2014•2015
FACULTEIT BEDRIJFSECONOMISCHE WETENSCHAPPEN
master in de toegepaste economische wetenschappen:
handelsingenieur in de beleidsinformatica

Masterproef

Business Intelligence voor academische onderzoekers

Promotor :
Prof. dr. Benoit DEPAIRE

Sofie Theunissen

Scriptie ingediend tot het behalen van de graad van master in de toegepaste economische wetenschappen: handelsingenieur in de beleidsinformatica

universiteit hasselt
KNOWLEDGE IN ACTION

Universiteit Hasselt | Campus Hasselt | Martelarenlaan 42 | BE-3500 Hasselt
Universiteit Hasselt | Campus Diepenbeek | Agoralaan Gebouw D | BE-3590 Diepenbeek

universiteit hasselt
KNOWLEDGE IN ACTION

2014•2015
FACULTEIT BEDRIJFSECONOMISCHE WETENSCHAPPEN
master in de toegepaste economische wetenschappen: handelsingenieur in de beleidsinformatica

Masterproef

Business Intelligence voor academische onderzoekers

Promotor :
Prof. dr. Benoit DEPAIRE

Sofie Theunissen
Scriptie ingediend tot het behalen van de graad van master in de toegepaste economische wetenschappen: handelsingenieur in de beleidsinformatica

universiteit hasselt
KNOWLEDGE IN ACTION

Woord vooraf

Voor deze thesis heb ik mogen onderzoeken in hoeverre een tool, door gebruik te maken van business intelligence, academici kan helpen bij het uitvoeren van een literatuuronderzoek. Dit was een boeiende en uitdagende opgave waarbij verschillende aspecten aan bod kwamen. Zo heb ik ervaring kunnen opdoen bij het bestuderen van bestaande literatuur, het afnemen van interviews, het analyseren van de gebruikersbehoeften, het effectief programmeren van een werkend prototype en het uitschrijven van de tekst van de thesis. Mogelijk kan deze thesis een beter inzicht verschaffen over business intelligence in het algemeen en de mogelijkheden die business intelligence te bieden heeft binnen het kader van literatuuronderzoek.

Mijn dank gaat uit naar prof. dr. Benoit Depaire die mij vakkundig heeft begeleid bij al deze factoren. Hij heeft me goede aanwijzingen gegeven en me indien nodig in de juiste richting gestuurd. Ik wil hem ook bedanken voor het nauwgezet nalezen van mijn ingediende teksten. Verder wil ik de professoren en doctoraatsstudenten die hun medewerking verleend hebben, bedanken voor hun bereidwillige samenwerking, het geven van interviews en het evalueren van het opgeleverde prototype.

Samenvatting

Academische onderzoekers maken voor hun literatuuronderzoek gebruik van het web en van zoekmachines voor het opzoeken van wetenschappelijke artikels omtrent een onderwerp. Het aantal resultaten dat men terugkrijgt is meestal erg hoog en niet alle gevonden artikels zijn even relevant. Mogelijk kan een business intelligence tool zinvolle informatie aanleveren over een welbepaald onderwerp, informatie die de selectie van relevante informatie kan vereenvoudigen. Er zal onderzocht worden op welke manier een BI-tool academici kan ondersteunen om een literatuuronderzoek efficiënter en effectiever te laten verlopen. Het onderzoek wordt uitgevoerd volgens de methode van design science theorie. Dit betekent dat er een prototype van een BI-tool wordt ontwikkeld om de effectiviteit te kunnen evalueren.

De studie van de bestaande literatuur omtrent het uitvoeren van een literatuuronderzoek toont aan dat een auteur vaak niet voldoende tijd besteedt om de beste auteurs en bronnen te identificeren in verband met zijn onderwerp en dat de belangrijkste bijdrages vaak gevonden worden in de voornaamste journals. Uit afgenomen interviews van onderzoekers kwam naar voren dat een overzicht van aanverwante onderwerpen, de evolutie van de onderzoeksinteresse in een onderwerp en een lijst van de belangrijkste journals gezien werden als belangrijke hulpmiddelen om relevante artikels te vinden.

Onderzoek van de geschreven literatuur omtrent business intelligence maakt duidelijk dat het te ontwikkelen prototype eerder aanleunt bij het domein van webmining, maar voor dit prototype zullen geen algoritmes voor webmining of tekstmining worden toegepast. Het prototype wil eerder meerwaarde creëren door relatief eenvoudig te achterhalen metagegevens zoals tijdgebonden informatie te visualiseren. In de literatuurstudie is bijgevolg meer aandacht besteed aan het onderdeel van de visuele analyse en voornamelijk aan het visualiseren van informatie uit tekst. Feature-based methodes maken gebruik van verschillende functies op het niveau van een woord of op het niveau van een document om tekst te visualiseren. Tijdlijnen en woordenwolken zijn maar enkele van de technieken die gebruikt kunnen worden. Ze worden ook toegepast in het prototype. Ze geven een intuïtieve visuele samenvatting van documenten door de sleutelwoorden te tonen in een compacte lay-out.

De meeste bestaande tools voor het opzoeken van informatie voor literatuuronderzoek maken geen gebruik van visualisatietechnieken die kunnen helpen bij het analyseren van de gevonden publicaties. ProQuest biedt wel al een grafiek aan die de evolutie in de tijd van de gevonden publicaties duidelijk maakt en waarop men de informatie verder kan filteren.

Het gerealiseerde prototype maakt achterliggend gebruik van de zoekmachine Google Scholar. De tool beschikt over een zoekbalk waarin men het onderwerp, de auteur en het begin- en eindjaartal van de zoekopdracht kan ingeven. De tool levert een alfabetisch geordende lijst van aanverwante kernwoorden, een lijst van journals gesorteerd naargelang het aantal voorkomens, een woordenwolk van aanverwante kernwoorden en een grafiek met het aantal artikels in de tijd om de onderzoeksevolutie van een onderwerp voor te stellen. De tool beschikt ook over de mogelijkheden om de onderzoeksresultaten te filteren op een kernwoord of een journal. Men kan ook filteren op een bepaald jaartal door in de tijdlijn te dubbelklikken. Een gedetailleerde beschrijving van het prototype en van het ontwikkelingsproces is opgenomen in dit werk.

Het prototype is door de voordien geïnterviewde onderzoekers geëvalueerd op criteria als bruikbaarheid en kwaliteit. De onderzoekers hebben geen eenduidig antwoord geformuleerd op de vraag of de opgeleverde tool hen kon helpen om relevante artikels terug te vinden tussen de aangeleverde zoekresultaten. Er zijn zowel aanwijzingen pro als contra.
Het ter beschikking stellen van de aanverwante onderwerpen, de lijst van journals en de evolutie in de tijd werden door de meeste onderzoekers als een meerwaarde ervaren. Het toepassen van visualisatietechnieken zoals een tijdlijn en een woordenwolk werd ook positief onthaald. Over het algemeen zagen de onderzoekers wel potentieel in het prototype om het uitvoeren van een literatuurstudie te ondersteunen mits de nodige uitbreidingen voorzien worden. De mogelijkheden van het prototype werden te beperkt bevonden om echt bruikbaar te zijn in de praktijk. Ontbrekende functionaliteiten zoals uitgebreide filter- en zoekmogelijkheden, die vaak wel aanwezig zijn in andere applicaties en zoekmachines, werden gemist. Het geringe aantal zoekresultaten die de tool terug gaf na een zoekopdracht werd als een beperking vermeld, evenals het feit dat er enkel gebruik gemaakt werd van Google Scholar als achterliggende zoekmachine.

Uit de evaluaties van het prototype kunnen enkele conclusies getrokken worden. Het gerealiseerde prototype kan geen duidelijkheid bieden over de meerwaarde van het gebruik van technieken uit het domein van de business intelligence voor academici bij het uitvoeren van een literatuurstudie. Een reden hiervoor was het te beperkt aantal gevonden zoekresultaten door de tool om een zinvol visuele analyse op toe te passen. Visuele analyse is vooral nuttig om relaties en structuren weer te geven in zeer grote volumes aan informatie.

De methodes en technologieën die binnen het domein van de business intelligence worden toegepast zijn complex en geavanceerd en kunnen moeilijk binnen het beperkte werkgebied van een prototype uitgewerkt worden. Het innoverende aspect van het prototype zit in het feit dat visualisatietechnieken uit de visuele analyse, die op zich niet vernieuwend zijn, worden toegepast voor het evalueren van de zoekresultaten in het kader van een zoekopdracht voor een literatuurstudie. Uit de positieve reacties van de onderzoekers ten opzichte van het conceptuele gebruik van deze visualisatietechnieken kan men afleiden dat het toepassen van een combinatie van automatische analyses met interactieve visualisaties waarschijnlijk kan bijdragen voor het analyseren van de zoekresultaten. Maar verder onderzoek op grotere schaal met deelname van een groter aantal van academici verdeeld over diverse onderzoeksgebieden en universiteiten is nodig om de precieze behoeften duidelijker in kaart te brengen.

Men kan zich de vraag stellen of een nieuwe BI-tool naast andere reeds bestaande zoekmachines en applicaties voor het opzoeken van wetenschappelijke literatuur een goede oplossing is. De nieuwe tool zal immers ook moeten beschikken over de brede waaier aan functionaliteiten waarover dergelijke applicaties beschikken. Andere oplossingsrichtingen zijn het uitbreiden van bestaande zoekmachines met visuele analyse technieken of een tool die zich enkel toelegt op het analyseren van zoekresultaten die verzameld werden via meerdere zoekopdrachten en via verschillende zoekmachines en databases. De opgehaalde zoekresultaten zou men misschien eerst naar een afzonderlijke datawarehouse kunnen exporteren.

Inhoudsopgave

1. Introductiefase

1.1. Inleiding

Literatuuronderzoek is een belangrijk, maar tegelijk ook een zeer tijdrovend onderdeel van wetenschappelijk onderzoek. Academici hebben er dus baat bij om dit literatuuronderzoek zo efficiënt mogelijk uit te voeren. Steeds vaker maken ze hierbij gebruik van het web voor het opzoeken van relevante wetenschappelijke artikels. Zoekmachines zoals Google Scholar zijn daarbij een goed hulpmiddel. Het aantal resultaten dat men terugkrijgt van een zoekopdracht is echter vaak nog steeds zeer hoog en niet alle gevonden artikels zijn even relevant. Volgens Randolph (2009) worden bij het gebruik van elektronische bronnen slechts tien procent van de relevante artikels gevonden.

Het idee van deze thesis is om academici te ondersteunen bij hun literatuuronderzoek en dit door middel van een BI-tool. Door gebruik te maken van talrijke bronnen van informatie op het Internet kan een BI-tool zinvolle informatie aanleveren over een welbepaald onderwerp en de selectie van relevante artikels vereenvoudigen. De BI-tool is waardevol indien deze kan tegemoetkomen aan tekortkomingen die onderzoekers ervaren bij het huidige proces van literatuuronderzoek en indien hiermee extra functionaliteiten worden aangeboden die onderzoekers helpen om een literatuuronderzoek efficiënter en effectiever te laten verlopen. De BI-tool is innovatief indien deze binnen het domein van Business Intelligence een leemte opvult, die blijkt uit de bijhorende literatuurstudie.

Dit onderzoek en de ontwikkelde tool zou de kwaliteit van academisch onderzoek kunnen verhogen aangezien ze de onderzoekers helpen om tot een systematische aanpak van de literatuurstudie te komen. Daardoor verhoogt de kans om alle relevante artikels te vinden en zal men deze ook sneller vinden, wat leidt tot een tijdsbesparing.

1.2. Onderzoeksvraag

Op welke manier kan een business intelligence tool academische onderzoekers ondersteunen bij het voeren van een literatuurstudie?

Dat is de onderzoeksvraag die in deze thesis verder onderzocht wordt.

Een deel van het onderzoek bestaat er uit om te achterhalen hoe onderzoekers een literatuurstudie uitvoeren om zo te kunnen bepalen welke functionaliteit voor hen een gewenst hulpmiddel is.

Door gebruik te maken van de BI-tool kunnen nieuwe patronen en nieuwe inzichten zichtbaar gemaakt worden. Er zijn reeds verschillende tools op de markt die kennis extraheren uit de grote hoeveelheid aan informatie die beschikbaar is op het Internet. De meeste tekstmining tools maken gebruik van complexe algoritmes. Onderzoek om de kwaliteit van tekstmining tool te verbeteren, bijvoorbeeld door de algoritmes te optimaliseren, valt buiten het doelgebied van deze thesis. Domeinkennis inbrengen in het proces van tekstmining is ook een mogelijkheid om de kwaliteit te verhogen, maar deze methode is hier niet van toepassing omdat deze thesis focust op literatuurstudies in het algemeen, ongeacht het onderzoeksdomein.
Wel is het zo dat tools als Hub Med (Von Isenburg, 2007) en Agilent Literature Search ("Agilent Literature Search") een eigen interface aanbieden om gegevens te zoeken en te visualiseren, terwijl ze achterliggend gebruik maken van bestaande zoekmachines. Deze applicaties zijn echter enkel bruikbaar in hun domeingebied van de biologie. De voor deze thesis te ontwikkelen BI-tool, zal op een gelijkaardige manier gebruik maken van bestaande zoekmachines, maar zal specifiek die gegevens ontsluiten die over alle domeinen van de wetenschap bruikbaar zijn in het kader van een literatuurstudie.

Vanuit het domein van de visuele analyse zijn er bestaande technieken die gebruikt kunnen worden om de gevonden gegevens te visualiseren. Hierbij denken we bijvoorbeeld aan interactieve zooming bij het voorstellen van de populariteit van een topic, waarbij men kan inzoomen op een bepaalde periode. De meeste tools voor het visualiseren van informatie zijn echter enkel bruikbaar binnen hun domein.

Deze thesis wil minder de nadruk leggen op het zoeken van de juiste informatie aan de hand van complexe algoritmes, maar meer op hoe je door middel van visualisatie van relatief eenvoudig te achterhalen gegevens, toch een meerwaarde kan bieden aan onderzoekers over verschillende wetenschappelijke onderzoeksgebieden. Veelal gaat het immers om metadata die aangeleverd wordt bij een gevonden document. Maar het is net het met elkaar in verband brengen van deze meta-gegevens, zoals bijvoorbeeld het vaktijdschrift waarin het document werd gepubliceerd, dat een meerwaarde kan bieden.

1.3. Methodologie

1.3.1. Design science

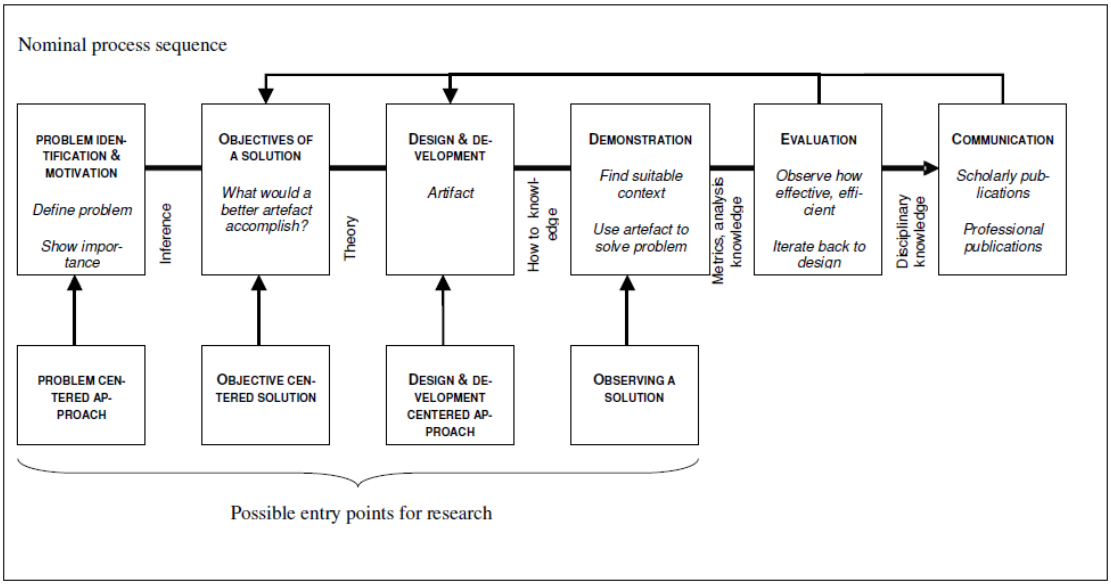
Dit onderzoek situeert zich in het domein van design science theorie. Design science onderzoek is erop gericht om via de ontwikkeling van een nieuw en nuttig artefact tot een algemene kennisbijdrage te komen. Design science in informatiesystemen zorgt dat de grenzen van de capabiliteit van mensen en organisaties verlegd worden door het creëren van innovatieve artefacten. Louter een beschrijving van de artefact, de toegepaste methode en technische regels volstaat niet om te kunnen spreken van een design theorie. Hiervoor is een prescriptieve beschrijving vereist die de design principes toelicht en uitlegt waarom de methode werkt en onder welke condities. De twee voornaamste activiteiten van design science zijn dan ook: bouwen en evalueren (March & Smith, 1995).

Onder IT artefacten verstaat men concepten, modellen om het probleem te representeren, methodes en ten slotte instanties van IT toepassingen. Ze hebben als doel om succesvolle informatiesystemen te analyseren, begrijpen en ontwikkelen. De gerealiseerde instanties moeten de haalbaarheid van het gebruik van de artefacten aantonen (March & Smit, 1995).

Volgens Peffers et al. (2007) vereist een design science research bijdrage dat men:

- een relevant IT probleem identificeert, beschrijft en motiveert;
- de doelstelling van de oplossing weergeeft en aantoont dat er voor het probleem nog geen adequate oplossingen bestaan;
- een innovatief artefact ontwikkelt en presenteert;
- de effectiviteit en de bruikbaarheid van het IT artefact aantoont en quoteert;
- de bijdrage van het artefact aan de kennis en de praktijk binnen IT aantoont en evalueert;
- communiceert over de gevolgen en het belang van de oplossing en het artefact.

De verschillende stappen in het procesmodel volgens Peffers et al. (2007) worden weergegeven in figuur één.



Figuur 1 Design science onderzoeksmethodologie procesmodel (Peffers, et al., 2007)

1.3.2. Concrete aanpak

Concreet betekent dit dat het onderzoek voor deze thesis verloopt in een aantal fases.

Introductiefase

In deze fase komt men tot een probleemdefinitie en een omschrijving van de onderzoeksvraag en eventuele deelvragen. Om tot deze onderzoeksvraag te komen wordt eerst de relevantie van literatuuronderzoek aangehaald. Hier komt het domein waarin het probleem zich situeert naar voor en wordt de bedrijfskundige relevantie verduidelijkt. De centrale onderzoeksvraag wordt gekaderd in zijn onderzoeksdomein. Ten slotte geven de deelvragen aan welke designtheorie verder bestudeerd dient te worden, in dit geval business intelligence. Vervolgens worden ook deelvragen opgesteld om na te gaan wat de doelstellingen en de scope van de te ontwikkelen BI-tool zijn.

Studie van literatuuronderzoek

Hoe verloopt het literatuuronderzoek bij onderzoekers?

Om te onderzoeken welke functionaliteiten door onderzoekers gewenst zijn voor het vergemakkelijken van literatuuronderzoek dient er eerst onderzocht te worden hoe onderzoekers momenteel hun literatuuronderzoek aanpakken. Het is belangrijk om inzicht te verwerven in het verloop van dit proces en om de verschillende stappen die de onderzoekers nemen tijdens een literatuuronderzoek te onderscheiden. Het is de bedoeling om te achterhalen waar er zich problemen

voor doen, welke problemen dit zijn en waar er mogelijke verbeteringen aangebracht kunnen worden.

Welke functionaliteiten zouden onderzoekers nodig hebben in een BI-tool?
Vooraleer er aan een prototype van een tool begonnen kan worden, moet er eerst onderzocht worden aan welke functionaliteiten onderzoekers behoefte hebben. De voornaamste functionele vereisten zullen in kaart gebracht worden.
Om inzicht te verwerven in literatuuronderzoek wordt relevante beschrijvende literatuur onderzocht. Daarnaast wordt aan de hand van interviews informatie verzameld over het uitvoeren van literatuurstudies en over de vereisten van onderzoekers in het kader van literatuuronderzoek.

Literatuuronderzoek Business Intelligence

Een studie van de literatuur omtrent Business Intelligence zorgt voor de kadering van de te ontwikkelen BI-tool binnen de theorie. Onderzoek naar reeds bestaande toepassingen van business intelligence voor literatuurstudies is nodig om te vermijden dat de tool onvoldoende innoverend is.

In welke mate ondersteunen bestaande BI tools onderzoekers reeds en bieden ze de nodige functionaliteiten aan?
Het ontwikkelde prototype dient vernieuwend te zijn. Om dit te verzekeren wordt eerst onderzocht welke functionaliteiten reeds aangeboden worden door bestaande BI-tools. Daartoe kan er een overzicht opgesteld worden van de voornaamste beschikbare BI-tools. Hierbij wordt voor iedere tool weergegeven welke functionaliteiten deze biedt, welke tekortkomingen er nog zijn en welke de voor- en nadelen zijn bij het gebruik van de tool, bijvoorbeeld beperkingen omtrent de omgeving waarin deze gebruikt kan worden.

Welke visualisatietechnieken zouden nuttig kunnen zijn voor onderzoekers in hun zoektocht naar wetenschappelijke relevante literatuur?
Binnen het domein van visuele analyse kunnen er verschillende visualisatie-technieken teruggevonden worden. De bruikbaarheid van de meeste tools voor het visualiseren van informatie is vaak afhankelijk van het domein waarbinnen ze gebruikt worden. In deze thesis wordt dus nagegaan welke visualisatietechnieken bruikbaar zijn voor het prototype van de BI-tool. Het is belangrijk dat een geschikte visualisatietechniek wordt toegepast. Indien de informatie op de juiste manier wordt gevisualiseerd, kan er hierdoor een meerwaarde gecreëerd worden voor de onderzoeker en zal deze sneller besluiten kunnen trekken. Omgekeerd is

het zo dat er nuttige informatie verloren kan gaan indien de gegevens niet aanschouwelijk worden voorgesteld. Het gevolg hiervan kan zijn dat de BI-tool niet meer als waardevol wordt aanzien en dat deze na verloop van tijd niet meer gebruikt wordt.

Ontwikkeling en beschrijving van de BI tool

Hoe kan men de verworven inzichten omvormen tot een prototype?

Ten slotte worden de gevonden inzichten omgevormd tot een prototype. Een grondige analyse is nodig om de functies die de BI-tool gaat aanleveren in detail te specifiëren en om na te gaan hoe deze aangereikt kunnen worden met behulp van het prototype. Het gerealiseerde prototype zal voldoende functionaliteit moeten bevatten. Het moet de academici een duidelijk beeld verschaffen van hoe de beoogde objectieven bereikt kunnen worden. De onderzoekers moeten aan de hand van het prototype de BI-tool kunnen beoordelen. Ze moeten kunnen nagaan in welke mate de tool hun vereisten kan invullen.

Dit is het omvangrijkste deel van de thesis. Het bevat niet enkel een omschrijving van de ontwikkelde BI-tool, maar het geeft ook inzicht in het verloop van het ontwikkelingsproces om tot de tool te komen.

Een grondige analyse is nodig om de functies die de Bi-tool gaat aanleveren in detail te specifiëren. Dit omvat een gedetailleerd onderzoek naar de gegevens die ontsloten dienen te worden, het ontwerp van de gebruikersinterface, de keuze van ontwikkelingstools, onderzoek naar haalbaarheid en performantie, de mogelijkheid om gebruik te maken van bestaande technieken inzake visuele analyse om de opgehaalde informatie te presenteren en inzake sociale netwerkanalyse om relevante linken te leggen tussen gegevens. Deze analyse begint vanuit de resultaten die reeds gevonden werden in de literatuurstudie en de afgenomen interviews.

