enquête is door 26 mensen ingevuld met een leeftijd van 15 tot en met 80 jaar. Er zitten ongeveer evenveel mannen als vrouwen in deze groep met allemaal een verschillend opleidingsniveau. Dit is gedaan om een zo goed mogelijk beeld te krijgen van de hoeveelheid steden die nodig is en wat het beste algoritme is.

Gebied 1 - straal 25km

In tabel 5.1 zijn de resultaten te zien voor een gebied met een straal van 25km. Aan de hand van deze resultaten kunnen we het gemiddelde aantal steden berekenen dat nodig is om dit gebied te representeren: $\frac{11*10+12*15+3*20}{26} \approx 13.46$ steden. Verder is te zien dat $\frac{24}{26} * 100\% \approx 92.31\%$ van de respondenten de representatie gegeven door algoritme 1 beter vindt, $\frac{2}{26} * 100\% \approx 7.69\%$ kiest voor algoritme 2.

Gebied 2 - straal 100km

In tabel 5.2 zijn de resultaten te zien voor een gebied met een straal van 100km. Aan de hand van deze resultaten kunnen we het gemiddelde aantal steden berekenen dat nodig is om dit gebied te representeren: $\frac{8*10+11*15+5*20}{24} \approx 14.38$ steden. Verder is te zien dat $\frac{23}{26} * 100\% \approx 88.46\%$ van de respondenten de representatie gegeven door algoritme 1 beter vindt, $\frac{3}{26} * 100\% \approx 11.54\%$ kiest voor algoritme 2. Er is hier bij het berekenen van het gemiddelde aantal steden dat nodig is voor de weergave van dit gebied gedeeld door 24 omdat twee respondenten geen van de drie representaties goed vonden. Omdat er ook als er op alle drie de vragen nee geantwoord een keuze



Figuur 5.2: Voorbeeld vraag gebiedsrepresentatie

gemaakt moet worden tussen de twee algoritmes is hier wel door 26 gedeeld.

Gebied 3 - straal 500km

In tabel 5.3 zijn de resultaten te zien voor een gebied met een straal van 500km. Ook nu weer kunnen we aan de hand van deze resultaten het gemiddelde aantal steden berekenen dat nodig is om dit gebied te representeren: $\frac{9*10+14*15+3*20}{26} \approx 13.85$ steden. Verder is te zien dat $\frac{24}{26} * 100\% \approx 92.31\%$ van de respondenten de representatie gegeven door algoritme 1 beter vindt, $\frac{2}{26} * 100\% \approx 7.69\%$ kiest voor algoritme 2.

Gebied 4 - straal 1000km

De resultaten voor een gebied met een straal van 1000km zijn te zien in tabel 5.4. Aan de hand van deze resultaten kunnen we weer het gemiddelde aantal steden berekenen dat nodig is om dit gebied te representeren: $\frac{13*10+3*15+9*20}{25} \approx 14.20$ steden. Verder is te zien dat $\frac{24}{26} * 100\% \approx 92.31\%$ van de respondenten de representatie gegeven door algoritme 1 beter vindt, $\frac{2}{26} * 100\% \approx 7.69\%$ kiest voor algoritme 2. Ook hier is door 25 gedeeld bij de berekening van het gemiddelde aantal steden omdat één van de respondenten geen van de drie representaties goed vond.

5.2 Toegevoegde waarde

Om er achter te komen of de ontwikkelde tool toegevoegde waarde heeft en zo ja wat deze toegevoegde waarde is, is er nog een gebruikersonderzoek gedaan. De opzet en resultaten van dit onderzoek zullen in deze sectie besproken worden.

