Project Heatmap

Onderzoeksrapport v0.5

11-03-11

Dennis Wagenaar

Inhoudsopgave

Inleiding

Dit onderzoeksrapport moet zicht geven op de mogelijkheden die er zijn tot het creëren van een heatmap. Er wordt per mogelijkheid kort uitgelegd hoe deze werkt, en de voor- en nadelen worden beschreven.

Daarnaast worden er meerdere opties voorgelegd voor de visualisatie van de heatmap.

Hieruit volgt een advies welke van de mogelijkheden het beste gebruikt kan worden voor het project.

Gheat

Website
http://code.google.com/p/gheat/

Voorbeeld
http://vort.org/media/data/crashes.html

Info

Met Gheat is het mogelijk om een heatmap te genereren voor Google Maps. Gheat draait onder de python webserver *Aspen*, en maakt gebruik van de imaging library *PIL* of *Pygame*. Wanneer een gebruiker de map bezoekt zal de server de gheat bestanden aanroepen die gebruik maken van een database bestand en de imaging library om afbeeldingen te genereren. Deze afbeeldingen worden als overlay gebruik op Google Maps om zo een heatmap te tonen.

Voordelen

Het voordeel is dat gheat opensource is, en naar wens kan worden aangepast, al is er wel kennis van python vereist. Ook is de installatie is vrij goed gedocumenteerd. Daarnaast is het resultaat grafisch vrij mooi.

Nadelen

De nadelen zijn dat het om een overlay met afbeeldingen gaat. Op elk zoomniveau moeten er nieuwe afbeeldingen worden gemaakt (afbeeldingen worden overigens alleen gemaakt van plekken waar werkelijk data te zien zou zijn, voor lege plekken wordt een lege afbeelding aangehouden). Stel er wordt per week een nieuwe heatmap gemaakt, dan zal de eerste gebruiker van die week een langere laadtijd hebben dan normaal, omdat hij de trigger is voor het maken van de afbeeldingen. Wel kan er gebruik worden gemaakt van een script dat de afbeeldingen maakt (standaard script hiervoor kan uren duren, aangepast script op locatie zal sneller gaan). Daarnaast kunnen de afbeeldingen behoorlijk in aantal oplopen, tot honderden per week (even aangenomen dat trending per week zal gaan).

Een variant van gheat is django-gheat, waarbij gebruik wordt gemaakt van het Django web framework waarin het o.a. mogelijk is om makkelijk connectie te maken met een SQL database.

Google Fusion Tables

Website
http://code.google.com/apis/fusiontables/

Voorbeeld
http://code.google.com/apis/maps/documentation/javascript/examples/layer-fusiontables-heatmap.html

Info

Met de Google Fusion Tables API kan een programma of script data uitlezen of wegschrijven van/naar Google Fusion Tables. Deze tables kunnen vervolgens worden gevisualiseerd in een paar vormen, waaronder een map met puntjes van elke entry, en een heatmap.

Voordelen

Het voordeel is dat de gebruiker de map volledig van de Google servers haalt, wat minder belastend zal zijn voor de eigen server omdat de database dus bij Google ligt. Ook is het vrij simpel op te zetten, en kan er makkelijk een andere visualisatie worden gekozen zoals een lijngrafiek of tijdlijn. Wel zal er een script moeten worden geschreven wat de data wegschrijft naar de Fusion Table.

Nadelen

Het nadeel is dat het allemaal vrij gesloten is. De tabels kan je vrij aanpassen, en de visualisatie naar een map met puntjes werkt goed, maar de hotspots op de heatmap verdwijnen als er dicht wordt ingezoomed op een bepaalde plek. Wordt er ingezoomed op Leiden, dan zal er dus niks te zien zijn, en voor zover ik heb kunnen vinden is dit nog niet aan te passen. Wel moet rekening worden gehouden met het feit Fusion Tables nog in een experimentele fase zit, later zullen er wellicht meer mogelijkheden zijn om dingen naar wens aan te passen.

OLHeatmap

Website
http://olheatmap.sourceforge.net/

Voorbeeld
http://blog.felipebarriga.cl/olheatmap_demo/

Info

In tegenstelling tot de eerder beschreven mogelijkheden gebruikt OLHeatmap geen Google Maps, maar OpenLayers met OpenStreetMap. In het voorbeeld worden de features opgehaald door middel van Xapi, waarna op een canvas op de locaties van de features gekleurde cirkels worden getekend om uiteindelijk een heatmap te vormen.

Voordelen

De voordelen zijn dat het opensource is, en dat het gebruik maakt van OpenStreetMap. Waar Google Maps eigendom blijft van Google, is OpenStreetMap eigendom van de gebruikers. De heatmap zelf is dus vrij aan te passen.

Nadelen

Het nadeel is dat OLHeatmap nog erg ruw en onafgewerkt is. Het voorbeeld werkte bij mij maar 2 of 3 keer, en er is vrij weinig documentatie over te vinden. De keren dat het werkte gaf het wel een mooi resultaat.

OpenLayers Heatmap

Website
http://trac.osgeo.org/openlayers/wiki/Future/OpenLayersWithCanvas#Evaluation

Voorbeeld
http://dev.openlayers.org/sandbox/camptocamp/canvas/openlayers/examples/heatMap.html

Info

Net als OLHeatmap (wat eigenlijk een afkorting is voor OpenLayers Heatmap), maakt dit ook gebruik van OpenLayers. In het geval van het voorbeeld worden er random punten aangemaakt, waarover met een canvas cirkels worden getekend om zo een heatmap creëren. Door een layer als OpenStreetMap toe te voegen kunnen mappen met meer detail worden weergeven.