Daarna dient een prototype uitgewerkt te worden. De volgende stap is het effectief ontwikkelen van een werkende BI-tool. Dit onderzoek beperkt zich tot het ontwikkelen van een prototype van de BI-tool. Tijdens deze fase wordt er een beschrijving gemaakt van het iteratief ontwikkelingsproces en worden tussentijdse testresultaten en bevindingen weergegeven.

Evaluatiefase

Tijdens deze fase wordt het artefact geëvalueerd op criteria als bruikbaarheid en kwaliteit. Dit kan nagegaan worden door onderzoekers met de tool te laten

experimenteren en hun vervolgens te bevragen naar de behulpzaamheid van de tool en hun beoordeling op basis van de reeds genoemde criteria.

Bespreking

De voornaamste designprincipes die ontdekt werden tijdens het project worden geïnventariseerd. Er wordt nagegaan wat geleerd werd uit het onderzoek. Er wordt getoetst of de bevindingen het doel van het onderzoek bevestigen of verwerpen.

Conclusie

De belangrijkste bevindingen worden in een conclusie gegoten. De centrale onderzoeksvraag die opgesteld werd in de introductiefase wordt beantwoord.

2. Studie van literatuuronderzoek

2.1. Methode

In eerste instantie moet er inzicht verkregen worden in hoe onderzoekers een literatuurstudie uitvoeren en welke problemen ze daarbij ervaren. Het verwerven van informatie over het uitvoeren van literatuurstudies door middel van onderzoek van bestaande literatuur gaat niet zonder moeilijkheden. Een algemene zoekterm als 'literature review' resulteert namelijk in een lijst van literatuurstudies en niet in een overzicht van wetenschappelijke artikels met betrekking tot literatuurstudies. Bovendien is het aantal papers met betrekking tot dit onderwerp vrij beperkt. Vandaar dat er voor de studie van het literatuuronderzoek gekozen is om 'phenomenological' research toe te passen. Het doel van deze methode is tot de essentie te komen van een bepaald fenomeen dat werd vastgesteld (Moustakas, 1994).

De eerste stap van 'phenomenological' research is 'bracketing' oftewel het identificeren van het probleem dat nader onderzocht wordt. Het fenomeen dat hier onderzocht wordt is de vaststelling dat onderzoekers moeilijkheden ondervinden bij het uitvoeren van een literatuurstudie. De volgende stap is het verzamelen van gegevens in verband met het fenomeen. De onderzoeker kan hiervoor personen interviewen die het fenomeen ervaren. Wanneer deze methode wordt toegepast als een review worden papers van wetenschappers doorgenomen in plaats van interviews af te nemen. Voor de studie van literatuuronderzoek in deze thesis worden zowel interviews afgenomen als papers doorgenomen. De volgende stappen zijn het identificeren van zinvolle verklaringen en betekenis geven aan deze verklaringen. De laatste stap ten slotte is om tot een beschrijving van de essentie van het fenomeen te komen (Moustakas, 1994).

2.2. Interviews

Om informatie te verzamelen omtrent de problemen die zich voordoen bij het uitvoeren van een literatuurstudie is besloten om drie doctoraatstudenten en twee professoren binnen de onderzoeksgroep Economie van de UHasselt te interviewen.

2.2.1. Interviewmethode

Er werd gebruik gemaakt van semi-gestructureerde interviews. Elke geïnterviewde ontving grotendeels dezelfde vragen. Bij het opstellen van de vragen werd een onderscheid gemaakt tussen het interviewen van

doctoraatstudenten en van professoren. Iedere geïnterviewde was actief binnen een ander Bedrijfseconomisch onderzoeksveld. De ondervraagden werden gecontacteerd via mail. Deze mail is terug te vinden in bijlage één. De onderstaande vragen werden gesteld aan de onderzoekers:

Algemeen

- Kan u me uitleggen hoe u juist aan een literatuuronderzoek begint?
- Welke informatie zou u nog kunnen gebruiken tijdens uw literatuuronderzoek?
    - de belangrijkste papers over onderwerp
    - de belangrijkste auteurs over een onderwerp
    - aanverwante onderwerpen
    - evolutie van de onderzoeksinteresse in een onderwerp
    - link tussen onderzoekers
    - de belangrijkste journals waarin een topic verschenen is
- Is er nog andere informatie die u handig zou vinden?

Doctoraatsstudent

- Sinds wanneer bent u al aan het doctoreren?
- Is er een verschil tussen hoe u nu aan uw literatuuronderzoek begint ten opzichte van het begin van uw doctoraat?
- Zijn er tips die u vroeger had willen krijgen?

Professoren

- Op welke manier bereidt u uw doctoraatsstudenten voor om aan een literatuuronderzoek te beginnen?
- Wat zijn de grootste lacunes bij masterstudenten op het vlak van literatuuronderzoek?

2.2.2. Interviewresultaten

"Kan u me uitleggen hoe u juist aan een literatuuronderzoek begint?"

Op de eerste vraag werden heel gevarieerde antwoorden geformuleerd. Al de onderzoekers beginnen vaak rechtstreeks op zoektermen te zoeken met behulp van Google Scholar, Web of Science of Science Direct. Maar andere manieren om aan een literatuurstudie te beginnen verschillen van persoon tot persoon. Zo kan onder andere het lezen van de inhoudsopgaven van specifieke tijdschriften, van

review artikels, van publicaties van specifieke onderzoekers en van congrespapers een eerste stap zijn om aan een literatuurstudie te beginnen.
Eén van de doctoraatstudenten begint via Google Scholar te zoeken naar een artikel en zoekt vervolgens met behulp van 'geciteerd door' naar andere artikels met betrekking tot het onderwerp. Bij een nieuw topic maakt deze doctoraatstudent eerst gebruik van Google voor hij naar wetenschappelijke literatuur zoekt.
Eén van de professoren maakt na het verzamelen en lezen van literatuur een overzicht met behulp van een tabel of een schema. Deze tabel wordt vervolgens gebruikt als houvast om de literatuurstudie te schrijven.

"Welke informatie zou u nog kunnen gebruiken tijdens uw literatuuronderzoek?"
Alle onderzoekers zijn het er over eens dat het niet eenvoudig is om te bepalen welke papers over een bepaald onderwerp belangrijk zijn. Ze vertrouwen hierbij voornamelijk op de resultaten van zoekmachines. Dit geldt ook voor de belangrijkste auteurs over een onderwerp. Alle onderzoekers zouden het handig vinden als ze aanverwante onderwerpen konden krijgen. Het is wel zo dat één van de onderzoekers deze informatie al ter beschikking had in zijn domein. Het gaat hierbij namelijk over PubMed. Verder werd ook de evolutie van de onderzoeksinteresse in een onderwerp als nuttig beschouwd. De link tussen onderzoekers is voor toegepaste onderzoekers vaak niet zo relevant. Vandaar dat één van de proffen geen toegevoegde waarde ziet in het aanbieden van een link. Voor de link tussen onderzoekers bestaat er ook reeds het Fris Onderzoeksportaal ("FRIS Onderzoeksportaal", 2008). De belangrijkste journals daarentegen zouden wel een toegevoegde waarde kunnen bieden.

"Is er nog andere informatie die u handig zou vinden?"
Een suggestie die gegeven werd door één van de doctoraatsstudenten is de beschikbaarheid van het volledige artikel en informatie over de prijs om toegang te krijgen tot dit artikel. Een andere doctoraatsstudent denkt hierbij eerder aan de onderverdeling van de hoofdstukken van een artikel en de belangrijkste universiteiten bij een topic. Eén van de professoren haalde aan dat het handig zou zijn als er specifiek naar reviewartikels gezocht kon worden.

"Sinds wanneer bent u al aan het doctoreren?"
De verschillende doctoraatstudenten waren anderhalf jaar, twee en een jaar en drie maanden en twee en een half jaar aan het doctoreren.

"Is er een verschil tussen hoe u nu aan uw literatuuronderzoek begint ten opzichte van het begin van uw doctoraat?"
De verschillende doctoraatstudenten merken geen groot verschil op tussen hoe ze vroeger en nu hun literatuuronderzoek aanvatten. Eén van de drie was voordien docent evidence-based practice en doceerde daarbij onder andere over literatuuronderzoek.

"Zijn er tips die u vroeger had willen krijgen?"
Een tip die één van de doctoraatstudenten vroeger had willen krijgen, is het gebruiken van Google Scholar.

"Op welke manier bereidt u uw doctoraatsstudenten voor om aan een literatuuronderzoek te beginnen?"
Eén van de twee professoren bereidt zijn doctoraatsstudenten voor door toe te lichten hoe hij zelf aan een literatuurstudie begint. De andere professor gaat ervan uit dat doctoraatsstudenten gedurende hun opleiding reeds voldoende vaardigheden verworven hebben.

"Wat zijn de grootste lacunes bij masterstudenten op het vlak van literatuuronderzoek?"
De literatuurstudie van masterstudenten mist volgens één van de professoren vaak structuur, ze blijven veel te breed. Ze verliezen veel tijd aan het volledig lezen van een artikel terwijl dit weinig oplevert. Volgens de tweede professor weten masterstudenten vaak niet waar ze het volledige artikel kunnen terugvinden.

2.3. Onderzoek bestaande literatuur

2.3.1. Definitie literatuuronderzoek

Een literatuuronderzoek dient de inhoud van bestaande wetenschappelijk studies omtrent een onderwerp te beschrijven, samen te vatten, te evalueren, te verduidelijken en te integreren (Cooper, 1988, in Khoo, Na en Jaidka, 2011). Volgens Sekaran en Bougie (2009) is een literatuurstudie een stapsgewijs proces waarbij artikels in verband met een onderwerp worden geïdentificeerd, geëvolueerd met betrekking tot het probleem en gedocumenteerd.

2.3.2. Doel literatuuronderzoek

Relevante literatuur is van essentieel belang voor ieder academisch project. Een goede literatuurstudie zorgt voor een goede basis voor het verkrijgen van kennis (Webster en Watson, 2002). Wanneer de literatuurstudie gebreken vertoont, kan de volledige studie als gebrekkig beschouwd worden. Een onderzoeker kan geen significant onderzoek uitvoeren zonder eerst de literatuur in het domein te begrijpen (Boote en Beile, 2005). Literatuuronderzoek zorgt ervoor dat theorieën makkelijker ontwikkeld kunnen worden en identificeert de gebieden waar onderzoek nodig is. Een goede literatuurstudie is volledig en focust op de verschillende concepten binnen een bepaald onderzoekdomein. Men beperkt zich hierbij niet tot één enkele onderzoekmethodologie, één bepaalde groep journals of één bepaalde geografische regio (Webster en Watson, 2002).

Volgens Khoo, Na en Jaidka (2011) is een literatuurstudie meer dan zomaar een overzicht van bestaande literatuur over een onderwerp. Een literatuurstudie situeert het onderzoek binnen een bepaald onderzoeksdomein, levert een verantwoording voor het uitvoeren van het onderzoek, geeft een aflijning van de hiaten in de bestaande literatuur die door het onderzoek worden ingevuld en helpt bij verschillende aspecten van het onderzoek zoals bijvoorbeeld de te hanteren methodologie. Deze doelstellingen vindt men ook terug in de zes functies die Hart (1998) identificeert. De zes functies van Hart zijn de volgende: omschrijven wat reeds gedaan is en wat nog gedaan moet worden, identificeren van belangrijke variabelen relevant aan het onderwerp, synthetiseren van eerdere resultaten en ideeën om een nieuw perspectief te vinden, rationaliseren van significante problemen, identificeren van de belangrijkste methodologieën en onderzoekstechnieken die reeds gebruikt werden en ten slotte plaatsen van het onderzoek in de context van state-of-the-art-ontwikkelingen.

2.3.3. Taxonomie van literatuuronderzoek

Een doeltreffende manier om een literatuurstudie aan te vatten is te beschouwen waar de studie zich situeert in de taxonomie van literatuurstudies van Cooper (1988, in Randolph, 2009). Cooper classificeert literatuurstudies aan de hand van vijf eigenschappen: focus, doel, perspectief, dekking, organisatie en doelpubliek. Deze eigenschappen worden nog verder onderverdeeld.

Zo bespreekt Cooper vier mogelijke focussen namelijk: onderzoeksuitkomst, onderzoeksmethode, theorieën en ten slotte toepassingen of practica.

- Focussen op de onderzoeksuitkomst is voor de hand liggend, maar wel heel belangrijk om ervoor te zorgen dat men geen informatie mist over

een bepaald onderdeel. Een loutere opsomming van citaten en de inhoud van bestaande literatuur en daaruit een conclusie trekken volstaat niet.

- Focussen op de onderzoeksmethode helpt om de sterktes en de zwaktes van een methode te identificeren.
- Focussen op theorieën is van belang om na te gaan welke theorieën reeds bestaan, om de relaties tussen verschillende theorieën duidelijk te maken en om te bepalen in welke mate bestaande theorieën werden onderzocht. Deze benadering is van belang voor onderzoeken die zich tot doel stellen om voortgang te realiseren in een nieuwe theorie.
- Focussen op toepassingen kan helpen om een bepaalde praktische behoefte in te vullen die tot nu toe nog niet werd verwezenlijkt.

Het doel van een literatuurstudie is meestal om bevindingen te integreren, te veralgemenen of om duidelijkheid te verschaffen in geval van een discussie in een bepaald domein. Een literatuurstudie kan ook als doel hebben om voorgaande studies kritisch te analyseren en problemen te identificeren.
Het perspectief van een literatuurstudie hangt vaak af van het feit of het gaat om een kwalitatief of een kwantitatief onderzoek. In een kwalitatief onderzoek hanteert de auteur vaak zijn eigen vooringenomen standpunt als basis om de discussie aan te gaan. In een kwantitatief onderzoek daarentegen neemt de auteur een neutraal positie in en geeft de bevindingen weer als feiten.
Het bepalen van de dekking van een literatuurstudie is een belangrijke stap. Cooper onderscheidt dan ook vier scenario's: een uitgebreide review, een uitgebreide review met selectieve citaten, een representatieve sample en een doelgerichte sample.

- In een uitgebreide review belooft de onderzoeker om iedere beschikbare informatie, gepubliceerd of niet, omtrent een onderwerp te plaatsen.
- Omdat de voorgaande manier van werken meer tijd in beslag kan nemen dan beschikbaar is, kan men het onderzoeksgebied afbakenen zodanig dat het aantal artikels te overzien blijft. Cooper spreekt in dit geval van een uitgebreide review met selectieve citaten.
- Een derde manier van aanpak is om representatieve selectie van artikels in beschouwing te nemen en deze bevindingen te extrapoleren op het volledige onderzoeksgebied. Maar het willekeurig selecteren van artikels kan fouten met zich meebrengen. Daarom is het bij deze methode zinvol op te bewijzen dat de geselecteerde artikels representatief zijn.

De laatste methode die Cooper aanhaalt is een doelgerichte selectie van artikels te maken. Het komt er in dit geval op neer om de lezer ervan te overtuigen dat de geselecteerde artikels kernartikels zijn en dat de artikels, die niet weerhouden werden, minder belangrijk zijn.
Men kan een literatuurstudie op verschillende manieren organiseren. Hierbij onderscheidt Cooper een conceptuele, een methodologische en een historische opmaak. Deze laatste is chronologisch georganiseerd.
De laatste eigenschap in de taxonomie van Cooper is het publiek. Het doelpubliek van een literatuurstudie kan bestaan uit gespecialiseerde of algemene academici, beleidsmakers of personen die de literatuurstudie in de praktijk toepassen. Men vermijdt best dat een literatuurstudie enkel voor een algemeen, niet academisch publiek geschreven wordt.

2.3.4. Uitvoering van literatuuronderzoek

Bij het schrijven van een literatuuronderzoek worden de volgende fases doorlopen: probleemstelling, het verzamelen van informatie, evaluatie van de data, analyse en interpretatie van de data en ten slotte de presentatie (Cooper, 1984, in Randolph, 2009).

Probleemstelling

De probleemstelling van een literatuurstudie begint bij het bepalen van de vragen die de literatuurstudie dient te beantwoorden. Bij het opstellen van deze vragen moet men rekening houden met het doel en de focus van de studie. Voor een studie die als doel heeft om problemen te identificeren en waarbij de focus ligt op theorieën, zal men een vraag kunnen stellen als: Welke theorieën werden gebruikt om een bepaald fenomeen te verklaren?
Vervolgens dient men de criteria vast te leggen die bepalen of een artikel in beschouwing wordt genomen of niet. Op die manier kan men relevante en irrelevante studies onderscheiden. Ook deze criteria zijn afhankelijk van de focus, de doelen en de dekking van de literatuurstudie (Randolph, 2009).

Verzamelen van informatie

Volgens Randolph (2009) is de volgende stap na het opstellen van een probleemstelling van de literatuurstudie, het verzamelen van literatuur.
Webster en Watson (2002) raden een gestructureerde benadering aan bij het zoeken naar bronnen voor de literatuurstudie. Volgens hun worden de belangrijkste bijdrages vaak gevonden in de voornaamste journals. Het is dus logisch om eerst in deze journals op zoek te gaan naar relevante artikels. Journal

databases zoals Proquest vergemakkelijken de identificatie van relevante artikels, maar het scannen van de inhoudspagina van een journal kan een handige manier zijn om artikels terug te vinden die niet gevonden zouden worden met een zoekopdracht aan de hand van een sleutelwoord. Een volgende stap kan zijn om verder te gaan met de referenties die men terugvindt in de gevonden artikels en de artikels op te zoeken waarnaar gerefereerd werd. Ten slotte kan men met behulp van Web of Science zoeken naar de artikels die de eerder gevonden papers citeren.

Uit het onderzoek van Randolph (2009) blijkt dan weer dat de verzameling van literatuur vaak begint met een elektronische zoektocht binnen academische databases en het Internet. Maar hij merkt op dat elektronische zoektochten vaak slechts tien procent van de artikels leveren. Er zijn verscheidene methodes om de overige negentig procent te lokaliseren. Randolph haalt hierbij aan dat de belangrijkste methode hiervoor is om te zoeken naar de artikels die als bron van de gevonden artikels dienden. Deze methode werd eerder al aangehaald door Webster en Watson (2002). Randolph (2009) haalt ook nog aan dat het nuttig kan zijn om referentielijsten te delen met collega's en experts in het veld om ontbrekende artikels terug te vinden.

Data-evaluatie

Nadat men de literatuur verzameld heeft, moet men deze evalueren. Men dient na te gaan of ze voldoet aan de eerder opgestelde criteria. Men hoort na te gaan of de kwaliteit van het onderzoek, dat besproken wordt in de gevonden literatuur, voldoende is (Randolph, 2009).

Data-analyse en interpretatie

Tijdens de data analyse en interpretatie fase dient de onderzoeker de verschillende data te integreren en te interpreteren (Randolph, 2009).

Presentatie

Ten slotte dient de auteur van de literatuurstudie te bepalen welke informatie belangrijk is en voorgesteld zal worden en welke informatie men kan weglaten (Randolph, 2009).

2.3.5. Vaak voorkomende fouten

Gall, Borg en Gall (1996) onderzochten vaak voorkomende fouten tijdens het uitvoeren van literatuurstudies. De zeven meest voorkomende fouten zijn de volgende:

1. Er is geen duidelijk verband tussen de bevindingen van de literatuurstudie en het eigen onderzoek van de auteur.
2. De auteur had niet voldoende tijd besteed om de beste auteurs en bronnen te identificeren in verband met zijn onderwerp.
3. Er wordt tijdens de literatuurstudie voornamelijk gebruik gemaakt van secundaire bronnen in plaats van primaire bronnen.
4. De onderzoeker neemt de bevindingen van een andere onderzoeker over zonder enige kritische reflectie.
5. De zoekprocedures die gebruikt werden in de literatuurstudie werden niet gerapporteerd.
6. Enkel geïsoleerde statistische resultaten worden gerapporteerd.
7. Tegenstrijdige bevindingen en alternatieve interpretaties in het synthetiseren van kwantitatieve literatuur worden niet beschouwd.

2.4. Evaluatie en interpretatie

Uit de afgenomen interviews blijkt dat alle ondervraagden een literatuurstudie op een andere manier aanvatten. De doctoraatsstudenten beginnen meestal onmiddellijk met het verzamelen van de gegevens. Aan de probleemstelling en het vastleggen van criteria om te bepalen welke artikels relevant zijn wordt door studenten weinig aandacht besteed. Dat blijkt ook uit de opmerking van één van de professoren die stelt dat de literatuurstudie van masterstudenten vaak structuur mist en veel te breed blijft. Studenten verliezen volgens hem veel tijd aan het volledig lezen van een artikel terwijl dit weinig oplevert. Volgens de tweede professor weten masterstudenten vaak niet waar ze het volledige artikel kunnen terugvinden.

Dit alles wijst erop dat men vooral veel energie steekt in het opzoeken van relevante data. De geïnterviewde onderzoekers geven allemaal aan dat het niet eenvoudig is om te bepalen welke papers over een bepaald onderwerp belangrijk zijn en welke auteurs van betekenis zijn. In het bijzonder in deze stap van verzamelen van informatie kan een BI-tool dus de onderzoekers mogelijk ondersteunen. Voor het uitvoeren van de overige stappen zoals de probleemstelling, de data-evaluatie, de data-analyse en interpretatie is er minder noodzaak aan een tool. Eén van de professoren maakt gebruik van een tabel, die als leidraad dient bij de uitvoering van de literatuurstudie. Dergelijke tabel kan een hulpmiddel zijn bij het evalueren en analyseren van de data en het aftoetsen van de gevonden artikels aan de vooropgestelde criteria. Daarnaast kan men in een tabel ook de links bijhouden tussen de gevonden artikels. Mogelijk zou een tool ook tijdens deze fase een hulpmiddel kunnen zijn, maar omdat het

voornaamste probleem zich situeert bij het opzoeken zelf, is er voor geopteerd om de scope van de te ontwikkelen tool te beperken tot de fase van het verzamelen van de informatie literatuuronderzoek. Uit de studie van de literatuur blijkt ook dat de auteur vaak niet voldoende tijd besteedt om de beste auteurs en bronnen te identificeren in verband met zijn onderwerp (Gall, Borg en Gall, 1996).

De volgende zaken werden door al de geïnterviewde personen vermeld als belangrijke hulpmiddelen om het vinden van relevante artikels te vereenvoudigen bij het zoeken van informatie via een zoekmachine:

- een overzicht van aanverwante onderwerpen;
- de evolutie van de onderzoeksinteresse in een onderwerp;
- een lijst van de belangrijkste journals.

Deze items kunnen onderzoekers helpen om sneller relevante onderwerpen te vinden. Uit de literatuur omtrent het verzamelen van informatie komt ook naar voren dat de belangrijkste bijdrages vaak gevonden worden in de voornaamste journals (Webster en Watson, 2002).

Minder belangrijk maar wel interessant vond men:

- de beschikbaarheid van het volledige artikel;
- informatie over de prijs om toegang te krijgen tot een artikel;
- de onderverdeling van de hoofdstukken van een artikel;
- de belangrijkste universiteiten bij een topic;
- de mogelijkheid om specifiek naar reviewartikels te zoeken.

Uit de interviews blijkt tevens dat Google Scholar een zoekmachine is die regelmatig gebruikt wordt door academici.

2.5. Essentie van het probleem

Onderzoekers ervaren veel problemen bij het zoeken naar relevante artikels. Daarom is het zinvol om de scope van een te ontwikkelen BI-tool te beperken tot de fase in de literatuurstudie van het verzamelen van de informatie. De items die in de voorgaande paragraaf opgesomd werden kunnen de onderzoekers helpen bij hun zoektocht naar relevante artikels. In een volgende fase zal er binnen het BI-domein bekeken worden hoe deze informatie van het Internet geëxtraheerd kan worden.

3. Literatuuronderzoek business intelligence

Business intelligence is een ruim begrip dat vaak gehanteerd wordt voor allerlei processen die te maken hebben met het verzamelen en analyseren van informatie. Daarom wordt er in dit hoofdstuk onderzocht wat verstaan wordt onder business intelligence om vervolgens te kunnen nagaan welke deelgebieden en technieken binnen het domein van business intelligence relevant zijn voor het ophalen en visualiseren van informatie omtrent literatuuronderzoek.