5.2.1 Opzet

Voor dit onderzoek hebben we van te voren een gebieden uitgekozen waar gebruikers naar nieuws moeten zoeken. We hebben de provincie Noord Holland, Nederland uitgekozen. De gebruikers

Kandidaat	10 steden	15 steden	20 steden	Algoritme
1	✗	✓		1
2	✗	✓		1
3	✗	✓		1
4	✓			1
5	✓			1
6	✗	✓		1
7	✗	✓		1
8	✗	✓		1
9	✓			1
10	✓			1
11	✓			1
12	✗	✗	✓	2
13	✓			1
14	✓			1
15	✓			1
16	✗	✓		1
17	✓			1
18	✓			1
19	✗	✓		2
20	✗	✓		1
21	✗	✗	✓	1
22	✗	✓		1
23	✗	✓		1
24	✗	✗	✓	1
25	✓			1
26	✗	✓		1

Tabel 5.1: Resultaten voor een gebied met straal 25km

beginnen met het zoeken naar nieuws door gebruik te maken van de traditionele methodes, nieuws sites, websites van kranten en zoekmachines al Google en BING. Hiervoor krijgen de gebruikers vijf minuten waarin ze zoveel mogelijk, voor hen interessant, nieuws proberen te vinden. Na deze vijf minuten krijgen ze de zelfde opdracht maar nu door gebruik te maken van de ontwikkelde tool. Om te kijken of de tool een toegevoegde waarde heeft gaan we kijken hoeveel nieuws de gebruikers in beide situaties hebben kunnen vinden. Daarnaast vragen we de gebruiker ook om feedback. Bij dit onderzoek is er voor gekozen om een zo breed mogelijk doelgroep te nemen. Dit omdat als we bijvoorbeeld alleen ICT studenten nemen de kans groot is dat zij veel beter dan een gemiddeld persoon weten hoe ze zoekopdrachten bij bijvoorbeeld Google uit moeten voeren.

5.2.2 Resultaten

De resultaten zoals deze in het spreadsheet opgeslagen zijn zijn te vinden in bijlage C. Hier zullen we deze resultaten bespreken. De enquête is door 8 mensen ingevuld met een leeftijd van 20 tot 65 jaar. Er zitten ongeveer even veel vrouwen als mannen in deze groep met allemaal een verschillend opleidingsniveau. Dit is gedaan om een zo goed mogelijk beeld te krijgen of en waarom de applicatie een toegevoegde waarde heeft. In tabel 5.5 zien we de resultaten voor dit onderzoek, we zien hier hoeveel nieuws artikelen de respondent in vijf minuten heeft kunnen vinden door gebruik te maken van de al bestaande middelen zoals nieuwssites en Google News (*# artikelen normaal*), verder zien we de hoeveelheid nieuws artikelen die gevonden zijn door gebruik te maken van de ontwikkelde applicatie (*# artikelen applicatie*) en we zien of de respondent vindt dat de applicatie toegevoegde waarde heeft. Kandidaat nummer 8 heeft bij *# artikelen normaal* en *# artikelen applicatie* een vraagteken staan omdat hier geen aantallen ingevuld zijn. Aan de rest van de resultaten kunnen we zien dat $\frac{7}{8} = 87.50\%$ van de respondenten vindt dat de applicatie een toegevoegde waarde is. Verder zien we dat bij iedereen de hoeveelheid nieuws die

Kandidaat	10 steden	15 steden	20 steden	Algoritme
1	✗	✓		1
2	✗	✓		1
3	✗	✗	✓	1
4	✗	✓		1
5	✗	✗	✓	1
6	✓			1
7	✗	✓		2
8	✓			2
9	✗	✗	✗	1
10	✓			1
11	✓			1
12	✗	✗	✓	1
13	✓			1
14	✗	✗	✗	1
15	✗	✓		1
16	✓			1
17	✗	✓		1
18	✓			2
19	✗	✓		1
20	✗	✓		1
21	✗	✓		1
22	✗	✗	✓	1
23	✗	✗	✓	1
24	✗	✓		1
25	✓			1
26	✗	✓		1

Tabel 5.2: Resultaten voor een gebied met straal 100km

Kandidaat	10 steden	15 steden	20 steden	Algoritme
1	✗	✓		1
2	✗	✓		1
3	✗	✗	✓	1
4	✗	✓		1
5	✗	✗	✓	1
6	✓			1
7	✗	✓		1
8	✓			1
9	✓			1
10	✓			1
11	✗	✓		1
12	✗	✗	✓	1
13	✗	✓		1
14	✓			1
15	✗	✓		2
16	✗	✓		1
17	✓			1
18	✗	✓		1
19	✓			1
20	✓			1
21	✓			1
22	✗	✓		1
23	✗	✓		1
24	✗	✓		2
25	✗	✓		1
26	✗	✓		1