Voordelen

Voordelen zijn dat het vrij makkelijk is op te zetten. Verder is het opensource en vrij goed gedocumenteerd, zowel een goede wiki als notes in de voorbeeldcode. Daarnaast is het goed aan te passen en uit te breiden.

Nadelen

Het voelt allemaal nog erg basic aan.

The definitive heat map – with Java

Website:
http://www.itstud.chalmers.se/~liesen/heatmap/

Voorbeeld:
http://www.itstud.chalmers.se/~liesen/heatmap/webstart/heatmap.jnlp

Info

Een versie van The definitive heatmap, in eerste instantie bedoeld om heatmaps te maken van websites om zo te zien waar gebruikers het meeste klikken. In het voorbeeld van deze Java versie kan op de achtergrondafbeelding worden geklikt, waarna een hotspot op de geklikte plaats verschijnt.

Voordelen

De maker van deze versie heeft deze aanpassing gemaakt omdat zijn opdracht hetzelfde idee had; maak een heatmap van de wifi sterkte en dekking. Bij hem ging het dan om een gebouw, en hier praten we over een hele stad, maar het zou wel mogelijk moeten zijn. Daarnaast is het opensource en vrij te gebruiken en aan te passen, en het ziet er best mooi uit.

Nadelen

Waar al het bovenstaande met Google Maps of OpenLayers werkt, werkt dit met een achtergrondafbeelding. Ergens moet dus een kaart vandaan worden gehaald. Ook zal er een interface moeten worden gemaakt, OL en Gmaps hebben dit al.

Visualisatie

Naast de techniek achter de visualisatie moet ook de visualisatie zelf onderzocht worden. Het moet duidelijk worden voor de gebruiker waar hij wel en niet zal kunnen internetten, en dit kan op verschillende manieren worden uitgevoerd.

Factoren

Om te beginnen zijn er een aantal factoren waar rekening mee gehouden moet worden:

- Bereik signaal node
- Locatie node
- Bereik ontvangst gebruiker
- Locatie gebruiker
- Omgeving
- Tijd

Om een beeld te schetsen;
De heatmap geeft aan dat er naast gebouw A wifi beschikbaar is. Dit wordt even gevisualiseerd door een cirkel met een diameter van 20 meter op de plek waar dit netwerk eerder is ontvangen.

De gebruiker gaat met zijn laptop naar die locatie, maar ontvangt daar geen wifi. Dit kan veroorzaakt worden door één of meer van de bovenstaande factoren. Zo kan de signaal sterkte van de node te zwak zijn, dit zou dan aangeven dat deze node eerder is ontvangen door iemand met een sterkere netwerkkaart/dongle/oid. Het zou ook kunnen dat de gebruiker op een iets andere positie staat, of dat de omgeving is veranderd: misschien zitten er meer obstakels tussen de gebruiker en de node waardoor het signaal niet bij de gebruiker kan aankomen. Of misschien is de node in de tussentijd wel weggehaald door de eigenaar.

Voor de visualisatie zal er rekening moeten worden gehouden met deze factoren en deze zo mogelijk op te lossen.

Uitvoering

Een applicatie als Gheat gebruikt een set statische cirkels. Op elke locatie uit de database wordt zo'n cirkel getekend. De gebruiker zal aannemen dat hij binnen deze cirkel bereik heeft, terwijl het bereik vanaf het meetpunt misschien niet verder komt dan 10 meter.

Toch zijn cirkels als basis geen slecht idee. Ze geven een beetje een idee aan de gebruiker waar hij terecht kan. Er zou ook voor puntjes kunnen worden gekozen, maar cirkels zijn misschien duidelijker.

Er zou ook gekozen kunnen worden voor cirkels waarvan de diameter wordt gekoppeld aan de signaalsterkte op die meetplek. Is er bijvoorbeeld een node met een sterkte van 25% gevonden, dan zal de diameter van de cirkel 25 meter worden.

Ook zou er een extra layer kunnen worden toegevoegd waarop de nodes worden weergeven. De locaties van de nodes zijn bekend, en als er een cirkel wordt getekend met een straal vanaf de node tot het meest ver gemeten punt kan het de gebruiker een beter beeld geven van de netwerken.

Eerste indruk

Met oog op alle voor- en nadelen lijkt een oplossing als OLHeatmap het beste; een combinatie van OpenLayers met OpenStreetMap en een script dat alle gegevens door middel van een canvas als heatmap laat zien.

OpenLayers en OpenStreetMap zijn beiden onder een opensource licentie te verkrijgen, in tegenstelling tot Google maps. Dit zal wat meer vrijheid geven in het verbruik. Ook OLHeatmap is onder een opensource licentie vrijgegeven, wat het dus mogelijk maakt delen van de code te gebruiken en aan te passen naar wens.

De manier waarop OLHeatmap zijn map presenteerd (even niet gelet op het controlpanel rechts) heeft aardige potentie, en de eisen van dit project zouden daar allemaal verwerkt in kunnen worden.

Conclusie

De conclusie volgt na het uitproberen van alle 5 de opties.