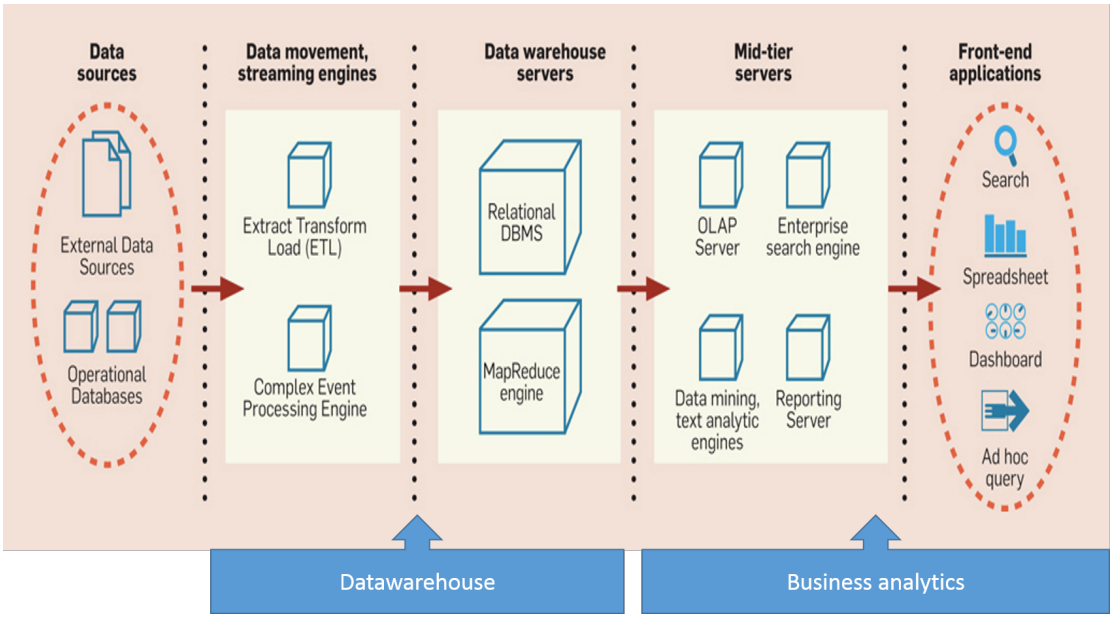
3.1. Definitie business intelligence

Business intelligence wordt gedefinieerd als het verzamelen en analyseren van gegevens om tot kennis te komen die helpt om betere managementbeslissingen te maken. Deze definitie werd eerder al gebruikt door Claus (2001) en sluit aan bij de definitie van Negash (2004). Deze stelt immers dat business intelligence systemen dataverzameling, data-opslag en kennismanagement combineren met analytische tools om zo complexe interne en competitieve informatie voor beslissingnemers te bekomen. Lönnqvist & Pirttimäki (2006) benadrukken het managementdoel van business intelligence. Volgens hen refereert business intelligence naar een managementfilosofie en een tool die gebruikt wordt om een organisatie te beheren en bedrijfsinformatie te verfijnen om doeltreffendere bedrijfsbeslissingen te kunnen nemen.

3.2. Onderverdeling business intelligence

Business intelligence is een zeer breed domein. Meestal vertrekt men vanuit een gecentraliseerde opslagplaats voor data. Deze opslagplaats is gewoonlijk een datawarehouse. Om op basis van deze data rapporten en query's te creëren en om deze data en rapporten te analyseren, worden eindgebruikertools gebruikt. Men verwijst ook wel naar deze tools met behulp van de term business analytics. Datavisualisatie behoort hier ook toe. Om niet voor de hand liggende relaties te vinden tussen grote hoeveelheden data maakt men gebruik van datamining. Indien het gaat over relaties tussen teksten spreekt men over tekstmining en in het geval van webdata over webmining. Business performance management wordt gebruikt om doelen te stellen en standaarden te bepalen. Business performance management controleert en meet de prestaties door gebruik te maken van de BI-methodologie (Turban & Liang, 2007).

De data waarover business intelligence taken uitgevoerd worden komen vaak van verschillende bronnen. Deze verschillende bronnen kunnen data bevatten van verschillende kwaliteit, waarbij men vaak geconfronteerd wordt met een inconsistent gebruik van formaat, code en voorstellingen. Met behulp van ETL-processen en complexe event processing engines worden de gegevens samengevoegd en opgeslagen op datawarehouse servers. ETL staat voor extraheren, transformeren en laden. Naast de datawarehouse servers worden er mid-tier servers voorzien die gespecialiseerde functionaliteiten aanbieden afhankelijk van het BI-scenario, zoals servers voor online analytic processing (OLAP), reporting-servers om rapporten te genereren, enterprise search engines voor de ondersteuning van zoekopdrachten via sleutelwoorden en datamining engines. Er zijn verschillende front-end applicaties op de markt die de gebruikers toelaten om hun BI-taken uit tevoeren, gaande van spreadsheets, portals voor zoekopdrachten, performance management applicaties, ad hoc query tools en visualisatietools voor datamining modellen (Chaudhuri et al., 2011).
Figuur twee toont een overzicht van de typische architectuur voor het ondersteunen van een business intelligence systeem. Daarnaast bestaan er nog andere BI-technieken zoals Web analytics, die niet opgenomen zijn in figuur twee.



Figuur 2 Business intelligence architectuur

3.2.1. Datawarehouse

Een datawarehouse is een verzameling van onderwerp-georiënteerde, niet-vluchtige, geïntegreerde en tijdsgebonden data, ter ondersteuning van managementbeslissingen (Claus, 2001 en Albano, 2013).

Onderwerp-georiënteerd

Een datawarehouse slaat data op per onderwerp, niet per applicatie. Een operationele database daarentegen slaat de gegevens op per applicatie. Deze onderwerpen zijn afhankelijk van de organisatie.

Niet vluchtig

De data in een datawarehouse dienen voornamelijk voor query's en analyse. Ze worden nooit interactief gewijzigd. Men kan nieuwe data periodisch toevoegen of verwijderen, maar het wijzigen van data is niet toegelaten.

Geïntegreerd

Data vanuit verschillende bronnen worden verzameld in een datawarehouse. Deze data worden samengevoegd in een coherent geheel. Omwille van het gebruik van verschillende bronnen zijn de data vaak op verschillende manieren geformatteerd en gecodeerd. Het is niet meer dan logisch dat er behoefte is aan een uniforme integratie en definiëring van de data.

Tijdsgebonden

De data in een datawarehouse zijn bedoeld voor analyse en ondersteuning van besluitvorming. Een datawarehouse bevat bijgevolg zowel recente gegevens als historische data die geïdentificeerd zijn met een bepaalde tijdsperiode.

Mate van detail van de datawarehouse

De mate van detail van de datawarehouse is een van de belangrijkste aspecten bij het bouwen van een datawarehouse. Het is een van de eerste fundamentele beslissingen die men moet nemen bij het bepalen van de betekenis van een record in de datawarehouse. Indien men kiest voor een hoge mate van detail, moet men vaak een enorme hoeveelheid data opslaan. Bij een lagere mate van detail loopt men het risico dat men geen gedetailleerde query's meer kan uitvoeren.

In de meeste organisaties is er zowel behoefte aan efficiëntie bij het opslaan van data als aan de mogelijkheid gedetailleerde gegevens te analyseren. Om hiervoor te zorgen kan men twee of meerdere niveaus van detail overwegen. Volgens W.H. Inmon (1996) zouden twee niveaus van detail zelfs standaard moeten zijn.

Extract transform load (ETL)

Bij het bouwen van een datawarehouse maakt men vaak gebruik van het extraheren, transformeren en laden van data. Het ETL-proces is een belangrijke component in elk data-centralisatie project. Aangezien incorrecte of misleidende data slechte beslissingen kunnen veroorzaken is een correct design van ETL-processen noodzakelijk in de beginfase van het bouwen van een datawarehouse. (Trujillo & Lujan-Mora, 2003)

3.2.2. Business analytics

Business intelligence omvat de verzameling van data en informatie vanuit verscheidene bronnen, het organiseren en opslaan van deze data in een datawarehouse en het analyseren en het gebruiken van deze gegevens voor besluitvoering. Business analytics staat in voor de modellen en de technieken voor het analyseren van de data. Business analytics verwijst dus naar de technologieën, systemen en applicaties om belangrijke data te analyseren en een organisatie te helpen bij de besluitvoering (Turban & Liang, 2007).

Web analytics

Web analytics is de verzameling, analyse, meting en rapportering van internet data om webgebruik te begrijpen en verbeteren. De internet data kunnen onder andere verzameld worden met behulp van clickstreams. De analyse gebeurt met behulp van datawebhouses. Voor het meten maakt men gebruik van webmaatstaven. Een webmaatstaaf is een kwantitatieve meting van een aantal feiten op een website. Het rapporteren ten slotte gebeurt met behulp van front-end tools (Clifton, 2010).

3.2.3. Datamining

Datamining is het proces dat statistische, mathematische, artificiële intelligentie en machine learning technieken gebruikt om bruikbare informatie en bijgevolg kennis te verkrijgen vanuit grote databases. De term datamining wordt ook gebruikt om het proces te beschrijven waarin niet triviale, onbekende patronen in data gevonden worden. Dit proces gaat op zoek gaat naar patronen die niet eerder vermoed werden. De algoritmes waar datamining gebruik van maakt, gaan verder dan de aggregatiefuncties in relationele databasemanagementsystemen en OLAP servers. Dergelijke analyses omvatten onder andere beslissingsbomen, market-basketanalyse en lineaire en logistieke regressie (Claus, 2001).

Tekstmining is de toepassing van datamining op ongestructureerde of minder gestructureerde tekstbestanden. Documenten hebben zelden een sterke interne

infrastructuur. Wanneer ze deze toch hebben, is deze vaak gebaseerd op het formaat van het document en niet op de inhoud van het document (Srivastava & Sahami, 2009).
Webmining wordt omschreven als het ontdekken en analyseren van interessante en bruikbare informatie van en over het web. Voor webmining wordt vaak gebruik gemaakt van webgebaseerde tools (Scime, 2005).

Tekstmining

Tekstmining is een veel gebruikte discipline waarin methodes als machine learning gehanteerd worden. Deze leunen aan bij het onderzoeksdomein van artificiële intelligentie en die gerelateerd zijn met statistiek. Door middel van complexe algoritmes haalt men niet-triviale patronen of kennis uit tekstdocumenten. Via natural language processing software tracht men de inhoud van een tekst te analyseren en te begrijpen (Hearst, 2003).
In Tan (1999) wordt een algemeen kader gegeven voor tekstmining dat bestaat uit twee componenten: tekstverfijning dat de vrije-vorm tekstdocumenten omzet in een tussenvorm en kennisdestillatie dat patronen of kennis haalt uit de tussenvorm. Dit voorgestelde kader wordt dan gebruikt om tekstmining producten en applicaties te bestuderen. Zo onderscheidt men een aantal producten die focussen op organisatie en visualisatie van documenten en navigatie tussen documenten. In het algemeen organiseren ze documenten gebaseerd op hun overeenkomsten en stellen ze clusters van de documenten voor in zekere grafische voorstellingen. Voorbeelden van dergelijke producten zijn Cartia's ThemeScape, Canis's cMap, Synthema, VizControls en SemioMap. Andere tekstmining producten focussen dan weer op tekstanalysefuncties zoals het verkrijgen, categoriseren en samenvatten van informatie. Voorbeelden van dergelijke producten zijn Knowledge Discovery System's Concept Explorer, IBM's Intelligent Miner, LinguistX, NetQuestion Solution, Tekstwise, DR-Link, Cindor en Chess. Volgens Tan is er onder andere nog verder onderzoek nodig naar de tussenvorm, naar multi-linguale tekstverfijning en naar de integratie van domeinkennis.

Literatuurmining

Literatuurmining is een populair applicatiegebied voor tekstmining, waarbij grote verzamelingen literatuur in een specifiek domein verwerkt worden via semi-automatische methodes om nieuwe patronen te ontdekken (wps.prenhall.com). PubMed is een gratis dienst voor het doorzoeken van literatuur op het web, ontwikkeld en onderhouden door het Nationaal Centrum voor Biotechnologische

Informatie (NCBI). PubMed bevat citaten en abstracten van meer dan 5000 life science journals voor biomedische artikels vanaf 1948. Ook al levert PubMed een brede, up-to-date en efficiënte zoekinterface, het wordt steeds uitdagender voor zijn gebruikers om snel relevante informatie te vinden voor hun individuele noden. Dit komt vooral door de altijd groeiende biomedische literatuur (Lu, 2011). PubMatrix is een web-gebaseerde tool die simpele tekst-gebaseerde mining van PubMed gebruikt. PubMatrix kan aan de hand van twee lijsten met sleutelwoorden een frequentiematrix van het samen voorkomen van deze termen geven (Becker et al., 2003). De PubMed webdiensten worden vooral gebruikt in het domein van bio-informatica. De kwaliteit van verschillende andere web tools, die vergelijkbare diensten leveren als PubMed om literatuur te doorzoeken is sterk toegenomen. Dit is te wijten aan de ontwikkeling van web tools om gebruikers sneller en doeltreffender te laten zoeken naar relevante publicaties en aan de volwassenheid van tools op het gebied van tekstmining. In de studie van Lu (2010) worden achtentwintig dergelijke tools vergeleken.

Webmining

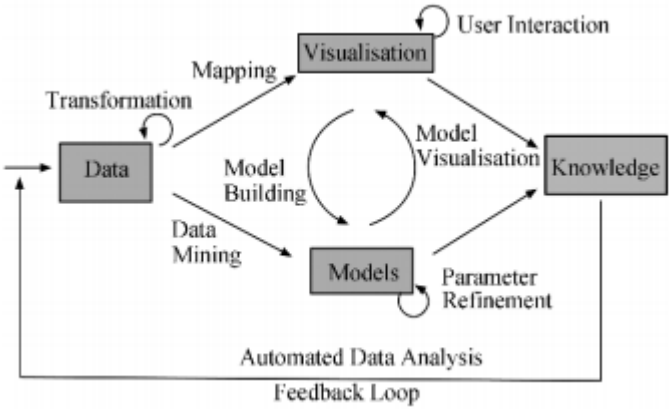
Om op een geautomatiseerde manier bruikbare kennis te kunnen extraheren uit de talloze webpagina's past men binnen het domein van webmining steeds nieuwere technieken toe, wat gezien het heterogene karakter en het ontbreken van structuur in de informatie, een grote uitdaging is. Web usage mining ontrafelt patronen aan de hand van de toegangswegen waarlangs gebruikers het Internet exploreren. Web structure mining tracht zinvolle informatie te halen uit de structuur van de hyperlinks en kan helpen om webpagina's te rangschikken naar belang, invloed en onderwerp. Web content mining heeft als doel bruikbare kennis uit de inhoud van de webpagina's te halen en is daarom relevant voor academische onderzoekers. Via technieken als clustering en classificatie kunnen webpagina's op een multidimensionale manier gegroepeerd worden op basis van hun inhoud (Fürnkranz, 2002).

3.2.4. Visuele analyse

Een effectieve analyse om waardevolle inzichten te verwerven uit grote hoeveelheden van informatie maakt het mogelijk om gefundeerde beslissingen te nemen en om strategieën uit te werken. Verschillende methodes die gebruik maken van complexe algoritmes zoals datamining worden gebruikt om de data op een automatische manier te analyseren. Alhoewel deze technieken hun nut hebben bewezen in verschillende toepassingen, zijn er toch nog een aantal grote uitdagingen, zoals de schaalbaarheid van de algoritmes, de toenemende volumes

van de data en het analyseren van heterogene data. Bovendien zijn deze methodes niet geschikt voor alle soorten van analyses. De interactieve inbreng van de kennis van de gebruikers is vaak nodig om de toegepaste methodes te verfijnen. Indien interessante maar complexe patronen ontdekt worden, is het vaak moeilijk om deze te begrijpen en te interpreteren. Om aan deze uitdagingen tegemoet te komen is de visuele analyse methode in de recente jaren uitgegroeid tot een combinatie van automatische analyse met interactieve visualisaties (Sun et al., 2013). De grote troef van visuele analyse is dat het de capaciteiten tot interceptie van het menselijk brein betrekt in het proces van de verkenning van de informatie (Hotho, 2005, in Sun et al., 2013).

Visuele analyse heeft een grote toegevoegde waarde in het proces van de gegevensanalyse en voor het analyseren van grote databanken. Het is een nuttige techniek in het geval er weinig gekend is over de data en wanneer het doel dat men wil bereiken eerder vaag is (Keim, 2002, in Sun et al., 2013).
Shneiderman et al. (1996, in Sun et al., 2013) stelden het proces van visuele analyse voor in drie stappen, ook wel het Information Seeking Mantra genoemd: "Overview first, zoom/filter, details on demand". Eerst krijgt de gebruiker een overzicht van de gegevens. De gebruiker kan interessante patronen ontdekken en focussen op één of meerdere patronen. Om de patronen te analyseren, moet de gebruiker over de mogelijkheid van 'drill-down' beschikken om van een algemeen niveau naar een meer gedetailleerd niveau te kunnen afdalen en moet hij toegang hebben tot de betreffende detailinformatie. Een alternatieve methode is om het visuele overzicht te verstoren (distortion) met detailinformatie van interessante gedeeltes (Keim, 2002, in Sun et al., 2013).
Nadien werd dit mantra nog uitgebreid door Keim et al. (2006, in Sun et al., 2013) en werd de nadruk gelegd op het verwerven van inzichten bij de analyse: "Analyze first, show the important, zoom/filter, analyze further, details on demands". Deze nieuwe mantra vestigt de aandacht op de combinatie van algoritmische data-analyse en interactieve visuele interfaces. Keim et al. (2010, in Sun et al., 2013) introduceerden een framework van het proces van visuele analyse. Figuur drie illustreert dit volledige proces.



Figuur 3 Visual analytics process (Keim et al., 2010)

Het proces vangt aan met het transformeren van de gegevens door middel van het filteren en het nemen van steekproeven voor verdere analyse. Vervolgens wordt afzonderlijk een visuele analyse, waarbij de gebruikers interactief reageren op de visuele interface, en een automatische analyse methode zoals datamining toegepast.
Visuele analyse kan men ook beschouwen als een proces bestaande uit vier stappen: verzamelen van de gegevens, transformeren van de data, mappen op grafische attributen en afbeelden.
Om een visualisatie van hoge kwaliteit te bekomen moet de informatie die afgebeeld wordt accuraat, volledig, voldoende, intuïtief en interactief zijn (Ward et al., 2011, in Sun et al., 2013).

Recent werden nog tal van andere modellen geïntroduceerd in het bijzonder om de ontwikkeling van visuele analyse systemen te vereenvoudigen. Data-Driven Documents (D3) is bijvoorbeeld een model voor de snelle ontwikkeling van visualisaties van online data (Bostock et al., 2011, in Sun et al., 2013). Het maakt het mogelijk om elementen van documenten rechtstreeks te manipuleren en biedt de mogelijkheid om animaties en gebruikersinteracties in te bouwen. WebCharts is een nieuw platform voor visualisaties dat een snelle ingebruikname en hergebruik van bestaande code mogelijk maakt (Fisher et al., 2010, in Sun et al., 2013).

In de paper van Keim (2002, in Sun et al., 2013) worden de verschillende technieken geclassificeerd aan de hand van het te visualiseren datatype, de visualisatietechniek, de interactietechniek en de techniek van distortion.

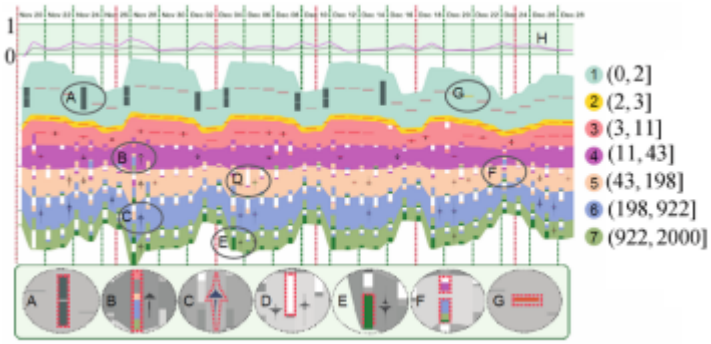
Aan de hand van het type van de te visualiseren data onderscheidt men zes klassen. De zes verschillende klassen zijn één-dimensionale data, twee-dimensionele data, multidimensionale data, tekst en hypertekst, hiërarchie en grafieken en ten slotte algoritmes en software.
De verschillende visualisatietechnieken worden dan weer opgedeeld in standaard 2D- of 3D-displays, geometrisch getransformeerde displays, icoon-gebaseerde displays, dichte pixel-displays en gestapelde displays.
Interactietechnieken en de techniek van distortion maken het voor gebruikers mogelijk om direct in te werken op de visualisaties. Hier onderscheidt men dynamische projectie, interactieve filtering, interactieve zooming, interactieve distortion en interactieve linking en brushing.
Dynamische projectie wordt gebruikt voor het verkennen van een multi-dimensionele dataset.
Interactieve filtering is interessant om grote datasets te analyseren, doordat men de dataset interactief kan opdelen in segmenten en focussen op de interessante deelgebieden. Dit kan door middel van een directe selectie van de gewenste set of via een query op kenmerken van de gewenste dataset.
De techniek van interactieve zooming laat toe om in te zoomen op een bepaalde selectie van de gegevens. Deze wordt dan niet enkel in een groter formaat getoond maar er worden ook meer details weergegeven.
De methode van visuele distortion biedt middelen aan om te focussen op details terwijl men tegelijkertijd het overzicht over de data bewaart. Het idee is om gedetailleerde informatie te tonen van een gedeelte van de data terwijl van de overige gegevens minder details worden getoond.
Onder interactieve linking en interactieve brushing vallen de visualisatiemethodes die verschillende technieken combineren om met de tekortkomingen van de afzonderlijke technieken te kunnen omgaan (Keim, 2002).

Sun et al. (2013) kwamen in hun studie tot vijf verschillende categorieën van applicaties die gebruik maken van visuele analyse: ruimte en tijd, multivariate data, tekst, grafieken en netwerk en ten slotte overige applicaties. Deze categorieën geven niet alleen een overzicht van de applicaties maar zorgen tevens voor een differentiatie op gebied van recente research.

Ruimte en tijd

Geospatiale data, tijdgerelateerde gegevens en de combinatie van beide zijn alomtegenwoordige datatypes in visuele analyse (Keim et al., 2008). Het ontdekken van patronen en relaties met betrekking tot locatie en tijd is een

vereiste in vele analyse taken (Keim et al., 2010). Door de enorme hoeveelheid en door de complexiteit van deze data brengt het realiseren van een effectieve analyse grote uitdagingen met zich mee, die geavanceerde technologie vereisen zowel wat betreft rekenkracht als visualisaties (Sun et al., 2013). Slingsby, Dykes en Wood (2011, in Sun et al., 2013) stellen een interactieve visuele analyse systeem voor om de resultaten van OAC te onderzoeken. OAC is een geodemografisch classificatiesysteem dat de bevolking in het Verenigd Koninkrijk indeelt in verschillende gebieden in een hiërarchie van drie niveaus en dit aan de hand van 41 demografische variabelen. BallotMaps is een interactieve grafische tool voor de analyse van onder andere geografische data gebaseerd op hiërarchisch georganiseerde grafieken. CloudLines maakt gebruik van een nieuwe compacte visuele metafoor om tijdreeksen voor te stellen. ChronoLenses biedt verschillende 'lenzen' om belangstellingsgebieden binnen tijdreeksen verder te onderzoeken. TimeSeer is een visualisatietool voor de analyse van tijdreeksen in meerdere dimensies. Dergelijke tijdreeksen, zoals financiële en economische multivariate gegevens, komen vaak voor in ons dagelijks leven maar vormen een uitdaging voor analisten. RankExplorer is een nieuwe techniek voor visuele analyse die verschillende zaken combineert en gebruik maakt van ThemeRiver voor het onderzoeken van wijzigingen in de volgorde in grote tijdreeksen (Sun et al., 2013). Figuur vier toont een voorbeeld van een visualisatie met behulp van RankExplorer.