Tabel 5.3: Resultaten voor een gebied met straal 500km

Kandidaat	10 steden	15 steden	20 steden	Algoritme
1	✗	✗	✓	1
2	✗	✓		1
3	✗	✗	✓	2
4	✗	✓		1
5	✗	✗	✓	1
6	✓			1
7	✓			1
8	✓			1
9	✓			1
10	✓			1
11	✓			1
12	✗	✗	✓	1
13	✓			1
14	✓			1
15	✓			1
16	✗	✗	✓	1
17	✓			1
18	✗	✗	✓	1
19	✓			1
20	✓			1
21	✗	✗	✓	1
22	✗	✗	✓	1
23	✗	✗	✓	1
24	✗	✗	✗	2
25	✓			1
26	✗	✓		1

Tabel 5.4: Resultaten voor een gebied met straal 1000km

Kandidaat	# artikelen normaal	# artikelen applicatie	Toegevoegde waarde
1	3	9	Ja
2	4	5	Nee
3	5	8	Ja
4	11	27	Ja
5	10	25	Ja
6	4	7	Ja
7	11	15	Ja
8	?	?	Ja

Tabel 5.5: Resultaten onderzoek toegevoegde waarde

ze kunnen vinden in vijf minuten tijd is toegenomen, gemiddeld is deze toename 98.71%.

Discussie en toekomstig onderzoek

6.1 Discussie

Aan de resultaten van het onderzoek naar de gebiedsrepresentatie is te zien dat het, gemiddeld genomen, niet veel uit maakt. Voor een gebied met een straal van $25km$ zijn gemiddeld 13.46 steden nodig, voor een groter gebied, met straal $1000km$ is dit gemiddeld 14.20 steden. Wel is het opvallend dat het gebied met een straal van $100km$ gemiddeld 14.38 steden nodig heeft.

Het lijkt er op dat het ideale aantal steden afhangt van de grootte van het gebied, we zien dat bij een gebied met een straal van $25km$ 3 mensen twintig steden voldoende vinden, bij $100km$ is zijn dit 5 mensen, maar ook 2 mensen die geen van de drie geboden representaties voldoende vinden. Kijken we naar een gebied met een straal van $500km$ dan vinden 3 mensen twintig steden voldoende en bij het grootste gebied, met een straal van $1000km$, zijn dit 9 mensen en 1 die geen van de drie representaties goed vindt. Er is dus te zien dat bij een groter gebied meer respondenten voor de representatie met 20 steden kiezen, al is dit bij een gebied met een straal van $500km$ weer lager. Dit zou kunnen komen omdat het ideale aantal steden niet alleen afhangt van de straal van het gebied maar ook van waar dit gebied gekozen wordt. Om dit te onderzoeken kunnen we de enquête uitbreiden door meerdere regio's aan te bieden met de zelfde straal.

In dit onderzoek is ook gekeken welke van de twee voorgestelde algoritmes, zie sectie 4.3, een betere representatie geeft. Bij de gebieden met een straal van $25km$, $500km$ en $1000km$ kiest 92.31% van de respondenten voor algoritme 1, waarbij een gebied steeds in vier delen gesplitst wordt. Bij het gebied met een straal van $100km$ is dit percentage met 88.46% iets lager. We zien dus dat algoritme 1 een beter representatie van een gebied geeft.

Aan de resultaten van het onderzoek naar de toegevoegde waarde kunnen we zien dat een dergelijke applicatie voor een groot deel, 87.50% een toegevoegde waarde heeft. Wel moet hierbij opgemerkt worden dat de groep respondenten te klein is om een echte conclusie te trekken. Ook zien we dat de hoeveelheid nieuws die in vijf minuten gevonden kan worden bij alle respondenten is toegenomen, hier zijn wel grote verschillen te zien. Deze toename loopt van 25% tot 200%.

We hebben niet alleen gevraagd of de applicatie een toegevoegde waarde heeft maar ook waarom wel of niet. Hoewel de reactie hier natuurlijk veel kunnen verschillen valt op dat 6 respondenten vinden dat het een toegevoegde waarde heeft omdat ze sneller en eenvoudiger nieuws (uit een regio) kunnen vinden. Van de respondenten vindt één dat de applicatie geen toegevoegde waarde heeft omdat er niet op subcategoriën gezocht kan worden, bij andere respondenten zien we dit terug als een van de verbeterpunten. Voor toekomstig onderzoek zou het dus een interessante optie zijn om naast het selecteren van een gebied ook een onderwerp op te kunnen geven. Op deze manier wordt het mogelijk om bijvoorbeeld naar overvallen in Noord Holland te zoeken.

6.2 Toekomstig onderzoek

Om de applicatie te verbeteren moet er bijvoorbeeld gekeken worden naar het gebruik van andere nieuws bronnen dan zoekmachines en nieuws sites. Denk hierbij aan bijvoorbeeld twitter data, maar ook kan het interessant zijn om naar lokale nieuws bronnen te gaan kijken.

6.2.1 Steden met onbekende populatie

Het probleem van de steden waarvan geen populatie bekend is, zoals beschreven in sectie 3.5, is op te lossen door op bijvoorbeeld Wikipedia¹ of WolframAlpha² de populatie op te zoeken. Dit kan gedaan worden door de pagina die over de stad gaat te downloaden en hier vervolgens de populatie uit te halen. We zouden dit kunnen doen op het moment dat een stad geselecteerd worden met populatie 0, een beter mogelijkheid is waarschijnlijk om dit te doen zodra de database opgebouwd wordt.

6.2.2 Dubbele steden

Bij het algoritme voor de gebiedsrepresentatie worden af en toe twee steden heel dicht bij elkaar gekozen, in sommige gevallen gaat dit zelfs om de zelfde stad. Dit komt omdat een aantal grote steden in meerdere delen in de database staan. Een voorbeeld hiervan is *Londen* deze stad staat er als *London* en *City of London* in. Er zijn twee manieren om dit op te lossen.

Ten eerste kan er gekeken worden of deze steden samen gevoegd kunnen worden. Bij het opbouwen van de database moet dan gekeken worden of er al een stad op dezelfde plek ligt, als dit het geval is wordt de populatie van de tweede stad bij de eerste opgeteld en slaan we dus maar één stad op.

De tweede optie is om het algoritme voor de gebiedsrepresentatie zo aan te passen dat niet twee steden vlak bij elkaar gekozen kunnen worden, ook al zijn dit misschien de grootste steden. Hiervoor moet je voor elke stad die je kiest kijken hoe dicht ze bij de al eerder gekozen steden liggen. Als deze afstand kleiner is dan een minimale afstand moet er een andere stad gekozen worden.

6.2.3 Lokaal nieuws - voorstel

Een interessante optie voor de toekomst is om te kijken of het mogelijk is om de nieuwsbronnen af te laten hangen van het geselecteerde gebied, dat wil zeggen dat er gebruik gemaakt wordt van de lokale nieuwsbronnen. Om te bepalen of we te maken hebben met een lokale nieuwsbron kunnen we om te beginnen kijken naar het top-level domain [37], ook kan er gekeken worden naar de WHOIS [36] informatie. Hiermee kan al een aardige schatting gemaakt worden bij welk land de nieuwssite hoort. Om te bepalen bij welke steden de site hoort kunnen we kijken naar bijvoorbeeld de homepage van de nieuwssite. Deze kunnen we analyseren, we kunnen alle namen van steden hier uit halen, als deze grotendeels bij elkaar liggen is de kans groot dat het om een lokale nieuwsbron gaat. Liggen de genoemde steden op de homepage ver uit elkaar, dan is het waarschijnlijk een minder lokale bron.