Figuur 4 Visualisatie van de top 100 Bing search queries met RankExplorer (Shi et al., 2012, in Sun et al., 2013)

Multivariate data

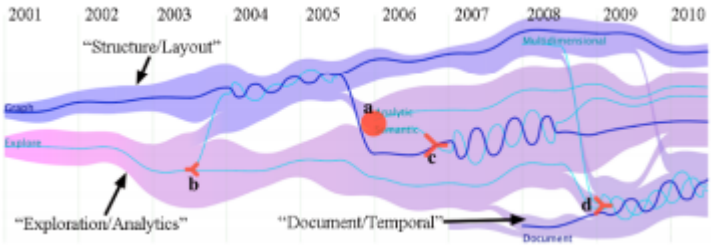
Verscheidene methodes worden aangewend om de relaties en de samenhang te ontdekken in data van verschillende dimensies. Ze kunnen ingedeeld worden in twee brede categorieën. Methodes die 'projection-based' zijn gebruiken technieken om de dimensie te verkleinen. Visuele methodes richten zich op visuele lay-outs. Local affine multidimensional projection (LAMP) is een nieuwe

methode gebaseerd op een orthogonale theorie voor het in kaart brengen van data van meerdere dimensies. LAMP is een efficiënte tool en biedt de gebruikers de mogelijkheid om de resultaten progressief te verfijnen met hun kennis. LAMP werd bijvoorbeeld gebuikt voor het projecteren van de verbanden dus beelden en muziek (Sun et al., 2013).

Tekst

Dagelijks worden enorme hoeveelheden tekst geproduceerd, verzameld en opgeslagen. Deze gegevens zijn meestal ongestructureerd. Voor de analyse worden algoritmes gebruikt om de tekst om te zetten naar gestructureerde informatie. Men kan een onderscheid maken tussen methodes die 'topic-based' zijn en methodes die 'feature-based' zijn (Sun et al., 2013).

Topic-based methodes halen onderwerpen uit teksten en analyseren de informatie door gebruik te maken van verschillende visualisatietechnieken. De tijdgebonden informatie die gelinkt is aan de documenten is een belangrijk onderdeel om te gebruiken bij het analyseren van de data (Kwon et al., 2012). TextFlow integreert technieken van datamining gebaseerd op topics in de interactieve visualisaties om zo de evolutie in de tijd visueel te kunnen analyseren. TextFlow gebruikt enkele tekstmining algoritmes om trends in de evolutie van topics te modelleren en om verbanden te ontdekken tussen sleutelwoorden. Om inzichten te verkrijgen uit de massa aan tekstuele data biedt Textflow drie visuele overzichten aan: een topicflow, een tijdlijn en een woordenwolk (Cui et al., 2011, in Sun et al., 2013). Figuur vijf toont een voorbeeld van een visualisatie met behulp van Textflow.



Figuur 5 Visualisatie van onderzoekevolutie geïllustreerd door het samenvoegen en splitsen van patronen van topics over de tijd door Textflow (Cui et al., 2011, in Sun et al., 2013).

Feature-based methodes maken gebruik van verschillende functies op het niveau van een woord of op het niveau van een document om tekst te visualiseren. Woordenwolken worden algemeen gebruikt. Ze geven een intuïtieve visuele samenvatting van documenten door de sleutelwoorden te tonen in een compacte lay-out. Sleutelwoorden die vaker voorkomen in de brontekst worden in een groter formaat afgebeeld. Verschillende algoritmes zoals Wordle en ManiWordle

worden voorgesteld om goede woordenwolken te maken. De semantische relatie tussen de verschillende sleutelwoorden in de oorspronkelijke tekst gaat bij deze voorstellingswijze wel verloren. Onderzoekers hebben 'force-based' algoritmes en 'seamcarving' algoritmes geïntroduceerd om woordenwolken te creëren die de context bewaren. Hierdoor worden woorden, die samen voorkomen in de oorspronkelijk tekst, dicht bij elkaar geplaatst in de woordenwolk (Sun et al., 2013). Figuur zes toont een voorbeeld van een woordenwolkvisualisatie.

Figuur 6 Visualisatie van dynamische tekst met behulp van een context behoudende woordenwolkvisualisatie (Kwon et al., 2012).

Grafieken en netwerk

De technieken voor de visuele analyse van grafieken en netwerken kan men indelen in twee categorieën: 'graph layout methods' en 'clutter reduction methods'. Ondanks het feit dat de ontwikkelingen in dit onderzoeksgebied elkaar snel opvolgen, blijft het nog lastig om grote grafieken te visualiseren zonder visuele vervuiling (Sun et al., 2013).

Storytelling

De methode van storytelling is een nieuwe tendens binnen de visuele analyse en heeft de laatste jaren veel aandacht gekregen. De meeste applicaties voor visuele analyse beschikken over een mogelijkheid om rapporten aan te maken van de bevindingen van de analyse. Interactieve verhalende visualisaties zijn beter geschikt om deze bevindingen op een effectieve manier over te brengen dan de klassieke rapporten, omdat ze diepere inzichten kunnen verschaffen en uitleg kunnen geven bij de ontdekte patronen. Maar het nadeel van deze methode is dat ze bij definitie heuristisch en subjectief is (Tanashi & Ma, 2012, Liu et al., 2013 en Hullman et al, 2013, in Sun et al., 2013).

3.3. Business intelligence voor literatuuronderzoek

Er bestaan verschillende applicaties voor literatuuronderzoek in het domein van tekstmining en literatuurmining. Zo is Hubmed een alternatieve interface voor de

PubMed database. HubMed haalt informatie direct van PubMed via de Entrez hulpprogramma's van NCBI. Hubmed biedt innovatieve manieren om te browsen en de literatuur te verkennen. In de toekomst zouden nog meer HubMed features toegevoegd moeten worden zoals het visueel voorstellen of clusteren van zoekresultaten en het incorporeren van digital object identifiers of kortweg DOI en 'geciteerd door' links ingebouwd in andere bibliografische databases. Een digital object identifier is een uniek en permanent blijvend identificatiemiddel voor een bestand op het Internet en dit is niet afhankelijk van het internetadres (Von Isenburg, 2007).

De Agilent Literature Search software is een meta search tool voor het automatisch bevragen van meerdere tekst-gebaseerde zoekmachines. Deze tool wordt gebruikt door biologen en helpt hun bij het manueel zoeken en extraheren van associaties tussen de genen of proteïnes, waar ze in geïnteresseerd zijn.

De software kan gebruikt worden samen met Cytoscape, dat hulp biedt bij het genereren van een netwerkoverzicht van deze associaties. De software biedt een gemakkelijk te gebruiken interface ("Agilent Literature Search" en "Agilent Literature Search Software").

Het interessante aan tools als Hubmed en Agilent Literature Search is dat ze als het ware een laag toevoegen en een eigen interface bieden om bestaande zoekmachines te benaderen en de gevonden resultaten te ontsluiten. In die zin kan men deze tools ook deels onderbrengen in het domein van de visuele analyse. Dergelijke tools kunnen zonder gebruik te maken van ingewikkelde algoritmes toch een wezenlijke bijdrage leveren aan het ontsluiten van bruikbare kennis.

ProQuest ten slotte levert informatie en technologie die de productiviteit van studenten, professionals en bibliotheken zouden moeten verhogen. ProQuest zorgt voor rijke vaste en gevarieerde informatie. ProQuest levert verschillende producten: onder andere ProQuest Central. ProQuest Central gebruikt achtentwintig van de meest gebruikte databases van ProQuest om de grootste academische onderzoeksbron te vormen die vandaag beschikbaar is. Onderzoekers kunnen kiezen om de individuele databases af te zoeken voor een bepaald onderzoek of alle databases tegelijk te gebruiken om relevante informatie en resultaten te vinden van andere disciplines ("ProQuest", 2015). ProQuest geeft ook een overzicht van de gevonden publicaties in de tijd aan de hand van een grafiek. Het aantal publicaties wordt hierin weergegeven per jaar. Wanneer men klikt op een jaar, wordt de selectie van de publicaties gefilterd en krijgt men een nieuwe grafiek van de publicaties van het geselecteerde jaar

ingedeeld per maand. Ook hier kan men weer op doorklikken en dan krijgt men een overzicht per dag.

3.4. Evaluatie en interpretatie

In de literatuurstudie komt tot uiting dat business intelligence in eerste instantie dient om bedrijfsinformatie te verfijnen en het management toe te laten om doeltreffende bedrijfsbeslissingen te nemen. Vanuit deze optiek vertrekt men meestal van een datawarehouse systeem. In de context van deze thesis is deze definitie te eng. De informatie voor literatuuronderzoek wordt van het Internet gehaald en in die zin is er eerder een raakvlak met webmining. Web content mining heeft als doel bruikbare kennis te halen uit de inhoud van de webpagina's en is daarom relevant voor academische onderzoekers. Het is echter niet de bedoeling van deze thesis om algoritmes te ontwikkelen en webmining of tekstmining technieken toe te passen om kennis te halen uit informatie op het Internet.

De voor deze thesis te ontwikkelen BI-tool maakt gebruik van bestaande zoekmachines en heeft als doel meerwaarde te bieden aan onderzoekers door het visualiseren van relatief eenvoudig te achterhalen metagegevens zoals bijvoorbeeld tijdsgebonden informatie. De tijdgebonden informatie die gelinkt is aan de documenten is een belangrijk onderdeel om te gebruiken bij het analyseren van de data (Kwon, Javed, Ghani et al., 2012).

De meeste bestaande tools voor opzoeken van informatie voor literatuuronderzoek maken gebruik van bestaande zoekmachines en bieden de gebruiker de mogelijkheid om te zoeken over verschillende databases. Ze doen dit aan de hand van een eigen interface waarin men extra filtercriteria kan toevoegen en waarin bijkomende informatie omtrent de gevonden publicaties weergeeft zoals de auteur van het artikel en de journal waarin het artikel gepubliceerd werd. De tools maken echter geen gebruik van visualisatietechnieken die kunnen helpen bij het analyseren van de gevonden publicaties. ProQuest biedt wel al een grafiek aan die de evolutie in de tijd van de gevonden publicaties duidelijk maakt en waarop men de informatie verder kan filteren.

Een groot deel van de literatuurstudie handelt over visuele analyse en geeft een beeld van de mogelijkheden die vandaag bestaan voor het visualiseren van grote hoeveelheden aan informatie. Voornamelijk het deel omtrent het visualiseren van informatie uit tekst is daarbij van belang voor dit onderzoek. Tijdlijnen en woordenwolken zijn maar enkele van de technieken die hierbij veel gebruikt worden. Feature-based methodes maken gebruik van verschillende functies op

het niveau van een woord of op het niveau van een document om tekst te visualiseren. Woordenwolken worden algemeen gebuikt. Ze geven een intuïtieve visuele samenvatting van documenten door de sleutelwoorden te tonen in een compacte lay-out. Sleutelwoorden die vaker voorkomen in de brontekst worden in een groter format afgebeeld (Sun et al., 2013).

In de recente jaren is visuele analyse uitgegroeid tot een combinatie van automatische analyse met interactieve visualisaties (Sun et al., 2013). De recente toepassingen van visuele analyse zijn complex en vereisen zowel datamining als geavanceerde visualisatie technieken, welke niet realiseerbaar zijn binnen de beperkte opzet van een prototype.

4. Ontwikkeling en beschrijving van de BI-tool

4.1. Requirements

In dit onderdeel worden de vereisten in kaart gebracht voor het te ontwikkelen prototype. De verworven inzichten uit de afgenomen interviews en het literatuuronderzoek worden omgezet in een prototype.

Uit de studie van de literatuur blijkt dat een vaak voorkomende fout bij een literatuurstudie is dat de auteur niet voldoende tijd besteedt om de beste auteurs en bronnen te identificeren in verband met zijn onderwerp (Gall, Borg en Gall, 1996). Randolph (2009) merkt dan weer op dat elektronische zoektochten slechts tien procent van de relevante artikels leveren. De te ontwikkelen tool kan hierin een bijdrage leveren.
Uit de interviews blijkt voorts dat het voor onderzoekers niet eenvoudig is om geschikte publicaties op te zoeken en om te bepalen welke papers over een bepaald onderwerp belangrijk zijn en welke auteurs van belang zijn. De geïnterviewde personen gaven aan dat het aanbieden van de volgende functies wenselijk is:

- een overzicht van aanverwante onderwerpen;
- de evolutie van de onderzoeksinteresse in een onderwerp;
- een lijst van de belangrijkste journals.

Dat de belangrijkste bijdrages vaak gevonden worden in de voornaamste journals, wordt ook bevestigd in het literatuuronderzoek (Webster en Watson, 2002).
Andere optionele informatie zoals de aanduiding of een volledig artikel beschikbaar is en de prijs om toegang te krijgen tot een artikel worden niet weerhouden in de tool. De reden hiervoor is dat deze informatie minder geschikt is om verbanden weer te geven tussen de gevonden artikels en ook geen aanduiding geeft van het belang van het artikel voor het onderwerp. Het gaat enkel over praktische info die op zich wel nuttig is. Een overzicht van de belangrijkste universiteiten bij een topic is dan weer wel relevant en verschaft ook een inzicht over het belang van een artikel. Deze vereiste is niet weerhouden omdat de informatie niet eenvoudig te achterhalen is en omdat dit in verhouding te veel tijd zou vergen, tijd die ontbreekt binnen de scope van een eerste prototype. De mogelijkheid om specifiek naar reviewartikels te zoeken is niet

weerhouden omdat het hier enkel om een bijkomend filtercriterium gaat. Deze filtermogelijkheid is bovendien wel al aanwezig in ProQuest.

Uit de literatuurstudie omtrent business intelligence is gebleken dat de te ontwikkelen tool aansluiting vindt bij het onderzoeksgebied rond webmining. De tool kan een aanvulling zijn door ideeën uit de visuele analyse te integreren bij het weergeven van de resultaten van een internetzoekopdracht in het kader van een literatuurstudie. Een doelstelling voor de te ontwikkelen tool is om de gebruiker inzichten te verschaffen in de samenhang tussen de beschikbare informatie in verband met de gevonden artikels en dit door een intuïtieve visuele samenvatting te geven van deze gegevens. Aangezien het gaat om tekstuele informatie wordt er gebruik gemaakt van feature-based methodes op het niveau van een document om tekst te visualiseren. Concreet kunnen de aanverwante onderwerpen voorgesteld worden in een woordenwolk waarbij de kernwoorden die veel voorkomen in een groter formaat worden afgebeeld. Om de contextuele samenhang tussen de aanverwante onderwerpen te bewaren zou gebruik gemaakt kunnen worden van 'force-based' algoritmes en 'seamcarving' algoritmes om de onderwerpen die samen voorkomen dicht bij elkaar te plaatsen, maar dit is binnen de scope van deze thesis niet realiseerbaar. Om de evolutie van de onderzoeksinteresse in een onderwerp in de tijd af te kunnen leiden wordt gebruik gemaakt van een tijdlijn. Deze laatste techniek wordt ook reeds gehanteerd in ProQuest, maar in ProQuest worden de aanverwante onderwerpen en de journals niet getoond en dus ook niet mee gefilterd bij het selecteren van een periode uit de tijdlijn.

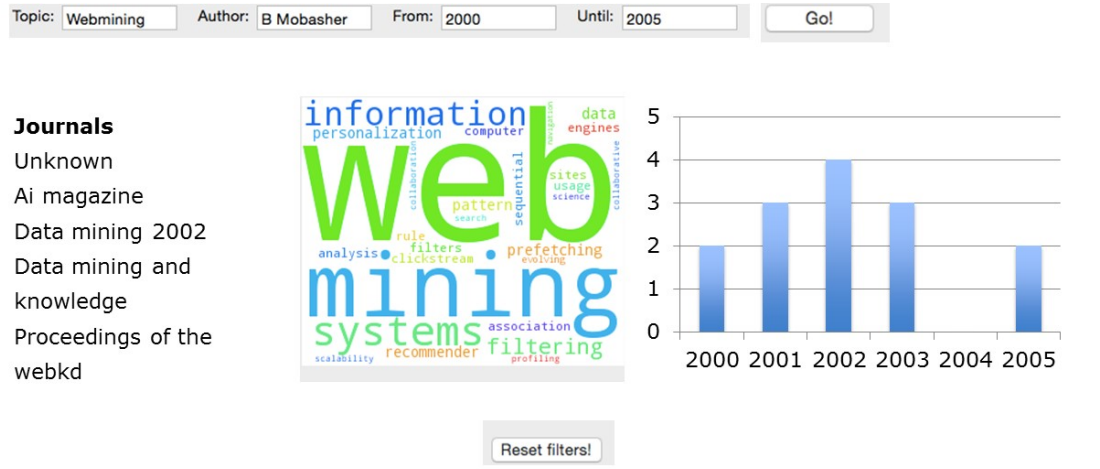
De techniek van interactieve filtering zal worden toegepast. Bij filtering op basis van één van de criteria, zullen de overzichten van al de items mee gefilterd worden. Indien men bijvoorbeeld een bepaalde journal selecteert, zal het overzicht met de betreffende aanverwante onderwerpen ook gefilterd worden op basis van het geselecteerde journal. De techniek van interactieve zooming zal in het prototype nog niet worden uitgewerkt. Deze techniek houdt immers in dat men bij het inzoomen op een bepaalde selectie, hiervoor meer detailinformatie gaat ophalen en tonen. Zo zou men bij het selecteren van bepaalde journal meer details kunnen tonen van de geselecteerde artikels van de journal. Dit is echter geen vereiste in deze context. Informatieve zooming is vooral interessant om inzicht te verwerven bij zeer grote datasets.

Vereisten voor de BI-tool

1. De gebruiker kan zoeken naar wetenschappelijke publicaties die achterliggend opgezocht worden via de Google Scholar zoekmachine.
2. De gebruiker heeft de mogelijkheid om een artikel op te zoeken met als zoekcriterium het onderwerp, de auteur, een jaartal of een combinatie van deze drie criteria. Dit zijn de criteria die ook voor een Google Scholar zoekopdracht ingegeven kunnen worden.
3. De geavanceerde zoekmogelijkheden die in Google Scholar voor handen zijn, zoals bijvoorbeeld het zoeken op een exact zinsdeel of het zoeken op een woord in de titel, zijn ook mogelijk via de tool, maar wel enkel door het intikken van de correcte zoekopdracht in de zoekbalk. Er wordt geen bijkomend scherm voorzien om het geavanceerd zoeken te ondersteunen.
4. De gevonden publicaties als gevolg van een zoekopdracht worden weergegeven in een overzicht. Deze lijst bevat de volgende informatie:
    - de titel;
    - de auteur;
    - het jaar van uitgave;
    - en de url waar het oorspronkelijke document terug te vinden is.
5. Er wordt een overzicht getoond van de journals waarin de gevonden publicaties werden gepubliceerd.
6. De aanverwante onderwerpen worden getoond in een woordenwolk waarbij de kernwoorden groter worden afgebeeld naargelang het aantal keren dat ze voorkomen in de oorspronkelijke documenten.
7. De evolutie van het aantal voorkomens in de tijd van de documenten betreffende het onderwerp wordt visueel weergegeven in een tijdlijn.
8. De gebruiker beschikt over een aantal mogelijkheden om de gegevens in het getoonde overzicht te filteren. De gebruiker kan filteren:
    - op een bepaald jaar van uitgave;
    - op een aanverwant onderwerp:
    - op een journal.

4.2. Ontwerp van de gebruikersinterface

Figuur zeven toont het ontwerp van de gebruikersinterface van het prototype van de BI-tool.



Title	Authors	Year	Journal	Link
Measuring the accuracy of sessionizers for web usage analysis	..., B Mobasher, M Spiliop...	2001	on web mining at	http://facweb.cti.depaul.edu/research/TechReports/TR01-006.pdf
Personalization	WUM Mobasher, WU Mining	2005	practical handb...	Unavaible
Webkdd 2005: web mining and web usage analysis post-works...	..., OR Zaiane, M Spiliopou...	2005	acm sigkdd	http://dl.acm.org/citation.cfm?id=1117475
Web usage mining for e-business applications	..., B Mobasher, M Spiliop...	2002	tutorial ecmmlp...	http://ecmlpkdd.cs.helsinki.fi/pdf/berendt-2.pdf
Web Mining: Information and Pattern Discovery on the Web	RMB Cooley, J Srivastava	2000	sigkdd explora ...	Unavaible
WiltshireJ. 2000. Discovery of aggregate usage profiles for we...	B Mobasher, H Dai, T Luo, ...	2000	proceedings of ...	Unavaible
Measuring the accuracy of sessionizers for web usage analysis	B Mobasher, B Berent, M ...	2001	proceedings of ...	Unavaible
Program Chairs and Organizing Committee	..., B Magnini, T ITC	Unknown	unknown	http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.67.5741&rep=re
Using sequential and non-sequential patterns in predictive web...	B Mobasher, H Dai, T Luo...	2002	data mining 20...	http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1184025
A hybrid web personalization model based on site connectivity	M Nakagawa, B Mobasher	2003	proceedings of ...	http://facweb.cs.depaul.edu/mobasher/research/papers/NM03b.pdf
Discovery and evaluation of aggregate usage profiles for web p...	B Mobasher, H Dai, T Luo, ...	2002	data mining an...	http://link.springer.com/article/10.1023/A:1013232803866
Effective personalization based on association rule discovery f...	B Mobasher, H Dai, T Luo, ...	2001	proceedings of ...	http://dl.acm.org/citation.cfm?id=502935
Web Usage Mining	B Berendt, B Mobasher, M ...	2003	unknown	http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.465.6549&rep=r
The workshop program at the nineteenth national conference ...	..., S Mannor, G Theocharo...	2005	ai magazine	http://www.aaai.org/ojs/index.php/aimagazine/article/viewArticle/1806
The impact of site structure and user environment on session r...	..., B Mobasher, M Nakaga...	2002	ining	http://link.springer.com/chapter/10.1007/978-3-540-39663-5_10
3rd International Workshop on Web Dynamics Part of the Works...	..., C Cardona, C Rojas, Y ...	Unknown	unknown	http://www.dcs.bbk.ac.uk/~ap/webDyn3/webdyn3_proceedings.pdf
A framework for the evaluation of session reconstruction heuri...	M Spiliopoulou, B Mobash...	2003	informs journal ...	http://pubsonline.informs.org/doi/abs/10.1287/ijoc.15.2.171.14445

Figuur 7 Ontwerp van de gebruikersinterface

Bovenaan het scherm is een zoekbalk aanwezig waarmee het onderwerp, de auteur en de jaartallen ingegeven kunnen worden. Men kan ervoor kiezen om enkel via het onderwerp of enkel via de auteur een zoekopdracht uit te voeren. Een zoekopdracht enkel via de periode wordt niet uitgevoerd, want dergelijk opdracht geeft geen zinvolle resultaten. Wanneer meerdere vakken worden ingevuld gaat het om een AND relatie. Dit is de standaard manier van zoeken via Google Scholar.

Naast de zoekbalk is er een knop voorzien om de zoekopdracht te starten.

Onderaan het scherm wordt een lijst getoond met de gevonden artikels. Deze lijst bevat de titel, de auteur(s), het jaar van uitgave en de url waar het artikel te vinden is. De lijst is gesorteerd volgens relevantie zoals bepaald door Google Scholar. De gevonden artikels worden met andere woorden in dezelfde volgorde weergegeven zoals door Google Scholar opgehaald werden.

Links van het scherm onder de zoekbalk wordt een lijst getoond van de journals waarin de artikels voorkomen. De journals zijn gesorteerd naargelang het aantal artikels dat er in voorkomt. De journals met de meeste artikels staan bovenaan de lijst.

Naast het overzicht van journals vindt men een woordenwolk van de aanverwante kernwoorden. De grootte van het lettertype weerspiegelt het aantal keer dat de het aanverwante kernwoord is voorgekomen in de zoekresultaten.
Rechts naast de woordenwolk wordt een tijdlijn getoond met een aanduiding van het aantal voorkomens van artikels per jaar.
In het midden van het scherm is een knop voorzien om de filters te resetten.

Wanneer de gebruiker op een aanverwant kernwoord in de gebruikersinterface klikt, dienen de resultaten gefilterd te worden. Wanneer men op een jaartal klikt in de tijdlijn dient dan weer de tijdperiode aangepast te worden. Enkel de resultaten van het betreffende jaar worden dan getoond. Ten slotte worden enkel de artikels van een bepaalde journal getoond wanneer men op een bepaalde journal klikt.

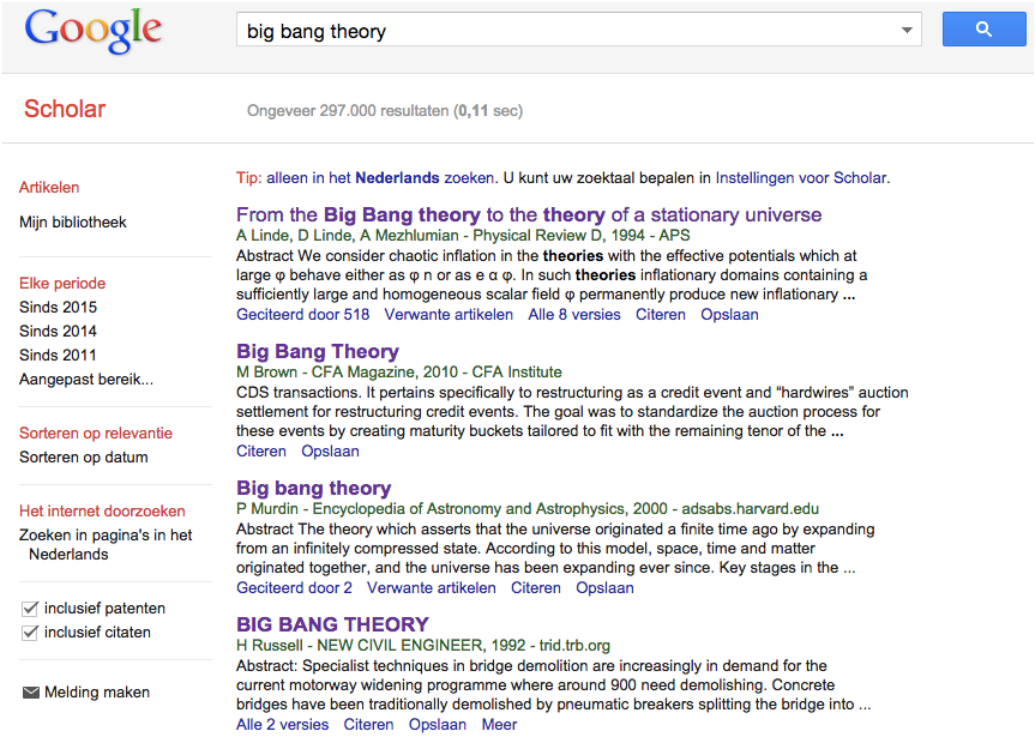
4.3. Keuze van omgeving en ontwikkelingstools

Het prototype in deze thesis laat enkel toe om zoekopdrachten uit te voeren op één zoekmachine. Het aanbieden van de mogelijkheid om zoekopdrachten uit te voeren, gebruikmakend van meerdere zoekmachines en databases is binnen de context van deze thesis niet realiseerbaar. Er is gekozen voor Google Scholar omdat dit voor wetenschappelijke literatuur de meest bekende en meest gebruikte zoekmachine is. Bovendien beperkt deze zich niet tot een bepaald vakgebied.
Voor dit werk werd de keuze gemaakt om te programmeren in Python. De reden hiervoor is dat Python een eenvoudige taal is die snel geleerd kan worden. Het is wel zo dat om de visualisatie van het scherm te realiseren een externe library gebruikt moet worden. Er is gekozen om te programmeren met behulp van Pycharm. PyCharm is immers een gratis ontwikkelomgeving voor open source projecten en voor educatief gebruik. Een voordeel van PyCharm is dat het werkt op Windows, Mac OS X en Linux. Tenslotte biedt PyCharm extensieve hulp tijdens het programmeren.

4.4. Haalbaarheidsonderzoek

Vooraleer de ontwikkeling van de eerder beschreven tool aangevangen werd, werd eerst nagegaan of al de informatie die gevisualiseerd moest worden beschikbaar was en opgehaald kon worden. Er werd dus nagegaan of de referentie van een paper kon opgehaald worden met behulp van zoekcriteria voor Google Scholar en of uit deze paper het jaartal, de journal, de aanverwante kernwoorden of andere informatie gehaald kon worden.

Als eerste werd nagegaan op welke manier de referentie van een paper gevonden kon worden. Hiervoor werd een Python module scholar.py gevonden die ontwikkeld werd door Christian Kriebich (“Scholar.py”, 2015).
Deze module haalt onder andere de publicatietitel, het publicatiejaar en de meest relevante weblink op. De Google Scholar zoekopdrachten die men met behulp van deze module kan uitvoeren hebben de mogelijkheid om een zoekwoord, een auteur en een tijdperiode te omvatten.



Figuur 8 Google Scholar zoekopdracht

Elk zoekresultaat van een Google Scholar zoekopdracht geeft de journal op dezelfde manier weer, zoals afgebeeld in figuur acht. De journals kunnen dus geparsed worden uit de broncode van het zoekresultaat. Wanneer de broncode van de verschillende zoekresultaten van een Google Scholar zoekopdracht opgehaald wordt, kunnen we vaak aanverwante kernwoorden terugvinden. Deze bevinden zich in de header met als attribuutnaam ‘citation_keywords’.
Er kan besloten worden dat de informatie die nodig is om de tool te ontwikkelen opgehaald kan worden. De ontwikkeling van de tool is dus haalbaar.

4.5. Realisatie van het prototype

4.5.1. Verloop van het ontwikkelingsproces

Het ontwikkelingsproces voor de realisatie van het prototype kan worden onderverdeeld in acht fases. De eerste fase behelst het uittekenen en aanmaken

van het scherm. Gedurende de tweede fase dient de Google Scholar zoekopdracht geprogrammeerd te worden. Een derde fase van het ontwikkelingsproces is het uitbreiden van het schermontwerp. Het ophalen en weergeven van de journals is de vierde fase. In een volgende fase worden de aanverwante kernwoorden opgehaald. Daarna worden in de zesde fase de aanverwante kernwoorden getoond in een lijst. De zevende fase bestaat uit de ontwikkeling van een woordenwolk van de kernwoorden. Aangezien de woordenwolk getoond wordt als een afbeelding is het niet mogelijk om de verschillende kernwoorden clickable te maken. Om dit op te lossen kan er een nieuwe library voor de creatie van woordenwolken voorzien worden maar dit valt buiten het onderzoek van deze masterproef. Daarom is besloten om behalve de woordenwolk de lijst met kernwoorden ook op het scherm te laten staan. In de achtste en laatste fase ten slotte wordt de filtering toegevoegd. De volledige code is terug te vinden in bijlage vijf en zes.

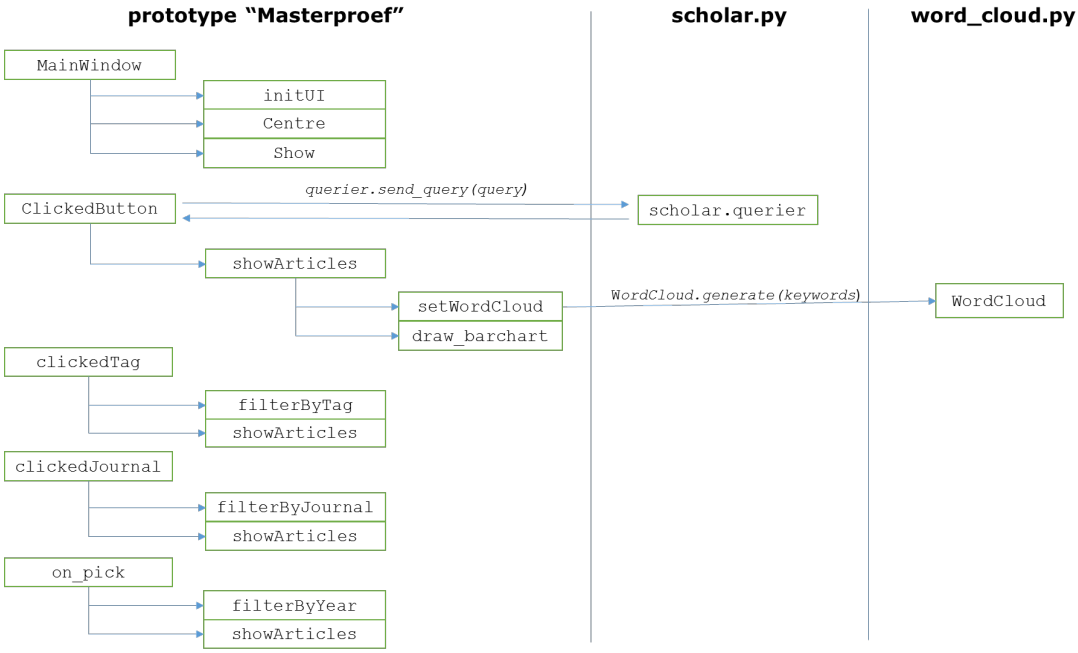
4.5.2. Grafische user interface

Voor het aanmaken van de grafische user interface wordt gebruik gemaakt van een GUI toolkit voor Python genaamd wxPython ("wxPython").

- Wx.TextCtrl wordt gebruikt voor het aanmaken van een textbox voor de invoervelden zoals het onderwerp, de auteur, het beginjaartal en het eindjaartal van de zoekopdracht.
- Wx.BoxSizer wordt toegepast om de elementen op het scherm visueel te ordenen.
- Wx.Expand wordt gebruikt om de grootte van een element op het scherm aan te passen.
- Wx.button dient voor het toevoegen van een knop op het scherm zoals de 'Go'-knop om de zoekopdracht te starten.
- Voor het manipuleren van een lijst op het scherm wordt een aangepaste versie van wx.ListCtrl gebruikt, namelijk AutoWidthListCtrl ("Advanced widgets in wxPython", 2007). Deze widget wordt toegepast voor de lijst van de kernwoorden, de lijst van de journals en het overzicht van gevonden artikels.
- Wx.Gridsizer wordt gebruikt om het scherm onder de zoekbalk op te delen in vier afzonderlijke blokken ("Type GridSizer", 2009). Het eerste blok bevat een tabel voor aanverwante kernwoorden, het tweede blok een tabel van journals. In het derde deel wordt de woordenwolk geplaatst en in de laatste blok wordt een staafdiagram toegevoegd.

4.5.3. Technische componenten

Figuur negen geeft de structuur van de applicatie weer. Voor de realisatie van het prototype wordt gebruik gemaakt van het scholar.py script. Dit is een Python module die de output van Google Scholar kan opvragen en kan parsen. Het script wordt voor deze toepassing uitgebreid met enkele specificaties.



Figuur 9 Structuur applicatie

MainWindow

De instantie van de MainWindow klasse wordt aangemaakt bij de start van het programma. Deze klasse erft over van de wx.Frame klasse. Bij de initialisatie maakt het window de user interface componenten aan, centreert het scherm en toont het scherm van de applicatie. De methodes initUI, Centre en Show worden achtereenvolgens uitgevoerd. In de initUI worden al de widgets aangemaakt en op het scherm geplaatst.

ClickedButton

Wanneer er op de knop 'Go' geklikt wordt, zal de methode ClickedButton uitgevoerd worden. Eerst wordt de query, die door het scholar.py script uitgevoerd dient te worden, geïnitialiseerd en samengesteld met de zoekcriteria die de gebruiker heeft ingevuld in de velden van de zoekopdracht. Indien er noch een topic noch een auteur ingevoerd werden, wordt de functie gestopt. In het andere geval wordt de query verzonden door het uitvoeren van de functie send_query van de ScholarQuerier klasse van scholar.py. De resultaten die

ScholarQuerier teruggeeft, worden vervolgens getoond met behulp van de methode showArticles.

clickedTag

De methode clickedTag wordt uitgevoerd wanneer de gebruiker klikt op een kernwoord in het overzicht van aanverwante kernwoorden. De methode filterByTag wordt vervolgens uitgevoerd om te filteren op het geselecteerde kernwoord. Daarna wordt de methode showArticles opnieuw uitgevoerd om de gefilterde resultaten op het scherm te tonen.

clickedJournal

De methode clickedJournal wordt uitgevoerd wanneer de gebruiker klikt op een journal in het overzicht van journals. De methode filterByJournal wordt vervolgens uitgevoerd om te filteren op het geselecteerde journal. Daarna wordt de methode showArticles opnieuw uitgevoerd om de gefilterde resultaten op het scherm te tonen.

on_pick

De methode on_pick wordt uitgevoerd wanneer de gebruiker klikt op een jaartal in het staafdiagram met de onderzoeksevolutie van een onderwerp. De methode filterByYear wordt vervolgens uitgevoerd om te filteren op een bepaald jaartal. Daarna wordt de methode showArticles opnieuw uitgevoerd om de gefilterde resultaten op het scherm te tonen.

showArticles

De showArticles methode verwerkt de informatie die de ScholarQuerier teruggeeft. Voor de verschillende artikels worden eerst alle gegevens behalve de titel standaard gedefinieerd als ‘Unknown’ of ‘Unavailable’. Vervolgens worden het jaar, de URL en de auteurs opgevuld indien deze gegevens beschikbaar zijn. De verschillende lijsten worden eerst leeggemaakt om de data van een vorige zoekopdracht te verwijderen en vervolgens worden deze lijsten opgevuld met behulp van self.list.Append.

De lijst met aanverwante kernwoorden wordt opgevuld en alfabetisch gesorteerd. Als er aanverwante kernwoorden aanwezig zijn dienen deze ook in een woordenwolk te worden opgenomen. De woordenwolk wordt aangemaakt via de methode setWordCloud.

De lijst met journals wordt opgevuld. De naam van de journal wordt eerst omgezet in kleine letters en speciale tekens zodat de namen vergeleken kunnen

worden. Met behulp van de methode getJournalOccurences wordt het aantal voorkomens van een journal opgehaald. De lijst met journals wordt gesorteerd op basis van dit aantal voorkomens.
Het staafdiagram van de onderzoeksevolutie wordt getoond. Hiervoor wordt gebruik gemaakt van de methode draw_barchart.

setWordCloud

Om een woordenwolk te kunnen tonen wordt gebruik gemaakt van de library wordcloud ("word_cloud", 2015). Deze wordt eerst geïmporteerd. In de methode setWordCloud worden de eigenschappen van de woordenwolk gedefinieerd en wordt de woordenwolk op het scherm getoond.

draw_barchart

Aan de draw_barchart methode wordt een lijst van jaartallen meegegeven. Eerst wordt het bereik van de data gezocht. Vervolgens worden de assen van het staafdiagram leeggemaakt en opnieuw getekend. De assen worden gedefinieerd en het staafdiagram wordt gecreëerd.

Uitbreidingen aan het scholar.py script

De journals, auteurs, keywords en jaartallen van de artikels worden niet aangereikt door het scholar.py script. Daarom wordt dit script aangepast om deze functionaliteit te kunnen aanbieden.
Binnen de klasse ScholarArticle van het scholar.py script worden attributen toegevoegd voor journal, auteurs en jaartal.
De twee versies van de ScholarArticleParser (ScholarArticleParser120201 en ScholarArticleParser120726) worden uitgebreid om de auteurs en de combinatie van journal en jaartal te parsen uit de html en vervolgens deze combinatie nog te splitsen in de journal en het jaartal.
Binnen de ScholarQuerier klasse wordt een methode get_tags toegevoegd. Deze methode zoekt naar verwante kernwoorden in de html die via de BeautifulSoup library wordt opgehaald. De hulpmethode parse_tags haalt de eigenlijke kernwoorden uit de broncode door te zoeken naar 'citation_keywords'.

4.6. Beschrijving van het prototype

4.6.1. Aanpassingen t.o.v. het ontwerp

De interface van het gerealiseerde prototype wijkt slechts in geringe mate af van het ontwerp zoals beschreven in paragraaf 4.2.

De mogelijkheid om het aantal zoekresultaten te preciseren is nog bijkomend voorzien in de zoekbalk. Het maximum aantal zoekresultaten is wel beperkt tot twintig omdat Google Scholar ook maximum twintig resultaten per pagina opvult.
Omdat het niet mogelijk was om te klikken in de woordenwolk, is ervoor gekozen om dezelfde aanverwante kernwoorden daarnaast ook nog te tonen in een lijst. In deze lijst kan de gebruiker wel een kernwoord selecteren waardoor er gefilterd wordt op het geselecteerde woord.

4.6.2. Mogelijkheden

Zoekbalk

De tool beschikt over een zoekbalk waarin men het onderwerp, de auteur en het begin- en eindjaartal van de zoekopdracht kan ingeven. Ook kan men in deze balk het aantal zoekresultaten preciseren. Men kan vrij kiezen hoeveel velden men voor deze zoekopdracht invult. Het is wel verplicht om minstens één topic of één auteur mee te geven. De mogelijkheden die men in een uitgebreide zoekopdracht van Google Scholar terugvindt zijn in deze tool ook van toepassing. Zo kan men voor het topic bijvoorbeeld 'allintitle:' toevoegen om ervoor te zorgen dat het topic in de titel teruggevonden wordt of '-' wanneer een woord niet mag teruggevonden worden in het zoekresultaat.

Resulaten

De tool levert een alfabetisch geordende lijst van aanverwante kernwoorden, een lijst van journals gesorteerd naar gelang het aantal voorkomens, een woordenwolk van aanverwante kernwoorden en een grafiek met het aantal artikels in de tijd om de onderzoeksevolutie van een onderwerp voor te stellen.
Wanneer men klikt op een bepaald kernwoord of een bepaald journal worden enkel de resultaten getoond met betrekking tot de artikels die hierop van toepassing zijn. Men kan ook filteren op een bepaald jaartal door in de grafiek te dubbelklikken.
Men kan steeds verder filteren door op andere kernwoorden of journals te klikken. Filters kunnen terug verwijderd worden met behulp van de knop 'reset filters'. Het overzicht van de aanverwante onderwerpen helpt de onderzoekers om verder te zoeken naar relevante artikels. De lijst met de belangrijkste journals geeft hen een idee van het belang van de gevonden artikels.

4.6.3. Beperkingen

Binnen de termijn van deze thesis kan er enkel een prototype opgeleverd worden. Dit betekent dat men rekening moet houden met een aantal beperkingen. Zo werd er voor de woordenwolk en de tijdlijn gebruik gemaakt van beschikbare Python libraries. Dit beknot de visuele mogelijkheden en de gebruiksvriendelijkheid van de tool. Zo kan men bij de gegenereerde woordenwolk de grootte van de kernwoorden niet zelf bepalen aan de hand van het aantal voorkomens. Op de kleur en de plaats van de woordenwolk heeft men geen invloed. De woorden zijn bovendien niet clickable. De indeling van de tijdlijn wordt automatisch bepaald, wat nogal vreemd kan overkomen omdat er soms met halve eenheden gewerkt wordt. Het meer in detail indelen van de tijdlijn na selectie van een jaartal is hierdoor ook niet mogelijk. Het zou immers veel zinvoller zijn indien men dan het aantal voorkomens per maand zou kunnen tonen.

Daarnaast zijn er nog een aantal voor de hand liggende verbeteringen mogelijk zoals het ophalen en tonen van meer dan twintig resultaten, het vermelden van het totaal aantal beschikbare documenten die aan de zoekcriteria voldoen en het ophalen van artikels uit meerdere databases. Het zou handig zijn als men in de lijst met artikels kan klikken op de url van een artikel en eventueel ook op het pdf-formaat van het artikel om het desbetreffende artikel onmiddellijk te openen.

5. Evaluatiefase

Voor de evaluatie van het prototype werd aan twee professoren en twee doctoraatsstudenten van de faculteit BEW een demo gegeven en feedback gevraagd aan de hand van een vragenlijst. Verder werd aan één doctoraatsstudent het prototype ter beschikking gesteld met behulp van Google Drive. Deze kreeg een beknopte omschrijving van de werking van de tool en dezelfde vragenlijst. De omschrijving van de werking van de tool en de vragenlijst kunnen teruggevonden worden in bijlage twee en drie. De mails die hiervoor verstuurd werden zijn omsloten in bijlage vier.

Positieve bevindingen

Eén professor omschreef vooral de functionaliteit van de aanverwante kernwoorden als nuttig omdat deze in een volgende zoekopdracht gebruikt konden worden. De andere professor vond deze functionaliteit ook bruikbaar vooral wanneer men minder vertrouwd is met een onderwerp. Volgens deze professor heeft de functionaliteit van de filtering op aanverwante kernwoorden in combinatie met de onderzoeksevolutie potentieel om kernwoorden te vinden die gedurende de tijd uit gebruik zijn geraakt of door een ander kernwoord zijn vervangen. Twee van de doctoraatsstudenten vonden de functionaliteit van aanverwante kernwoorden ook nuttig. Eén van de twee nuanceerde dat dit enkel nuttig was indien deze kernwoorden niet reeds met behulp van standaardtaal in artikels aangegeven worden. In zijn domein zijn de verschillende kernwoorden reeds geordend en moeten deze met behulp van een standaardtaal opgenomen worden in de broncode van het artikel. Een andere doctoraatsstudent gaf aan dat de grafiek met de onderzoeksevolutie conceptueel wel nuttig is.

Mogelijke verbeteringen

De voornaamste beperking die aangehaald werd door alle doctoraatsstudenten en professoren is het geringe aantal zoekresultaten dat wordt getoond.

Allen misten een verduidelijking bij het gebruik van de zoekbalk. Men had nood aan een aanduiding van de verschillende mogelijke commando's voor de zoekopdracht of ondersteuning via een bijkomend scherm voor geavanceerd zoeken zoals in Google Scholar voor handen is. Een doctoraatsstudent miste voornamelijk meer filtercriteria zoals die ook bij de UHasselt bibliotheek en Pubmed teruggevonden kunnen worden. Een voorbeeld van een filter die door deze doctoraatsstudent gegeven werd, is het type van het onderzoek. Een automatische aanvulling of het geven van suggesties van onderwerpen van zodra

getypt wordt in de zoekbalk, zou volgens deze doctoraatsstudent een bijkomende verbetering zijn. Eén van de professoren vond dat er een indicatie moet zijn wanneer de tool bezig is met het verwerken van de zoekopdracht.
Verschillende opmerkingen werden gegeven in verband met de getoonde zoekresultaten. Allen vonden het een nadeel dat er niet op de link bij de zoekresultaten geklikt kon worden. Er zou dan moeten worden doorverwezen naar een webpagina waar toegang verkregen kan worden tot deze referentie. Eén van de professoren had in plaats van de link naar het artikel liever een verwijzing naar de Google Scholar pagina van het artikel gezien zodat meteen ook de citaten geraadpleegd kunnen worden. Een gegeven dat nog ontbreekt in het overzicht van gevonden resultaten is volgens één van de doctoraatsstudenten een aanduiding van de impact factor of SSCI index. Een andere doctoraatsstudent zag een mogelijke uitbreiding in het aangeven of de volledige tekst beschikbaar is via Google Scholar. Eén van de professoren had graag een aanduiding gezien van het type van de link is. Hiermee wordt bedoeld of het gaat om een artikel, een boek of nog een ander type. Op één professor na vonden alle geïnterviewde personen het een beperking dat er enkel gebruik gemaakt wordt van Google Scholar. Eén van de professoren had graag over de mogelijkheid beschikt om de tabel met zoekresultaten te kunnen sorteren.
Een mogelijke verbetering die aangehaald werd door één van de professoren is het weergeven van de belangrijkste auteurs.
Deze professor vond dat er nog nood is aan extra informatie om het gebruik van de tool te verduidelijken. Hij had graag een kleine algemene handleiding in de interface van de tool.
Eén van de doctoraatsstudenten vermeldde dat er soms vreemde resultaten verschijnen tussen de kernwoorden en dat een getoond journal niet overeenkwam met de realiteit. Dit wijst erop dat de manier waarop de gegevens opgehaald en geparsed worden nog kan worden verfijnd.