¹<https://en.wikipedia.org/>

²<http://www.wolframalpha.com/>

Conclusie

Op dit moment is het nog niet (eenvoudig) mogelijk om nieuws te zoeken aan de hand van een locatie, dit zou wel heel nuttig kunnen zijn omdat gebruikers hierdoor een beter begrip van nieuws krijgen. In deze scriptie hebben we gekeken of een applicatie waarmee het mogelijk is om nieuws te zoeken aan de hand van een selectie op een kaart toegevoegde waarde heeft. Om dit te onderzoeken is eerst gekeken hoe we een dergelijke applicatie kunnen opbouwen, er is gekeken welke nieuwsbronnen we kunnen gebruiken en of we een gebied kunnen representeren met een beperkt aantal steden. Voor deze representatie zijn twee algoritmes ontworpen waarvan we de beste aan de hand van een gebruikersonderzoek bepaald hebben.

Het onderzoek naar de gebiedsrepresentatie heeft laten zien dat het aantal steden niet heel veel af hangt van de grootte van een gebied maar dat het ideaal aantal steden wel iets toeneemt naarmate het gebied groter wordt. In het tweede onderzoek is gekeken naar de toegevoegde waarde van de applicatie, hieruit blijkt dat een dergelijke applicatie zeker van toegevoegde waarde kan zijn.

Deze scriptie heeft bewezen dat een applicatie waarmee nieuws gezocht kan worden aan de hand van een locatie van toegevoegde waarde is. Door een gebied op de kaart te selecteren wordt het eenvoudiger en sneller om nieuws uit een bepaalde regio te vinden. Hopelijk wordt er in de toekomst meer onderzoek gedaan naar deze manier van nieuws zoeken zodat dit voor iedereen beschikbaar komt.

Bibliografie

- [1] Benjamin E. Teiler, Micheal D. Lieberman, Daniele Panozzo, Jagan Sankaranarayanan, Hanan Samet and Jon Sperling. *NewsStand: A New View on News*. In Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS 2008), IRVINE, CA, November 2008
- [2] Micheal D. Lieberman, Hanan Samet, Jagan Sankaranarayanan and Jon Sperling. *STEWART: Architecture of a Spatio-Textual Search Engine* 15th ACM GIS, Seattle, WA, November 2007
- [3] Hanan Samet, Jagan Sankaranarayanan, Micheal D. Lieberman, Marco D. Adelfio, Brendan C. Fruin, Jack M. Lotkowski, Daniele Panozzo, Jon Sperling and Benjamin E. Teiler. *Reading News with Maps by Exploiting Spatial Synonyms* Communications of the ACM, October 2014
- [4] Einat Amitay, Nadav Har'El, Ron Sivan, Aya Soffer. *Web-a-Where: Geotagging Web Content*. SIGIR'04, July 25-29, 2004, Sheffield, South Yorkshire, UK
- [5] Jochen L. Leidner. *Toponym Resolution in Text: "Which Sheffield is it?"*. SIGIR 2004 Sheffield, UK
- [6] Wikipedia. *Geographical distance* URL: https://en.wikipedia.org/wiki/Geographical_distance (april 2015)
- [7] Jan Philip Matuschek. *Finding Points Within a Distance of a Latitude/Longitude Using Bounding Coordinates*. URL: <http://janmatuschek.de/LatitudeLongitudeBoundingCoordinates> (april 2015)
- [8] Movable Type Scripts. *Calculate distance, bearing and more between Latitude/Longitude points*. URL: <http://www.movable-type.co.uk/scripts/latlong.html> (april 2015)
- [9] Google. *Google Maps JavaScript API*. URL: <https://developers.google.com/maps/documentation/javascript/tutorial> (april 2015)
- [10] Google. *News Search (Deprecated)*. URL: <https://developers.google.com/news-search/> (mei 2015)
- [11] BING. *Bing Search API*. URL: <http://datamarket.azure.com/dataset/bing/search> (mei 2015)
- [12] Yahoo. *News Service*. URL: https://developer.yahoo.com/boss/search/boss_api_guide/news.html (mei 2015)
- [13] Faroo. *Free API*. URL: <http://www.faroo.com/hp/api/api.html> (mei 2015)
- [14] *SQLite*. URL: <https://www.sqlite.org/> (april 2015)
- [15] Laurens Verspeek. *Trusting websites using geo-graphical consistency*. June 20, 2014
- [16] Python. *CherryPy*. URL: <http://www.cherrypy.org/> (april 2015)