Visualisatie

De eenvoud van de interface van de tool werd zeer positief bevonden door één van de doctoraatsstudenten. Eén van de professoren vond de interface van het prototype eenvoudig maar in orde. Een algemene opmerking was dat de verschillende titeltjes groter mochten zijn. De woordenwolk vond men een zeer goede manier van visualiseren indien deze clickable zou zijn en indien de grootte van de kernwoorden gebaseerd is op het aantal voorkomens van het kernwoord. Eén van de doctoraatsstudenten en één van de professoren gaf ook aan dat de woordenwolk eerder moet opgevat worden als een begrippenwolk dan als een

wolk met losse woorden. Nu staan bijvoorbeeld ‘event’ en ‘log’ afzonderlijk vermeld, terwijl ‘event log’ verwacht werd als kernwoord.
De zoekresultaten worden momenteel voorgesteld met behulp van één lijn in een tabel met daarin onder andere de titel, auteur en link van het artikel. Doordat dit veel informatie is voor één lijn merkte één van de professoren op dat het mogelijk beter is om deze informatie over twee lijnen te tonen. Deze professor zou ook graag een trendlijn op het staafdiagram zien. Allen zagen nog mogelijkheid tot verbetering van het staafdiagram. De assen worden immers automatisch en daardoor onduidelijk ingedeeld. Eén van de professoren merkte ook op dat de jaartallen op de assen voor de staven in het staafdiagram worden geplaatst in plaats van eronder wat het interpreteren van het staafdiagram minder eenvoudig maakt. Eén van de doctoraatsstudenten dacht aan een mogelijke toevoeging in de vorm van een soort netwerk rond een bepaald topic waarin auteurs met elkaar verbonden zijn.
Eén van de professoren merkte op dat de volgorde van de verschillende widgets afhangt van hoe de tool gebruikt wordt, maar dat hij de zoekresultaten liever naast de aanverwante kernwoorden had gezien. Deze professor zou de zoekresultaten immers gebruiken als een validatie voor de gevonden aanverwante kernwoorden.
Hij had verder graag kunnen filteren op meerdere aanverwante kernwoorden tegelijk. De mogelijkheid om te filteren op een artikel uit de lijst van gevonden zoekresultaten zou voor hem ook een toegevoegde waarde hebben. Momenteel worden de aanverwante kernwoorden gezocht in de HTML van de link. Een aanvulling die door een doctoraatsstudent werd aangehaald zou zijn om ook te zoeken in het abstract en de titel. Het ontwikkelen van een algoritme om te zoeken in het abstract en de titel werd ook door één van de professoren als een mogelijkheid vermeld. Om de snelheid van het prototype te verbeteren gaf deze professor als suggestie om de kernwoorden van de resultaten parallel op te halen en niet één voor één. De meeste van de geïnterviewde personen gaven verder nog aan dat het handig zou zijn om een indicatie te hebben van de filtering die op dat moment van toepassing is.

Bruikbaarheid en verwachtingen

Het prototype voldoet volledig aan de verwachtingen van één van de doctoraatsstudenten. De tool zou onmiddellijk door deze doctoraatsstudente gebruikt worden indien er nog enkele aanpassingen gemaakt worden.
Een andere doctoraatsstudent vond dat de tool potentieel heeft om het uitvoeren van een literatuurstudie te ondersteunen, mits enkele uitbreidingen om de tool in

de praktijk werkbaar te maken. Volgens hem zal een onderzoeker een nieuwe tool pas gebruiken indien deze een duidelijk aantoonbare meerwaarde heeft, waardoor het de moeite loont inspanningen te leveren vertrouwd te raken met de tool. Hij merkte verder op eerder te geloven in de integratie en verbetering van bestaande tools. De grafiek die de onderzoeksevolutie weergeeft zou bijvoorbeeld een waardevolle toevoeging kunnen zijn aan Google Scholar zelf.

Eén professor zag zeker potentieel in het prototype en zou de tool gebruiken wanneer de voornaamste beperkingen opgelost worden.

Een andere professor zou de tool niet in eerste plaats gebruiken maar zou er in bepaalde omstandigheden, zoals het gebrek aan toegang tot de universiteitsbibliotheek, wel gebruik van maken.

Een andere doctoraatsstudent zou de tool totaal niet gebruiken. Deze doctoraatsstudent was meer op zoek naar een tool waarmee alles gedaan kan worden met betrekking tot het zoeken naar literatuur en daarvoor is deze tool te beperkt. Hij oordeelde dan ook dat de enige voor hem relevante meerwaarde ten opzichte van de UHasselt bibliotheek de lijst van journals is.

6. Bespreking

Uit de evaluaties van de onderzoekers kan men afleiden dat het ontwikkelde prototype de vooropgestelde doelstelling niet volledig kan waarmaken. Het ter beschikking stellen van de aanverwante onderwerpen, de lijst van journals en de evolutie in de tijd werden door de meeste onderzoekers wel als een meerwaarde ervaren. Het toepassen van visualisatietechnieken zoals een tijdlijn en een woordenwolk werd ook positief onthaald. Wel blijkt dat de uitwerking en de manier waarop de gegevens gevisualiseerd worden, uiterst belangrijke factoren zijn. Zo stelde één professor bijvoorbeeld voor om een trendlijn te voorzien op het staafdiagram. De doctoraatsstudenten stelden ook tekortkomingen vast aan het diagram, zoals de onlogische indeling van de assen. De woordenwolk werd niet optimaal bevonden. Indien visuele voorstellingen gebreken vertonen, worden ze als minder bruikbaar ervaren en gaan ze aan het doel voorbij. De bedoeling van dergelijke visualisatietechnieken is om de gebruiker inzichten te verschaffen en patronen duidelijk te maken in de aangereikte informatie. De onderzoekers hebben geen uitspraak gedaan of de opgeleverde tool hen hierin kan helpen of niet.

Een vaststelling is wel dat de onderzoekers de tool vooral beoordeelden op gebruiksvriendelijkheid en op de aangeboden functionaliteiten. Uit de evaluaties blijkt dat de onderzoekers vooral een tool wensen die een ruime waaier aan functionaliteiten ter beschikking stelt. Zo wil men kunnen beschikken over uitgebreide filtercriteria en zoekmogelijkheden met een automatische aanvulling of het geven van suggesties van onderwerpen van zodra er getypt wordt. Men wenst bovendien veel meer informatie terug te krijgen van een zoekdracht. Concreet werden de impact factor of SSCI index, de belangrijkste auteurs, een aanduiding of het volledige artikel beschikbaar is, de prijs om toegang te krijgen tot een artikel, de universiteit waaraan het artikel is gelinkt en het type van het onderzoek vernoemd in de evaluatie of in de voorafgaande interviews. Het feit dat de tool enkel gebruik maakt van Google Scholar en vooral het geringe aantal zoekresultaten werden als beperkingen vermeld.

Pas als al deze basisvereisten om informatie te zoeken zijn ingevuld, kan men mogelijk meerwaarde leveren door het toevoegen van technieken van datamining en visuele analyse uit het domein van de business intelligence. Het idee om een netwerk te visualiseren rond een bepaald topic waarin auteurs met elkaar verbonden zijn is hier een voorbeeld van, net als het voorstel om een overzicht te geven van de belangrijkste universiteiten bij een topic. Een bedenking hierbij is

wel dat deze technieken enkel zinvol zijn om relaties en structuren weer te geven in zeer grote volumes aan informatie. Het is dus pas nuttig als men over de mogelijkheid beschikt om alle gegevens vooraf op te halen, wat bij Google Scholar bijvoorbeeld niet het geval is. Dit was een ernstige beperking voor het opgeleverde prototype.

Over het algemeen zagen de onderzoekers wel potentieel in het prototype om het uitvoeren van een literatuurstudie te ondersteunen mits de nodige uitbreidingen voorzien worden. Momenteel werden de mogelijkheden van het prototype als te beperkt ervaren om echt bruikbaar te zijn in de praktijk.

7. Conclusie

De doelstelling van dit onderzoek was om na te gaan of men academici kan ondersteunen bij het maken van een literatuurstudie door gebruik te maken van business intelligence. Volgens de design science methode werd een prototype ontwikkeld om na te gaan in welke mate een BI-tool hierin kan tegemoetkomen. De belangrijkste bevindingen worden hierna opgesomd. Tot slot wordt er nog kritische reflectie gemaakt bij het onderzoek.

7.1. Bevindingen

Het gerealiseerde prototype geeft geen eenduidig antwoord op de vraag of het gebruik van technieken uit het domein van de business intelligence een grote meerwaarde is voor academici bij het uitvoeren van een literatuurstudie. Het prototype wordt door de academici aanzien als een interessante aanvulling naast de bestaande tools en zoekmachines die ze momenteel gebruiken.

De methodes en technologieën die binnen het domein van de business intelligence worden toegepast zijn complex en vernieuwend en kunnen moeilijk binnen de beperkte scope van een prototype uitgewerkt worden. Het toepassen van business analytics vergt een architectuur met mid-tier servers waarop engines voor datamining en tekstmining draaien die complexe algoritmes kunnen uitvoeren. Uit de literatuurstudie blijkt dat de laatste evolutie van de visuele analyse gaat in de richting van een combinatie van automatische analyse en interactieve visualisaties. De visualisatietechnieken die gebruikt worden zijn geavanceerd en vereisen specifieke kennis. Ze hebben als voornaamste doelstelling om inzicht te bezorgen in geval van zeer grote hoeveelheden aan informatie, die vaak zelfs meerdere dimensies bevat.

De gerealiseerde BI-tool is louter een front-end applicatie, die wel gebruikt maakt van enkele visualisatietechnieken uit de visuele analyse. Deze technieken zijn op zich niet vernieuwend. Het innoverende aspect zit in de context waarin deze technieken worden toegepast, namelijk om visuele ondersteuning te geven bij het evalueren van de zoekresultaten in het kader van een zoekopdracht voor een literatuurstudie. De reacties van de academici ten opzichte van de visualisaties waren positief, zij het iets minder over de concrete uitwerking. Men kan hieruit wel afleiden dat het gebruik van dergelijke visualisatietechnieken waarschijnlijk kan bijdragen voor het analyseren van de zoekresultaten, vooral wanneer deze gebruikt wordt in combinatie met tekstmining. Maar onderzoek op grotere schaal met deelname van een groter aantal van academici verdeeld over diverse

onderzoeksgebieden en universiteiten is nodig om de precieze behoeften duidelijker in kaart te brengen.
Het aantal gevonden zoekresultaten van de opgeleverde BI-tool waren te beperkt om een zinvol visuele analyse op toe te passen. Hiervoor moet een tool een grote hoeveelheid van zoekresultaten kunnen ophalen afkomstig van verschillende zoekmachines en databases.
Opdat academici een BI-tool zullen gebruiken, zal deze ook moeten beschikken over allerlei mogelijkheden die nu reeds terug te vinden zijn in de verschillende zoekprogramma's voor wetenschappelijke literatuur, zoals uitgebreide zoek- en filtercriteria, anders is de opzet te beperkt.
Een alternatief kan zijn om een afzonderlijk BI-tool te ontwikkelen, louter voor de visuele analyse van de gevonden resultaten, maar dan zou men over de mogelijkheid moeten beschikken om de gevonden resultaten van meerdere zoekopdrachten te exporteren vanuit reeds bestaande zoekmachines en deze vervolgens op te laden in een afzonderlijk datawarehouse.
Een andere denkrichting is dat bestaande zoekomgevingen uitgebreid worden met technieken uit de visuele analyse. ProQuest heeft met het tonen van een tijdlijn van de artikels reeds een stap gezet in deze richting. De universiteitsbibliotheek of Google Scholar zijn voorbeelden van andere toepassingen die visuele analyse technieken zouden kunnen aanwenden om de onderzoekers te ondersteunen.

7.2. Reflectie bij het onderzoek

De bedoeling van het geleverde prototype was om meer inzicht te verschaffen in de mogelijkheden die business intelligence kan bieden om academici te ondersteunen bij het uitvoeren van een literatuuronderzoek. Om er voor te zorgen dat het aangereikte prototype beter aansluit bij de wensen van de academici is het belangrijk dat deze doelstelling duidelijker geschetst en afgelijnd wordt bij aanvang van het onderzoek. Uit de evaluaties van het opgeleverde prototype bleek immers dat de verwachtingen van de onderzoekers hier niet mee overeenstemden. De academici hadden blijkbaar een tool verwacht die de vergelijking met bestaande zoekmachines voor wetenschappelijke literatuur kan doorstaan.
De interviews dienen niet enkel duidelijk te maken welke informatie de onderzoekers kan helpen om relevante documenten terug te vinden, maar ook welke patronen of verbanden tussen de gegevens men had willen ontdekken.
Om patronen te kunnen ontdekken is het echter noodzakelijk dat men een groot aantal zoekresultaten kan evalueren, wat nu niet het geval was. Door ervoor te

zorgen dat het prototype meer zoekresultaten kan ophalen, kan men meer betekenisvolle resultaten tonen, waardoor men bepaalde evoluties of verbanden kan ontdekken, die nu niet zichtbaar zijn. De filtering op aanverwante kernwoorden in combinatie met de onderzoeksevolutie heeft bijvoorbeeld potentieel om kernwoorden te vinden die gedurende de tijd uit gebruik zijn geraakt of door een ander kernwoord zijn vervangen, maar om dit duidelijk te maken is het noodzakelijk dat er meer zoekresultaten in beschouwing worden genomen. Door de Scholar.py library uit te breiden, zodanig dat deze telkens de volgende reeks van beschikbare artikels gaat ophalen, zouden meer zoekresultaten getoond kunnen worden.

Het is zinvol om in een eerste fase meer aandacht te besteden aan het onderzoeken van de mogelijkheden en de technische beperkingen van een te ontwikkelen prototype zodanig dat men deze mee in overweging kan nemen tijdens de interviews met de onderzoekers. Op die manier creëert men geen verwachtingen waaraan de tool later niet kan voldoen. De mogelijke visualisatietechnieken kan men bijvoorbeeld vooraf reeds toelichten en aan bod laten komen.

Om door middel van een prototype aan te tonen dat de toepassing van business intelligence in een literatuurstudie zinvol is, dient men rekening te houden met het feit dat de visuele analyse evolueert in de richting van een combinatie van automatische analyse en interactieve visualisaties. Beide facetten van de visuele analyse zijn echter geavanceerd en vereisen een specifieke technische kennis.

Referentielijst

Advanced widgets in wxPython. (2009). Beschikbaar op 26 juli, 2015, via http://zetcode.com/wxpython/advanced/).

Agilent Literature Search (z.d.). Opgevraagd op 4 mei 2014, via http://apps.cytoscape.org/apps/agilentliteratureresearch.

Agilent Literature Search Software (z.d.). Opgevraagd op 4 mei, 2014, via http://www.agilent.com/labs/research/litsearch.html.

Albano, A. (2013). Decision Support Databases Essentials.

Becker, K. G., Hosack, D. A., Dennis, G., Lempicki, R. A., Bright, T. J., Cheadle, C., & Engel, J. (2003). PubMatrix: a tool for multiplex literature mining. *BMC bioinformatics*, *4*(1), 61.

Boote, D. N., & Beile, P. (2005). Scholars before researchers: On the centrality of the dissertation literature review in research preparation. *Educational Researcher, 34*(6), 3-15.

Chaudhuri, S., Dayal, U., Narasayya, V. An Overview of Busines Intelligence Technology, *Communications of the ACM*, August 2011.

Claus, B. (2001). Business intelligence: Het bouwen van een (enterprise) data warehouse: Case study: ALZ genk. Diepenbeek: LUC.

Clifton, B. Advanced Web Metrics with Google Analytics. Wiley, 2010 (2nd edition) or 2012 (3rd edition). Chps. 1,2.

Ding, Y. (2011). Scientific collaboration and endorsement: Network analysis of coauthorship and citation networks. *Journal of informetrics*, *5*(1), 187-203.

FRIS Onderzoeksportaal. (2008). Beschikbaar op 26 juli, 2015, via http://www.researchportal.be/index.html

Fürnkranz, J. (2002). Web structure mining exploiting the graph structure of the world-wide web.

Gall, M. D., Borg, W. R., & Gall, J. P. (1996). *Educational research: An introduction* . Longman Publishing.

Getting started with wxPython. (2013). Beschikbaar op 26 juli, 2015, via http://wiki.wxpython.org/Getting%20Started.

Glossary (z.d.). Opgevraagd op 4 mei, 2014, via http://wps.prenhall.com/bp_turban_dsbis_9/141/36108/9243675.cw/content/index.html.

Gregor, S., & Hevner, A. R. (2013). POSITIONING AND PRESENTING DESIGN SCIENCE RESEARCH FOR MAXIMUM IMPACT. *MIS Quarterly*, *37*(2).

Han, J., Kamber, M., & Pei, J. (2006). *Data mining: concepts and techniques*. Morgan kaufmann.

Hart, C. (1998). *Doing a literature review: Releasing the social science research imagination.* London: Sage.

Hearst, M. (2003). What is text mining. *Retrieved February*, *7*, 2011.

Hevner, A.R., March, S.T., Park, J. and Ram, S. (2004), Design Science in Information Systems Research. *MIS Quarterly, 28(1),* pp. 75-105.

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems*, *19*(2), 4.

Hotho, A., Nürnberger, A., & Paaß, G. (2005, May). A Brief Survey of Text Mining. In *Ldv Forum* (Vol. 20, No. 1, pp. 19-62).

Inmon, W. H. (1996). The data warehouse and data mining. *Communications of the ACM*, *39*(11), 49-50.

Keim, D. A. (2002). Information visualization and visual data mining. *Visualization and Computer Graphics, IEEE Transactions on*, *8*(1), 1-8.

Keim, D. A., Mansmann, F., Schneidewind, J., & Ziegler, H. (2006, July). Challenges in visual data analysis. In *Information Visualization, 2006. IV 2006. Tenth International Conference on* (pp. 9-16). IEEE.

Khoo, C. S., Na, J. C., & Jaidka, K. (2011). Analysis of the macro-level discourse structure of literature reviews. *Online Information Review*, *35*(2), 255-271.

Kwon, B. C., Javed, W., Ghani, S., Elmqvist, N., Yi, J. S., & Ebert, D. S. (2012). Evaluating the role of time in investigative analysis of document collections.*Visualization and Computer Graphics, IEEE Transactions on*, *18*(11), 1992-2004.

Meyer, D., Hornik, K., & Feinerer, I. (2008). Text mining infrastructure in R. *Journal of Statistical Software*, *25*(5), 1-54.

Lim, E. P., Chen, H., & Chen, G. (2013). Business intelligence and analytics: Research directions. *ACM Transactions on Management Information Systems (TMIS)*, *3*(4), 17.

Iivari, J. (2007). A paradigmatic analysis of information systems as a design science. *Scandinavian Journal of Information Systems*, *19*(2), 5.

Lönnqvist, A., & Pirttimäki, V. (2006). The measurement of business intelligence. *Information Systems Management*, *23*(1), 32.

Lu, Z. (2011). PubMed and beyond: a survey of web tools for searching biomedical literature. *Database: the journal of biological databases and curation*, *2011*(Preprint).

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision support systems*, *15*(4), 251-266.

March, S. T., & Storey, V. C. (2008). Design science in the information systems discipline: an introduction to the special issue on design science research. *Management Information Systems Quarterly*, *32*(4), 6.

Mobasher, B. (2007). Data mining for web personalization. In *The adaptive web* (pp. 90-135). Springer Berlin Heidelberg.

Moustakas, C. (1994). *Phenomenological research methods*. Sage Publications.

Negash, S. (2004). Business intelligence. *The Communications of the Association for Information Systems*, *13*(1), 54.

Otte, E., & Rousseau, R. (2002). Social network analysis: a powerful strategy, also for the information sciences. *Journal of information Science*, *28*(6), 441-453.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research.*Journal of management information systems*, *24*(3), 45-77.

ProQuest (z.d.). Opgevraagd op 4 mei, 2014, via http://www.proquest.com.

Python functions. 2015. Beschikbaar op 26 juli, 2015, via http://www.tutorialspoint.com/python/python_functions.htm.

Randolph, J. J. (2009). A guide to writing the dissertation literature review. *Practical Assessment, Research & Evaluation*, *14*(13), 2.

Sabater, J., & Sierra, C. (2002, July). Reputation and social network analysis in multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1* (pp. 475-482). ACM.

Scholar.py. (2015). Beschikbaar op 26 juli, 2015, via https://github.com/ckreibich/scholar.py/blob/master/README.md

Scime, A. (Ed.). (2005). *Web mining: applications and techniques*. IGI Global.

Sekaran, U., & Bougie, R. (2009). Research methods for business: A skill-building approach. NYC: John Willey Sons.

Simon, H. A. (1996). *The sciences of the artificial* (Vol. 136). MIT press.

Srivastava, A. N., & Sahami, M. (Eds.). (2009). *Text mining: Classification, clustering, and applications*. CRC Press.

Stasko, J., Görg, C., & Liu, Z. (2008). Jigsaw: supporting investigative analysis through interactive visualization. *Information visualization*, *7*(2), 118-132.

Sun, G. D., Wu, Y. C., Liang, R. H., & Liu, S. X. (2013). A survey of visual analytics techniques and applications: State-of-the-art research and future challenges. *Journal of Computer Science and Technology*, *28*(5), 852-867.

Tan, A. H. (1999, April). Text mining: The state of the art and the challenges. In *Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases* (pp. 65-70).

Trujillo, J., & Luján-Mora, S. (2003). A UML based approach for modeling ETL processes in data warehouses. In *Conceptual Modeling-ER 2003* (pp. 307-320). Springer Berlin Heidelberg.

TurbanE, A., & LiangT, S. (2007). Decision support and business intelligence systems, 8th EditionPrenticeHall.

Type GridSizer. (2009). Beschikbaar op 26 juli, 2015, via http://www.wxpython.org/docs/api/wx.GridSizer-class.html.

Type Sizer. (2009). Beschikbaar op 26 juli, 2015, via http://www.wxpython.org/docs/api/wx.Sizer-class.html#AddSpacer.

Von Alan, R. H., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS quarterly*, *28*(1), 75-105.

Von Isenburg, M. (2007). HubMed. *Journal of the Medical Library Association*, *95*(1), 95.

Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *Management Information Systems Quarterly*,*26*(2), 3.

Word_cloud. (2015). Beschikbaar op 26 juli, 2015, via https://github.com/amueller/word_cloud.

wxPython. (z.d.). Beschikbaar op 26 juli, 2015, via http://www.wxpython.org/.

Yang, Z., Hong, L., & Davison, B. D. (2010, April). Topic-driven multi-type citation network analysis. In *Adaptivity, Personalization and Fusion of Heterogeneous Information* (pp. 24-31). LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

Lijst van figuren

Bijlage 1 Mail interview 1

Geachte,

Mijn naam is Sofie Theunissen. Ik zit momenteel in mijn tweede master Handelsingenieur in de Beleidsinformatica. Mijn masterproef heeft als doel een tool te ontwikkelen die academische onderzoekers helpt gedurende hun literatuuronderzoek. De bedoeling van de BI tool die ik wil ontwikkelen is om te helpen bij een literatuurstudie. Door gebruik te maken van bijvoorbeeld Google Scholar zou mogelijk een tool ontwikkeld kunnen worden die aan de hand van een gegeven topic (of een bestaand artikel) zinvolle informatie aanlevert.

Om mijn onderzoeksonderwerp af te bakenen zou ik een goed inzicht willen krijgen in hoe jullie te werk gaan bij het literatuuronderzoek. Ik zou dan ook willen vragen of jullie tijd kunnen maken in jullie drukke agenda voor een kort interview.

Alvast enorm bedankt voor de medewerking,

Met vriendelijke groeten,

Sofie Theunissen

Bijlage 2 Beknopte beschrijving tool

Opstarten van de tool

De tool werd gedeeld via Google Drive. U kan deze hier downloaden. Wanneer u de tool wil opstarten krijgt u eerst een foutmelding. Wanneer u op 'OK' klikt, wordt de tool gewoon opgestart.