- [17] Python. *Cheetah*. URL: <http://www.cheetahtemplate.org/> (april 2015)
- [18] JSON. URL: <http://json.org/> (april 2015)
- [19] AJAX. URL: [http://en.wikipedia.org/wiki/Ajax_\(programming\)](http://en.wikipedia.org/wiki/Ajax_(programming)) (april 2015)
- [20] Geonames. URL: <http://www.geonames.org/> (april 2015)
- [21] JQuery. URL: <https://jquery.com/> (april 2015)
- [22] RSS Feeds. URL: <http://en.wikipedia.org/wiki/RSS> (mei 2015)
- [23] Python. *sqlite3*. URL: <https://docs.python.org/2/library/sqlite3.html> (april 2015)
- [24] Python. *zipfile*. URL: <https://docs.python.org/2/library/zipfile.html> (april 2015)
- [25] Python. *urllib2*. URL: <https://docs.python.org/2/library/urllib2.html> (april 2015)
- [26] Wikipedia. *Spherical law of cosines*. URL: https://en.wikipedia.org/wiki/Spherical_law_of_cosines (april 2015)
- [27] Wikipedia. *Haversine Formula*. URL: https://en.wikipedia.org/wiki/Haversine_formula (april 2015)
- [28] Wikipedia. *Circle of latitude*. URL: https://en.wikipedia.org/wiki/Circle_of_latitude (april 2015)
- [29] Wikipedia. *Circle of a sphere*. URL: https://en.wikipedia.org/wiki/Circle_of_a_sphere (april 2015)
- [30] Wikipedia. *Great circle*. URL: https://en.wikipedia.org/wiki/Great_circle (april 2015)
- [31] WolframAlpha. URL: <http://www.wolframalpha.com/> (april 2015)
- [32] Wikipedia. *Hypertext Transfer Protocol*. URL: http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol (mei 2015)
- [33] Wikipedia. *Application programming interface*. URL: http://en.wikipedia.org/wiki/Application_programming_interface (mei 2015)
- [34] Wikipedia. *SQL*. URL: <http://en.wikipedia.org/wiki/SQL> (april 2015)
- [35] Wikipedia. *Pythagorean theorem*. URL: http://en.wikipedia.org/wiki/Pythagorean_theorem (april 2015)
- [36] Wikipedia. *WHOIS*. URL: <https://en.wikipedia.org/wiki/WHOIS> (juni 2015)
- [37] Wikipedia. *Domain name, Top-level domains* URL: https://en.wikipedia.org/wiki/Domain_name#Top-level_domains (juni 2015)
- [38] W. Gellert, S. Gottwald, M. Hellwich, H. Kstner, and H. Kstner. *The VNR Concise Encyclopedia of Mathematics*. 2nd ed., ch. 12 (Van Nostrand Reinhold: New York, 1989).

Installatie

A.1 Systeem vereisten

- Linux server (bijvoorbeeld Ubuntu Server)
- Python 2.7
- CherryPy
- Cheetah
- sqlite3 (Python module)
- SQLite
- Minimaal 7MB vrije ruimte op HDD

A.2 Toevoegen extensies aan SQLite

Om extensies aan SQLite toe te kunnen voegen moeten we eerste het pakket *libsqlite3-dev* installeren. Op een Ubuntu Server gaat dat als volgt:

```
sudo apt-get install libsqlite3-dev
```

Vervolgens moeten we het C bestand met de extensie compileren:

```
gcc -fPIC -lm -shared extension-functions.c -o libsqlitefunctions.so
```

A.3 Server installatie

Om de server te installeren moeten we deze downloaden van de volgende link:

<https://github.com/thomasvanophem/Afstudeerproject/tree/master/Source/Server>

Deze bestanden moeten allemaal in de zelfde map op de server geplaatst worden.

A.4 Server starten

De server starten gaat met het volgende commando:

```
python server.py
```

De server wordt opgestart en is nu te vinden via `http://<ip adres server>:8080`, bijvoorbeeld:

`http://192.168.0.13:8080`

De eerste keer dat de server gestart wordt moet de database opgebouwd worden, dit gebeurt door de variable *download* in *config.py* op *True* te zetten. Het is aan te raden om hierna deze variable op *False* te zetten zodat de database niet elke keer opgebouwd wordt als de server herstart wordt.

Resultaten onderzoek gebiedsrepresentatie

[illegible]

Resultaten onderzoek toegevoegde waarde

sex	Man
age	22
level	WO
# normal	3
sources	Noord Hollands Dagblad
# application	9
added value	Ja
why	Makkelijk en sneller om nieuws te vinden. Een duidelijk overzicht
improvements	Ook de mogelijkheid bieden om een onderwerp op te geven.

sex	Vrouw
age	18
level	MBO
# normal	4
sources	Google, dichtbij
# application	5
added value	Nee
why	Hij gaat niet op subcategorieën.
improvements	Via de telefoon werk hij niet zo snel. Layout kan iets mooier

sex	Man
age	61
level	Voortgezet onderwijs
# normal	5
sources	Telegraaf noordhollands dagblad nieuws .nl
# application	8
added value	Ja
why	regio selectie per plaats naam uitzoeken sneller over zicht van het nieuws door korte inhoud z.g koppensnellen hierdoor voor mij interessanten nieuws
improvements	Is het mogelijk op onderwerp te zoeken?

sex	Man
age	23
level	WO
# normal	11
sources	Google news rtvnh.nl noordhollandsdagblad.nl dichtbij.nl nu.nl parool.nl
# application	27
added value	Ja
why	Eenvoudig en snel nieuws vinden uit een regio overzichtelijk weergegeven
improvements	Zoeken op onderwerp (dus gebied selecteren maar ook onderwerp, bijvoorbeeld; schietpartij noord holland

sex	Vrouw
age	54
level	HBO
# normal	10
sources	Noord hollands dagblad, Nu.nl
# application	25
added value	Ja
why	verzameling onderwerpen staat handig onder elkaar vermeld.
improvements	bronvermelding bij onderwerp. datum van artikel.

sex	Man
age	25
level	WO
# normal	4
sources	google news, rtvnh.nl
# application	7
added value	Ja
why	Het nieuws is accurater en de applicatie verzamelt berichten uit een groot aantal bronnen. Het is makkelijk om een snel een overzicht te krijgen van al het nieuws uit een regio.
improvements	De manier van het selecteren van een regio zou makkelijker kunnen. Het bewegen over de kaart en het slepen om een cirkel te vormen is niet 100% intuïtief.

sex	Man
age	22
level	WO
# normal	11
sources	google news, rtvnh.nl
# application	15
added value	Ja
why	Een handige manier om lokaal nieuws op te zoeken, zonder daarvoor verschillende nieuwswebsites te hoeven bekijken. Een voordeel is dat de applicatie snel reageert op een nieuw geselecteerd gebied, dat draagt bij aan het gebruiksgemak en de totale ervaring.
improvements	<ul style="list-style-type: none"> - Mogelijk maken om huidige selectie aan te passen - nu wordt direct een nieuwe selectie getekend. - Preview van nieuwsitems weergeven in het nieuwsoverzicht aan de rechterkant. - Afbeeldingen bij nieuwsberichten ophalen en tonen in het nieuwsoverzicht (als het bericht een afbeelding bevat). - Melding geven als er geen nieuwsberichten zijn gevonden.

sex	Vrouw
age	23
level	MBO
# normal	?
sources	Telegraaf
# application	?
added value	Ja
why	Ik snap er niks van van de onderzoek... Als ik het goed begrijp word er onderzocht wat voor nieuws de mensen meer aanspreken onder bepaalde leeftijden.”
improvements	Duidelijkheid waar dit precies over gaat