Mogelijkheden van de tool

Zoekbalk

De tool beschikt over een zoekbalk waarin men het onderwerp, de auteur en het begin- en eindjaartal van de zoekopdracht kan ingeven. Ook kan men in deze balk het aantal zoekresultaten preciseren. Dit aantal resultaten is in het prototype nog beperkt tot een maximum van 20 vermits Google Scholar, waarmee de tool achterliggend werkt, maar maximum 20 resultaten per pagina weergeeft.
Men kan vrij kiezen hoeveel velden men voor deze zoekopdracht invult. Het is wel verplicht om minstens één topic of één auteur mee te geven.
De mogelijkheden die men in een uitgebreide zoekopdracht van Google Scholar terugvindt zijn in deze tool ook van toepassing. Zo kan men voor het topic bijvoorbeeld 'allintitle:' toevoegen om ervoor te zorgen dat het topic in de titel teruggevonden wordt of '-' wanneer een woord niet mag teruggevonden worden in het zoekresultaat.
Omdat het zoeken naar aanverwante kernwoorden enige tijd in beslag neemt, duurt het soms lang vooraleer de resultaten getoond worden. De tool geeft geen indicatie dat de applicatie bezig is. In de tool is nog geen aandacht besteed aan de performantie.

Resulaten

De tool levert hierna een alfabetisch geordende lijst van aanverwante kernwoorden, een lijst van journals gesorteerd naar gelang het aantal voorkomens, een woordenwolk van aanverwante kernwoorden en een grafiek met het aantal artikels in de tijd om de onderzoeksevolutie van een onderwerp voor te stellen. Voor de woordenwolk en grafiek werd gebruik gemaakt van reeds bestaande Python libraries. Wanneer eigen libraries ontwikkeld zouden worden, zou het mogelijk zijn om de grootte van de kernwoorden te bepalen aan de hand

van het aantal voorkomens en zouden de assen van het staafdiagram duidelijker kunnen zijn.
Wanneer men klikt op een bepaald kernwoord of een journal worden enkel de resultaten getoond met betrekking tot de artikels die hierop van toepassing zijn. Men kan ook filteren op een bepaald jaartal door in de grafiek te dubbelklikken. Men kan steeds verder filteren door op andere kernwoorden of journals te klikken. Filters kunnen terug verwijderd worden met behulp van de knop ‘reset filters’.

Bijlage 3 Evaluatieformulier

Welke functionaliteiten van de tool vindt u nuttig en waarom? (bijvoorbeeld de lijst van kernwoorden, de grafiek van de onderzoeksevolutie, …)

Heeft u opmerkingen op de manier waarop de resultaten worden voorgesteld? Wat vindt u goed, wat kan beter? (doorklikken, woordenwolk, lijst, staafdiagram, ….)

Wat zou u anders willen zien aan de tool? Hoe kan de tool verbeterd worden?

Welke functionaliteiten ontbreken er volgens u nog aan de tool?

In welke mate voldoet deze tool aan uw verwachtingen?

Helpt deze tool u in het algemeen bij het uitvoeren van een literatuurstudie?

Zou u gebruik maken van deze tool bij het zoeken naar literatuur?
Neen, of nee tenzij of ja op voorwaarde dat …

Heeft u nog andere opmerkingen?

Bijlage 4 Mail demo

Geachte,

Gedurende de loop van het eerste semester heb ik van u een kort interview afgenomen in het kader van mijn masterproef. De bedoeling van het interview was om te peilen op welke wijze een BI tool kan helpen bij een literatuurstudie. Ondertussen heb ik een prototype van een tool ontwikkeld. U zou me een grote dienst verlenen indien ik bij u langs zou kunnen komen om een demo van de tool te geven en om hierop feedback te krijgen.

Bedankt voor de medewerking,

Met vriendelijke groeten,

Sofie Theunissen

Geachte,

Mijn masterproef heeft als doel een tool te ontwikkelen die academische onderzoekers helpt gedurende hun literatuuronderzoek. Ik heb ondertussen een prototype van een tool ontwikkeld. U zou me een grote dienst verlenen indien ik bij u langs zou kunnen komen om een demo van de tool te geven en om hierop feedback te krijgen.

Bedankt voor de medewerking,

Met vriendelijke groeten,

Sofie Theunissen

Bijlage 5 Code

```
import locale
import re

from matplotlib.patches import Rectangle

__author__ = 'Sofie Theunissen'

import wx
import scholar
import HTMLParser

from wx.lib.mixins.listctrl import ListCtrlAutoWidthMixin, ColumnSorterMixin

import matplotlib
matplotlib.use('WXAgg')

from matplotlib.figure import Figure
from matplotlib.backends.backend_wxagg import \
    FigureCanvasWxAgg as FigCanvas

from wordcloud import WordCloud

class AutoWidthListCtrl(wx.ListCtrl, ListCtrlAutoWidthMixin):
    def __init__(self, parent):
        wx.ListCtrl.__init__(self, parent, -1, style=wx.LC_REPORT)
        ListCtrlAutoWidthMixin.__init__(self)
        #ColumnSorterMixin.__init__(self, 1)
        self.setResizeColumn(0)

    def GetListCtrl(self):
        return self


class MainWindow(wx.Frame):

    def __init__(self, parent, title):
        super(MainWindow, self).__init__(parent, title=title,
            size=(1400, 700))

        self.InitUI()
        self.Centre()
        self.Show()

    def InitUI(self):

        panel = wx.Panel(self)
```

```
mainSizer = wx.BoxSizer(wx.VERTICAL)
inputSizer = wx.BoxSizer(wx.HORIZONTAL)

lblTopic = wx.StaticText(panel, label="Topic: ")
self.editTopic = wx.TextCtrl(panel)
inputSizer.Add(lblTopic)
inputSizer.Add(self.editTopic, wx.EXPAND)


lblAuthor = wx.StaticText(panel, label="Author: ")
self.editAuthor = wx.TextCtrl(panel)
inputSizer.AddSpacer(16)
inputSizer.Add(lblAuthor)
inputSizer.Add(self.editAuthor, wx.EXPAND)


lblStartTime = wx.StaticText(panel, label="From: ")
self.editStartTime = wx.TextCtrl(panel)
inputSizer.AddSpacer(16)
inputSizer.Add(lblStartTime)
inputSizer.Add(self.editStartTime, wx.EXPAND)

lblEndTime = wx.StaticText(panel, label="Until: ")
self.editEndTime = wx.TextCtrl(panel)
inputSizer.AddSpacer(16)
inputSizer.Add(lblEndTime)
inputSizer.Add(self.editEndTime, wx.EXPAND)

lblResults = wx.StaticText(panel, label="Results: ")
self.editResults = wx.TextCtrl(panel)
self.editResults.SetValue('10')
inputSizer.AddSpacer(16)
inputSizer.Add(lblResults)
inputSizer.Add(self.editResults, wx.EXPAND)


buttonGo = wx.Button(panel, label="Go!")
self.Bind(wx.EVT_BUTTON, self.ClickedButton, buttonGo)
inputSizer.AddSpacer(16)
inputSizer.Add(buttonGo)

mainSizer.Add(inputSizer, flag=wx.TOP | wx.LEFT | wx.RIGHT, border=8)

gridSizer = wx.GridSizer(rows=1, cols=4, hgap=8, vgap=8)

self.keywordList = AutoWidthListCtrl(panel)
self.keywordList.SetMinSize((300,250))
self.keywordList.InsertColumn(0, 'Keywords')
self.Bind(wx.EVT_LIST_ITEM_ACTIVATED, self.clickedTag,
```

```
self.keywordList)
        gridSizer.Add(self.keywordList, wx.EXPAND)

        self.journalList = AutoWidthListCtrl(panel)
        self.journalList.SetMinSize((300,250))
        self.journalList.InsertColumn(0, 'Journals')
        self.Bind(wx.EVT_LIST_ITEM_ACTIVATED, self.clickedJournal,
self.journalList)
        gridSizer.Add(self.journalList, wx.EXPAND)

        self.wordcloud = wx.StaticBitmap(panel)
        self.wordcloud.SetMinSize((300, 250))
        gridSizer.Add(self.wordcloud, wx.EXPAND)

        self.dpi = 50
        self.fig = Figure((7.0, 5.0), dpi=self.dpi)
        self.canvas = FigCanvas(panel, -1, self.fig)
        self.axes = self.fig.add_subplot(111)
        self.canvas.mpl_connect('pick_event', self.on_pick)
        self.axes.clear()
        gridSizer.Add(self.canvas)

        mainSizer.Add(gridSizer, 1,  wx.ALL | wx.EXPAND, border=8)

        buttonReset = wx.Button(panel, label="Reset filters!")
        self.Bind(wx.EVT_BUTTON, self.ClickedResetFilters, buttonReset)
        mainSizer.Add(buttonReset, 0, wx.ALIGN_CENTER | wx.ALIGN_BOTTOM,
border=8)


        listSizer = wx.BoxSizer(wx.HORIZONTAL)
        self.list = AutoWidthListCtrl(panel)
        self.list.InsertColumn(0, 'Title')
        self.list.InsertColumn(1, 'Authors')
        self.list.InsertColumn(2, 'Year')
        self.list.InsertColumn(3, 'Journal')
        self.list.InsertColumn(4, 'Link')

        listSizer.Add(self.list, 1, flag=wx.EXPAND | wx.ALL, border=8)
        mainSizer.Add(listSizer, 1, wx.EXPAND)

        panel.SetSizerAndFit(mainSizer)

    def clickedTag(self, event):
        filteredArticles = self.filterByTag(self.articles, event.GetText())
        self.showArticles(None, filteredArticles, False)

    def clickedJournal(self, event):
        filteredArticles = self.filterByJournal(self.articles, event.GetText())
        self.showArticles(None, filteredArticles, False)
```

```
    def on_pick(self, event):
        if isinstance(event.artist, Rectangle):
            patch = event.artist
            year = patch.get_x()
            filteredArticles = self.filterByYear(self.articles, year)
            self.showArticles(None, filteredArticles, False)

    def isNumber(self, value):
        try:
            value += 1
            return True
        except TypeError:
            return False

    def draw_barchart(self, data):
        self.data = data

        minBin = min(data) -1
        maxBin = max(data) +1
        binData = range(minBin, maxBin + 1)
        freqData = []
        for x in range(minBin, maxBin + 1):
            freqData.append(0)
        for x in data:
            freqData[x - minBin] += 1

        self.axes.clear()

        self.axes.grid(False)
        self.axes.autoscale(tight=True)

        self.axes.bar(binData, freqData, color='green', alpha=0.44, picker=True)

        self.axes.set_title("Articles per year")
        self.axes.set_xlabel("Year")
        self.axes.set_ylabel("Amount")
        self.axes.set_picker(True)

        self.canvas.draw()

    def setWordCloud(self, text):
        wc = WordCloud(background_color="white", width=300,
height=250).generate(text)
        wc.to_file('wordcloud.png')
        png = wx.Image('wordcloud.png', wx.BITMAP_TYPE_ANY)

        self.PhotoMaxSize = 300
        img = wx.Image('wordcloud.png', wx.BITMAP_TYPE_ANY)
        W = img.GetWidth()
```

```
        H = img.GetHeight()
        if W > H:
            NewW = self.PhotoMaxSize
            NewH = self.PhotoMaxSize * H / W
        else:
            NewH = self.PhotoMaxSize
            NewW = self.PhotoMaxSize * W / H
        img = img.Scale(NewW,NewH)
        self.wordcloud.SetBitmap(wx.BitmapFromImage(img))

    def toNumber(self, value):
        try:
            return int(value)
        except ValueError:
            return None

    def filterByTag(self, articles, filterTag):
        filteredArticles = []
        htmlFormatter = HTMLParser.HTMLParser()

        for article in articles:
            if article['tags'] != None:
                for tag in article['tags']:
                    if tag.strip().lower() == filterTag:
                        filteredArticles.append(article)

        return filteredArticles

    def filterByJournal(self, articles, journalName):
        filteredArticles = []

        for article in articles:
            if article['journal'] != None:
                if article['journal'] == journalName:
                    filteredArticles.append(article)

        return filteredArticles

    def filterByYear(self, articles, filterYear):
        filteredArticles = []
        htmlFormatter = HTMLParser.HTMLParser()

        for article in articles:
            if article['year'] != None:
                year = int(htmlFormatter.unescape(article['year']))
                if year == filterYear:
                    filteredArticles.append(article)

        return filteredArticles
```

```
def getJournalOccurences(self, journalName):
    journalCount = 0
    articles = self.articles

    for article in articles:
        if article['journal'] != None:
            if article['journal'] == journalName:
                journalCount += 1

    return journalCount


def sortKeywords (self, int1, int2):

    val1 = self.keywordList.GetItemText(int1)
    val2 = self.keywordList.GetItemText(int2)
    print val1, val2

    return val1 < val2


def showArticles(self, querier, articles, getTags=False):
    self.list.DeleteAllItems()
    self.journalList.DeleteAllItems()
    self.keywordList.DeleteAllItems()

    htmlFormatter = HTMLParser.HTMLParser()
    tagString = ""
    years = []
    hasUnknownJournal = False
    journalCount = 0
    tagCount = 0
    tagList = []
    journals = []
    journals.append("unknown")

    for article in articles:
        title = htmlFormatter.unescape(article['title'])

        authors = "Unknown"
        link = "Unavaible"
        year = "Unknown"
        pdf = "Unavailable"
        journal = "unknown"

        if article['journal'] != None:
            journal = htmlFormatter.unescape(article['journal'])
            if journal is not "unknown":
                journal = journal.lower()
                journal = re.sub('[^0-9a-zA-Z ]+', '', journal)
```

```
                article['journal'] = journal
                journals.append(journal)
            else:
                article['journal'] = "unknown"

            if article['year'] != None:
                year =  htmlFormatter.unescape(article['year'])
                years.append(year)
            if article['url'] != None:
                link = article['url']
                if getTags:
                    tags = querier.get_article_tags(link)
                    article['tags'] = tags
                else:
                    tags = article['tags']
                for tag in tags:
                    tag = tag.strip().lower()
                    if len(tag) > 0:
                        tagList.append(tag)

            if article['authors'] != None:
                authors = htmlFormatter.unescape(article['authors'])

            self.list.Append([title.strip(), authors.strip(), year.strip(), journal.strip(),
link.strip()])

        sortedJournals = reversed(sorted(set(journals), key=self.getJournalOccurences))
        tagList = sorted(set(tagList))

        for tag in tagList:
            self.keywordList.Append([tag])
            self.keywordList.SetItemData(tagCount, long(tagCount))
            tagCount += 1
            tagString += tag + "\n"

        for journal in sortedJournals:
            self.journalList.Append([journal])
            self.journalList.SetItemData (journalCount, long(journalCount) )
            journalCount += 1

        if len(tagString) > 0:
            self.setWordCloud(tagString)
        years = map(int, years)
        self.draw_barchart(years)

    def ClickedResetFilters(self, event):
        if self.articles is not None:
            self.showArticles(None, self.articles, False)

    def ClickedButton(self, event):
```

```
        querier = scholar.ScholarQuerier()
        settings = scholar.ScholarSettings()
        querier.apply_settings(settings)
        query = scholar.SearchScholarQuery()

        topic = self.editTopic.GetValue()
        if len(topic.strip()) > 1:
            query.set_words(topic)

        author = self.editAuthor.GetValue()
        if len(author.strip()) > 1:
            query.set_author(author)

        start = self.toNumber(self.editStartTime.GetValue().strip())
        end = self.toNumber(self.editEndTime.GetValue().strip())

        if start != None and self.isNumber(start) and end != None and
self.isNumber(end):
            query.set_timeframe(start,end)
        elif start != None and self.isNumber(start) and end == None:
            query.set_timeframe(start,None)
        elif start == None  and end == None and self.isNumber(end):
            query.set_timeframe(None,end)

        if len(topic.strip()) == 0 and len(author.strip()) == 0:
            return

        num_result_field = self.editResults.GetValue().strip()
        num_results = 10
        if len(num_result_field) > 0:
            num_results = self.toNumber(num_result_field)

        query.set_num_page_results(num_results)

        querier.send_query(query)

        articles = querier.articles

        self.articles = articles
        self.showArticles(querier, articles, True)


if __name__ == '__main__':
    app = wx.App()
    MainWindow(None, title='MasterProef')
    app.MainLoop()
```

Bijlage 6 Aangepast scholar.py script

```
import optparse
import os
import sys
import re

try:
    # Try importing for Python 3
    # pylint: disable-msg=F0401
    # pylint: disable-msg=E0611
    from urllib.request import HTTPCookieProcessor, Request, build_opener
    from urllib.parse import quote, unquote
    from http.cookiejar import MozillaCookieJar
except ImportError:
    # Fallback for Python 2
    from urllib2 import Request, build_opener, HTTPCookieProcessor
    from urllib import quote, unquote
    from cookielib import MozillaCookieJar

# Import BeautifulSoup -- try 4 first, fall back to older
try:
    from bs4 import BeautifulSoup
except ImportError:
    try:
        from BeautifulSoup import BeautifulSoup
    except ImportError:
        print('We need BeautifulSoup, sorry...')
        sys.exit(1)

# Support unicode in both Python 2 and 3. In Python 3, unicode is str.
if sys.version_info[0] == 3:
    unicode = str # pylint: disable-msg=W0622
    encode = lambda s: s # pylint: disable-msg=C0103
else:
    def encode(s):
        if isinstance(s, basestring):
            return s.encode('utf-8') # pylint: disable-msg=C0103
        else:
            return str(s)


class Error(Exception):
    """Base class for any Scholar error."""


class FormatError(Error):
    """A query argument or setting was formatted incorrectly."""
```

```
class QueryArgumentError(Error):
    """A query did not have a suitable set of arguments."""


class ScholarConf(object):
    """Helper class for global settings."""

    VERSION = '2.9'
    LOG_LEVEL = 1
    MAX_PAGE_RESULTS = 20 # Current maximum for per-page results
    SCHOLAR_SITE = 'http://scholar.google.com'

    # USER_AGENT = 'Mozilla/5.0 (X11; U; FreeBSD i386; en-US; rv:1.9.2.9)
Gecko/20100913 Firefox/3.6.9'
    # Let's update at this point (3/14):
    USER_AGENT = 'Mozilla/5.0 (X11; Linux x86_64; rv:27.0) Gecko/20100101
Firefox/27.0'

    # If set, we will use this file to read/save cookies to enable
    # cookie use across sessions.
    COOKIE_JAR_FILE = None

class ScholarUtils(object):
    """A wrapper for various utensils that come in handy."""

    LOG_LEVELS = {'error': 1,
                  'warn':  2,
                  'info':  3,
                  'debug': 4}

    @staticmethod
    def ensure_int(arg, msg=None):
        try:
            return int(arg)
        except ValueError:
            raise FormatError(msg)

    @staticmethod
    def log(level, msg):
        if level not in ScholarUtils.LOG_LEVELS.keys():
            return
        if ScholarUtils.LOG_LEVELS[level] > ScholarConf.LOG_LEVEL:
            return
        sys.stderr.write('[%5s]  %s' % (level.upper(), msg + '\n'))
        sys.stderr.flush()


class ScholarArticle(object):
```

```
    """
    A class representing articles listed on Google Scholar.  The class
    provides basic dictionary-like behavior.
    """
    def __init__(self):
        # The triplets for each keyword correspond to (1) the actual
        # value, (2) a user-suitable label for the item, and (3) an
        # ordering index:
        self.attrs = {
            'title':         [None, 'Title',          0],
            'url':           [None, 'URL',            1],
            'year':          [None, 'Year',           2],
            'authors':       [None, 'Authors',        3],
            'journal':       [None, 'Journals',       4],
            'num_citations': [0,    'Citations',      5],
            'num_versions':  [0,    'Versions',       6],
            'cluster_id':    [None, 'Cluster ID',     7],
            'url_pdf':       [None, 'PDF link',       8],
            'url_citations': [None, 'Citations list', 9],
            'url_versions':  [None, 'Versions list', 10],
            'url_citation':  [None, 'Citation link', 11],
            'excerpt':       [None, 'Excerpt',       12],
            }

        # The citation data in one of the standard export formats,
        # e.g. BibTeX.
        self.citation_data = None

    def __getitem__(self, key):
        if key in self.attrs:
            return self.attrs[key][0]
        return None

    def __len__(self):
        return len(self.attrs)

    def __setitem__(self, key, item):
        if key in self.attrs:
            self.attrs[key][0] = item
        else:
            self.attrs[key] = [item, key, len(self.attrs)]

    def __delitem__(self, key):
        if key in self.attrs:
            del self.attrs[key]

    def set_citation_data(self, citation_data):
        self.citation_data = citation_data

    def as_txt(self):
```

```
        # Get items sorted in specified order:
        items = sorted(list(self.attrs.values()), key=lambda item: item[2])
        # Find largest label length:
        max_label_len = max([len(str(item[1])) for item in items])
        fmt = '%%%%ds %%s' % max_label_len
        res = []
        for item in items:
            if item[0] is not None:
                res.append(fmt % (item[1], item[0]))
        return '\n'.join(res)

    def as_csv(self, header=False, sep='|'):
        # Get keys sorted in specified order:
        keys = [pair[0] for pair in \
                sorted([(key, val[2]) for key, val in list(self.attrs.items())],
                       key=lambda pair: pair[1])]
        res = []
        if header:
            res.append(sep.join(keys))
        res.append(sep.join([unicode(self.attrs[key][0]) for key in keys]))
        return '\n'.join(res)

    def as_citation(self):
        """
        Reports the article in a standard citation format. This works only
        if you have configured the querier to retrieve a particular
        citation export format. (See ScholarSettings.)
        """
        return self.citation_data or ''


class ScholarArticleParser(object):
    """
    ScholarArticleParser can parse HTML document strings obtained from
    Google Scholar. This is a base class; concrete implementations
    adapting to tweaks made by Google over time follow below.
    """
    def __init__(self, site=None):
        self.soup = None
        self.article = None
        self.site = site or ScholarConf.SCHOLAR_SITE
        self.year_re = re.compile(r'\b(?:20|19)\d{2}\b')
        self.authors_re = re.compile(r'[^-|*')
        self.journalyear_re = re.compile(r'-[^-]*, (?:20|19)\d{2}')

    def handle_article(self, art):
        """
        The parser invokes this callback on each article parsed
        successfully.  In this base class, the callback does nothing.
        """
```

```
def handle_num_results(self, num_results):
    """
    The parser invokes this callback if it determines the overall
    number of results, as reported on the parsed results page. The
    base class implementation does nothing.
    """

def parse(self, html):
    """
    This method initiates parsing of HTML content, cleans resulting
    content as needed, and notifies the parser instance of
    resulting instances via the handle_article callback.
    """
    self.soup = BeautifulSoup(html)

    # This parses any global, non-itemized attributes from the page.
    self._parse_globals()

    # Now parse out listed articles:
    for div in self.soup.findAll(ScholarArticleParser._tag_results_checker):
        self._parse_article(div)
        self._clean_article()
        if self.article['title']:
            self.handle_article(self.article)

def _clean_article(self):
    """
    This gets invoked after we have parsed an article, to do any
    needed cleanup/polishing before we hand off the resulting
    article.
    """
    if self.article['title']:
        self.article['title'] = self.article['title'].strip()

def _parse_globals(self):
    tag = self.soup.find(name='div', attrs={'id': 'gs_ab_md'})
    if tag is not None:
        raw_text = tag.findAll(text=True)
        # raw text is a list because the body contains <b> etc
        if raw_text is not None and len(raw_text) > 0:
            try:
                num_results = raw_text[0].split()[1]
                # num_results may now contain commas to separate
                # thousands, strip:
                num_results = num_results.replace(',', '')
                num_results = int(num_results)
                self.handle_num_results(num_results)
            except (IndexError, ValueError):
                pass
```

```
def _parse_article(self, div):
    self.article = ScholarArticle()

    for tag in div:
        if not hasattr(tag, 'name'):
            continue

        if tag.name == 'div' and self._tag_has_class(tag, 'gs_rt') and \
                tag.h3 and tag.h3.a:
            self.article['title'] = ''.join(tag.h3.a.findAll(text=True))
            self.article['url'] = self._path2url(tag.h3.a['href'])
            if self.article['url'].endswith('.pdf'):
                self.article['url_pdf'] = self.article['url']

        if tag.name == 'font':
            for tag2 in tag:
                if not hasattr(tag2, 'name'):
                    continue
                if tag2.name == 'span' and \
                   self._tag_has_class(tag2, 'gs_fl'):
                    self._parse_links(tag2)

def _parse_links(self, span):
    for tag in span:
        if not hasattr(tag, 'name'):
            continue
        if tag.name != 'a' or tag.get('href') is None:
            continue

        if tag.get('href').startswith('/scholar?cites'):
            if hasattr(tag, 'string') and tag.string.startswith('Cited by'):
                self.article['num_citations'] = \
                    self._as_int(tag.string.split()[-1])

            # Weird Google Scholar behavior here: if the original
            # search query came with a number-of-results limit,
            # then this limit gets propagated to the URLs embedded
            # in the results page as well. Same applies to
            # versions URL in next if-block.
            self.article['url_citations'] = \
                self._strip_url_arg('num', self._path2url(tag.get('href')))

            # We can also extract the cluster ID from the versions
            # URL. Note that we know that the string contains "?",
            # from the above if-statement.
            args = self.article['url_citations'].split('?', 1)[1]
            for arg in args.split('&'):
                if arg.startswith('cites='):
                    self.article['cluster_id'] = arg[6:]
```

```
        if tag.get('href').startswith('/scholar?cluster'):
            if hasattr(tag, 'string') and tag.string.startswith('All '):
                self.article['num_versions'] = \
                    self._as_int(tag.string.split()[1])
            self.article['url_versions'] = \
                self._strip_url_arg('num', self._path2url(tag.get('href')))

        if tag.getText().startswith('Import'):
            self.article['url_citation'] = self._path2url(tag.get('href'))


    @staticmethod
    def _tag_has_class(tag, klass):
        """
        This predicate function checks whether a BeatifulSoup Tag instance
        has a class attribute.
        """
        res = tag.get('class') or []
        if type(res) != list:
            # BeautifulSoup 3 can return e.g. 'gs_md_wp gs_ttss',
            # so split -- conveniently produces a list in any case
            res = res.split()
        return klass in res

    @staticmethod
    def _tag_results_checker(tag):
        return tag.name == 'div' \
            and ScholarArticleParser._tag_has_class(tag, 'gs_r')

    @staticmethod
    def _as_int(obj):
        try:
            return int(obj)
        except ValueError:
            return None

    def _path2url(self, path):
        """Helper, returns full URL in case path isn't one."""
        if path.startswith('http://'):
            return path
        if not path.startswith('/'):
            path = '/' + path
        return self.site + path

    def _strip_url_arg(self, arg, url):
        """Helper, removes a URL-encoded argument, if present."""
        parts = url.split('?', 1)
        if len(parts) != 2:
            return url
```

```
        res = []
        for part in parts[1].split('&'):
            if not part.startswith(arg + '='):
                res.append(part)
        return parts[0] + '?' + '&'.join(res)


class ScholarArticleParser120201(ScholarArticleParser):
    """
    This class reflects update to the Scholar results page layout that
    Google recently.
    """
    def _parse_article(self, div):
        self.article = ScholarArticle()

        for tag in div:
            if not hasattr(tag, 'name'):
                continue

            if tag.name == 'h3' and self._tag_has_class(tag, 'gs_rt') and tag.a:
                self.article['title'] = ''.join(tag.a.findAll(text=True))
                self.article['url'] = self._path2url(tag.a['href'])
                if self.article['url'].endswith('.pdf'):
                    self.article['url_pdf'] = self.article['url']

            if tag.name == 'div' and self._tag_has_class(tag, 'gs_a'):
                year = self.year_re.findall(tag.text)
                authors = self.authors_re.findall(tag.text)
                line = self.journalyear_re.findall(tag.text)
                print line

                if len(line.rsplit(',', 1)) == 2: # Gotcha; journal present
                    self.article['journal'] = line.rsplit(',', 1)[0].strip()

                #self.article['authors'] = authors[0].strip() if len(authors) > 0 else None
                self.article['authors'] =  authors[0].strip() if len(authors) > 0 else None
                self.article['year'] = year[0] if len(year) > 0 else None

            if tag.name == 'div' and self._tag_has_class(tag, 'gs_fl'):
                self._parse_links(tag)


class ScholarArticleParser120726(ScholarArticleParser):
    """
    This class reflects update to the Scholar results page layout that
    Google made 07/26/12.
    """
    def _parse_article(self, div):
        self.article = ScholarArticle()
```

```
        for tag in div:
            if not hasattr(tag, 'name'):
                continue
            if str(tag).lower().find('.pdf'):
                if tag.find('div', {'class': 'gs_ttss'}):
                    self._parse_links(tag.find('div', {'class': 'gs_ttss'}))

            if tag.name == 'div' and self._tag_has_class(tag, 'gs_ri'):
                # There are (at least) two formats here. In the first
                # one, we have a link, e.g.:
                #
                # <h3 class="gs_rt">
                #   <a href="http://dl.acm.org/citation.cfm?id=972384" class="yC0">
                #     <b>Honeycomb</b>: creating intrusion detection signatures using
                #       honeypots
                #   </a>
                # </h3>
                #
                # In the other, there's no actual link -- it's what
                # Scholar renders as "CITATION" in the HTML:
                #
                # <h3 class="gs_rt">
                #   <span class="gs_ctu">
                #     <span class="gs_ct1">[CITATION]</span>
                #     <span class="gs_ct2">[C]</span>
                #   </span>
                #   <b>Honeycomb</b> automated ids signature creation using honeypots
                # </h3>
                #
                # We now distinguish the two.
                try:
                    atag = tag.h3.a
                    self.article['title'] = ''.join(atag.findAll(text=True))
                    self.article['url'] = self._path2url(atag['href'])
                    if self.article['url'].endswith('.pdf'):
                        self.article['url_pdf'] = self.article['url']
                except:
                    # Remove a few spans that have unneeded content (e.g. [CITATION])
                    for span in tag.h3.findAll(name='span'):
                        span.clear()
                    self.article['title'] = ''.join(tag.h3.findAll(text=True))

                if tag.find('div', {'class': 'gs_a'}):
                    year = self.year_re.findall(tag.find('div', {'class': 'gs_a'}).text)
                    authors = self.authors_re.findall(tag.find('div', {'class': 'gs_a'}).text)
                    journalline = self.journalyear_re.findall(tag.find('div', {'class':
'gs_a'}).text)

                    self.article['authors'] = authors[0] if len(authors) > 0 else None
                    self.article['year'] = year[0] if len(year) > 0 else None
```

```
            line = journalline[0] if len(journalline) > 0 else None


            if line is not None and len(line.rsplit(',', 1)) == 2: # Gotcha; journal
present
                self.article['journal'] = line.rsplit(',', 1)[0].strip()[2:]

        if tag.find('div', {'class': 'gs_fl'}):
            self._parse_links(tag.find('div', {'class': 'gs_fl'}))

        if tag.find('div', {'class': 'gs_rs'}):
            # These are the content excerpts rendered into the results.
            raw_text = tag.find('div', {'class': 'gs_rs'}).findAll(text=True)
            if len(raw_text) > 0:
                raw_text = ''.join(raw_text)
                raw_text = raw_text.replace('\n', '')
                self.article['excerpt'] = raw_text


class ScholarQuery(object):
    """
    The base class for any kind of results query we send to Scholar.
    """
    def __init__(self):
        self.url = None

        # The number of results requested from Scholar -- not the
        # total number of results it reports (the latter gets stored
        # in attrs, see below).
        self.num_results = ScholarConf.MAX_PAGE_RESULTS

        # Queries may have global result attributes, similar to
        # per-article attributes in ScholarArticle. The exact set of
        # attributes may differ by query type, but they all share the
        # basic data structure:
        self.attrs = {}

    def set_num_page_results(self, num_page_results):
        msg = 'maximum number of results on page must be numeric'
        self.num_results = ScholarUtils.ensure_int(num_page_results, msg)

    def get_url(self):
        """
        Returns a complete, submittable URL string for this particular
        query instance. The URL and its arguments will vary depending
        on the query.
        """
        return None

    def _add_attribute_type(self, key, label, default_value=None):
```

```
        """
        Adds a new type of attribute to the list of attributes
        understood by this query. Meant to be used by the constructors
        in derived classes.
        """
        if len(self.attrs) == 0:
            self.attrs[key] = [default_value, label, 0]
            return
        idx = max([item[2] for item in self.attrs.values()]) + 1
        self.attrs[key] = [default_value, label, idx]

    def __getitem__(self, key):
        """Getter for attribute value. Returns None if no such key."""
        if key in self.attrs:
            return self.attrs[key][0]
        return None

    def __setitem__(self, key, item):
        """Setter for attribute value. Does nothing if no such key."""
        if key in self.attrs:
            self.attrs[key][0] = item

    def _parenthesize_phrases(self, query):
        """
        Turns a query string containing comma-separated phrases into a
        space-separated list of tokens, quoted if containing
        whitespace. For example, input

          'some words, foo, bar'

        becomes

          '"some words" foo bar'

        This comes in handy during the composition of certain queries.
        """
        if query.find(',') < 0:
            return query
        phrases = []
        for phrase in query.split(','):
            phrase = phrase.strip()
            if phrase.find(' ') > 0:
                phrase = '"' + phrase + '"'
            phrases.append(phrase)
        return ' '.join(phrases)


class ClusterScholarQuery(ScholarQuery):
    """
    This version just pulls up an article cluster whose ID we already
```

```
    know about.
    """
    SCHOLAR_CLUSTER_URL = ScholarConf.SCHOLAR_SITE + '/scholar?' \
        + 'cluster=%(cluster)s' \
        + '&num=%(num)s'

    def __init__(self, cluster=None):
        ScholarQuery.__init__(self)
        self._add_attribute_type('num_results', 'Results', 0)
        self.cluster = None
        self.set_cluster(cluster)

    def set_cluster(self, cluster):
        """
        Sets search to a Google Scholar results cluster ID.
        """
        msg = 'cluster ID must be numeric'
        self.cluster = ScholarUtils.ensure_int(cluster, msg)

    def get_url(self):
        if self.cluster is None:
            raise QueryArgumentError('cluster query needs cluster ID')

        urlargs = {'cluster': self.cluster,
                   'num': self.num_results or ScholarConf.MAX_PAGE_RESULTS}

        for key, val in urlargs.items():
            urlargs[key] = quote(encode(val))

        return self.SCHOLAR_CLUSTER_URL % urlargs


class SearchScholarQuery(ScholarQuery):
    """
    This version represents the search query parameters the user can
    configure on the Scholar website, in the advanced search options.
    """
    SCHOLAR_QUERY_URL = ScholarConf.SCHOLAR_SITE + '/scholar?' \
        + 'as_q=%(words)s' \
        + '&as_epq=%(phrase)s' \
        + '&as_oq=%(words_some)s' \
        + '&as_eq=%(words_none)s' \
        + '&as_occt=%(scope)s' \
        + '&as_sauthors=%(authors)s' \
        + '&as_publication=%(pub)s' \
        + '&as_ylo=%(ylo)s' \
        + '&as_yhi=%(yhi)s' \
        + '&as_sdt=%(patents)s%%2C5' \
        + '&as_vis=%(citations)s' \
        + '&btnG=&hl=en' \
```

```
    + '&num=%(num)s'

def __init__(self):
    ScholarQuery.__init__(self)
    self._add_attribute_type('num_results', 'Results', 0)
    self.words = None # The default search behavior
    self.words_some = None # At least one of those words
    self.words_none = None # None of these words
    self.phrase = None
    self.scope_title = False # If True, search in title only
    self.author = None
    self.pub = None
    self.timeframe = [None, None]
    self.include_patents = True
    self.include_citations = True

def set_words(self, words):
    """Sets words that *all* must be found in the result."""
    self.words = words

def set_words_some(self, words):
    """Sets words of which *at least one* must be found in result."""
    self.words_some = words

def set_words_none(self, words):
    """Sets words of which *none* must be found in the result."""
    self.words_none = words

def set_phrase(self, phrase):
    """Sets phrase that must be found in the result exactly."""
    self.phrase = phrase

def set_scope(self, title_only):
    """
    Sets Boolean indicating whether to search entire article or title
    only.
    """
    self.scope_title = title_only

def set_author(self, author):
    """Sets names that must be on the result's author list."""
    self.author = author

def set_pub(self, pub):
    """Sets the publication in which the result must be found."""
    self.pub = pub

def set_timeframe(self, start=None, end=None):
    """
    Sets timeframe (in years as integer) in which result must have
```

```
    appeared. It's fine to specify just start or end, or both.
    """
    if start:
        start = ScholarUtils.ensure_int(start)
    if end:
        end = ScholarUtils.ensure_int(end)
    self.timeframe = [start, end]

def set_include_citations(self, yesorno):
    self.include_citations = yesorno

def set_include_patents(self, yesorno):
    self.include_patents = yesorno

def get_url(self):
    if self.words is None and self.words_some is None \
       and self.words_none is None and self.phrase is None \
       and self.author is None and self.pub is None \
       and self.timeframe[0] is None and self.timeframe[1] is None:
        raise QueryArgumentError('search query needs more parameters')

    # If we have some-words or none-words lists, we need to
    # process them so GS understands them. For simple
    # space-separeted word lists, there's nothing to do. For lists
    # of phrases we have to ensure quotations around the phrases,
    # separating them by whitespace.
    words_some = None
    words_none = None

    if self.words_some:
        words_some = self._parenthesize_phrases(self.words_some)
    if self.words_none:
        words_none = self._parenthesize_phrases(self.words_none)

    urlargs = {'words': self.words or '',
               'words_some': words_some or '',
               'words_none': words_none or '',
               'phrase': self.phrase or '',
               'scope': 'title' if self.scope_title else 'any',
               'authors': self.author or '',
               'pub': self.pub or '',
               'ylo': self.timeframe[0] or '',
               'yhi': self.timeframe[1] or '',
               'patents': '0' if self.include_patents else '1',
               'citations': '0' if self.include_citations else '1',
               'num': self.num_results or ScholarConf.MAX_PAGE_RESULTS}

    for key, val in urlargs.items():
        urlargs[key] = quote(encode(val))
```

```
        return self.SCHOLAR_QUERY_URL % urlargs


class ScholarSettings(object):

    """
    This class lets you adjust the Scholar settings for your
    session. It's intended to mirror the features tunable in the
    Scholar Settings pane, but right now it's a bit basic.
    """
    CITFORM_NONE = 0
    CITFORM_REFWORKS = 1
    CITFORM_REFMAN = 2
    CITFORM_ENDNOTE = 3
    CITFORM_BIBTEX = 4

    def __init__(self):
        self.citform = 0 # Citation format, default none
        self.per_page_results = ScholarConf.MAX_PAGE_RESULTS
        self._is_configured = False

    def set_citation_format(self, citform):
        citform = ScholarUtils.ensure_int(citform)
        if citform < 0 or citform > self.CITFORM_BIBTEX:
            raise FormatError('citation format invalid, is "%s"' \
                              % citform)
        self.citform = citform
        self._is_configured = True

    def set_per_page_results(self, per_page_results):
        msg = 'page results must be integer'
        self.per_page_results = ScholarUtils.ensure_int(per_page_results, msg)
        self.per_page_results = min(self.per_page_results,
                                    ScholarConf.MAX_PAGE_RESULTS)
        self._is_configured = True

    def is_configured(self):
        return self._is_configured


class ScholarQuerier(object):

    """
    ScholarQuerier instances can conduct a search on Google Scholar
    with subsequent parsing of the resulting HTML content.  The
    articles found are collected in the articles member, a list of
    ScholarArticle instances.
    """

    # Default URLs for visiting and submitting Settings pane, as of 3/14
```

```
    GET_SETTINGS_URL = ScholarConf.SCHOLAR_SITE + '/scholar_settings?' \
      + 'sciifh=1&hl=en&as_sdt=0,5'

    SET_SETTINGS_URL = ScholarConf.SCHOLAR_SITE + '/scholar_setprefs?' \
      + 'q=' \
      + '&scisig=%(scisig)s' \
      + '&instart=0' \
      + '&as_sdt=1,5' \
      + '&as_sdtp=' \
      + '&num=%(num)s' \
      + '&scis=%(scis)s' \
      + '%(scisf)s' \
      + '&hl=en&lang=all&instq=&inst=569367360547434339&save='

    # Older URLs:
    # ScholarConf.SCHOLAR_SITE +
'/scholar?q=%s&hl=en&btnG=Search&as_sdt=2001&as_sdtp=on

    class Parser(ScholarArticleParser120726):
      def __init__(self, querier):
        ScholarArticleParser120726.__init__(self)
        self.querier = querier

      def handle_num_results(self, num_results):
        if self.querier is not None and self.querier.query is not None:
          self.querier.query['num_results'] = num_results

      def handle_article(self, art):
        self.querier.add_article(art)

    def __init__(self):
      self.articles = []
      self.query = None
      self.cjar = MozillaCookieJar()

      # If we have a cookie file, load it:
      if ScholarConf.COOKIE_JAR_FILE and \
         os.path.exists(ScholarConf.COOKIE_JAR_FILE):
        try:
          self.cjar.load(ScholarConf.COOKIE_JAR_FILE,
                         ignore_discard=True)
          ScholarUtils.log('info', 'loaded cookies file')
        except Exception as msg:
          ScholarUtils.log('warn', 'could not load cookies file: %s' % msg)
          self.cjar = MozillaCookieJar() # Just to be safe

      self.opener = build_opener(HTTPCookieProcessor(self.cjar))
      self.settings = None # Last settings object, if any

    def apply_settings(self, settings):
```

```
    """
    Applies settings as provided by a ScholarSettings instance.
    """
    if settings is None or not settings.is_configured():
        return True

    self.settings = settings

    # This is a bit of work. We need to actually retrieve the
    # contents of the Settings pane HTML in order to extract
    # hidden fields before we can compose the query for updating
    # the settings.
    html = self._get_http_response(url=self.GET_SETTINGS_URL,
                                   log_msg='dump of settings form HTML',
                                   err_msg='requesting settings failed')
    if html is None:
        return False

    # Now parse the required stuff out of the form. We require the
    # "scisig" token to make the upload of our settings acceptable
    # to Google.
    soup = BeautifulSoup(html)

    tag = soup.find(name='form', attrs={'id': 'gs_settings_form'})
    if tag is None:
        ScholarUtils.log('info', 'parsing settings failed: no form')
        return False

    tag = tag.find('input', attrs={'type':'hidden', 'name':'scisig'})
    if tag is None:
        ScholarUtils.log('info', 'parsing settings failed: scisig')
        return False

    urlargs = {'scisig': tag['value'],
               'num': settings.per_page_results,
               'scis': 'no',
               'scisf': ''}

    if settings.citform != 0:
        urlargs['scis'] = 'yes'
        urlargs['scisf'] = '&scisf=%d' % settings.citform

    html = self._get_http_response(url=self.SET_SETTINGS_URL % urlargs,
                                   log_msg='dump of settings result HTML',
                                   err_msg='applying setttings failed')
    if html is None:
        return False

    ScholarUtils.log('info', 'settings applied')
    return True
```

```
    def parse_tags(self, html):
        soup = BeautifulSoup(html)
        all_tags = []
        tags = soup.findAll(attrs={"name": "citation_keywords"})

        for tag in tags:
            keywords =
tag['content'].replace('\r\n','').replace('\t','').replace(';',',').split(',')
            map(unicode.strip, keywords)
            all_tags.extend(keywords)

        return all_tags


    def get_article_tags(self, url):
        if url is None:
            return ''
        html = self._get_http_response(url=url,
                                       log_msg='dump of query response HTML',
                                       err_msg='result retrieval failed')

        if html is None:
            return []

        tags = self.parse_tags(html)

        return tags

    def send_query(self, query):
        """
        This method initiates a search query (a ScholarQuery instance)
        with subsequent parsing of the response.
        """
        self.clear_articles()
        self.query = query

        html = self._get_http_response(url=query.get_url(),
                                       log_msg='dump of query response HTML',
                                       err_msg='results retrieval failed')
        if html is None:
            return

        self.parse(html)

    def get_citation_data(self, article):
        """
        Given an article, retrieves citation link. Note, this requires that
        you adjusted the settings to tell Google Scholar to actually
        provide this information, *prior* to retrieving the article.
```

```
        """
        if article['url_citation'] is None:
            return False
        if article.citation_data is not None:
            return True

        ScholarUtils.log('info', 'retrieving citation export data')
        data = self._get_http_response(url=article['url_citation'],
                                       log_msg='citation data response',
                                       err_msg='requesting citation data failed')
        if data is None:
            return False

        article.set_citation_data(data)
        return True

    def parse(self, html):
        """
        This method allows parsing of provided HTML content.
        """
        parser = self.Parser(self)
        parser.parse(html)

    def add_article(self, art):
        self.get_citation_data(art)
        self.articles.append(art)

    def clear_articles(self):
        """Clears any existing articles stored from previous queries."""
        self.articles = []

    def save_cookies(self):
        """
        This stores the latest cookies we're using to disk, for reuse in a
        later session.
        """
        if ScholarConf.COOKIE_JAR_FILE is None:
            return False
        try:
            self.cjar.save(ScholarConf.COOKIE_JAR_FILE,
                           ignore_discard=True)
            ScholarUtils.log('info', 'saved cookies file')
            return True
        except Exception as msg:
            ScholarUtils.log('warn', 'could not save cookies file: %s' % msg)
            return False

    def _get_http_response(self, url, log_msg=None, err_msg=None):
        """
        Helper method, sends HTTP request and returns response payload.
```

```
        """
        if log_msg is None:
            log_msg = 'HTTP response data follow'
        if err_msg is None:
            err_msg = 'request failed'
        try:
            ScholarUtils.log('info', 'requesting %s' % unquote(url))

            req = Request(url=url, headers={'User-Agent':
ScholarConf.USER_AGENT})
            hdl = self.opener.open(req)
            html = hdl.read()

            ScholarUtils.log('debug', log_msg)
            ScholarUtils.log('debug', '>>>>' + '-'*68)
            ScholarUtils.log('debug', 'url: %s' % hdl.geturl())
            ScholarUtils.log('debug', 'result: %s' % hdl.getcode())
            ScholarUtils.log('debug', 'headers:\n' + str(hdl.info()))
            ScholarUtils.log('debug', 'data:\n' + html.decode('utf-8')) # For Python 3
            ScholarUtils.log('debug', '<<<<' + '-'*68)

            return html
        except Exception as err:
            ScholarUtils.log('info', err_msg + ': %s' % err)
            return None

def txt(querier, with_globals):
    if with_globals:
        # If we have any articles, check their attribute labels to get
        # the maximum length -- makes for nicer alignment.
        max_label_len = 0
        if len(querier.articles) > 0:
            items = sorted(list(querier.articles[0].attrs.values()),
                           key=lambda item: item[2])
            max_label_len = max([len(str(item[1])) for item in items])

        # Get items sorted in specified order:
        items = sorted(list(querier.query.attrs.values()), key=lambda item: item[2])
        # Find largest label length:
        max_label_len = max([len(str(item[1])) for item in items] + [max_label_len])
        fmt = '[G] %%%ds %%s' % max(0, max_label_len-4)
        for item in items:
            if item[0] is not None:
                print(fmt % (item[1], item[0]))
        if len(items) > 0:
            print

    articles = querier.articles
    for art in articles:
        print(encode(art.as_txt()) + '\n')
```

```
def csv(querier, header=False, sep='|'):
    articles = querier.articles
    for art in articles:
        result = art.as_csv(header=header, sep=sep)
        print(encode(result))
        header = False

def citation_export(querier):
    articles = querier.articles
    for art in articles:
        print(art.as_citation() + '\n')


def main():
    usage = """scholar.py [options] <query string>
A command-line interface to Google Scholar.

Examples:

# Retrieve one article written by Einstein on quantum theory:
scholar.py -c 1 --author "albert einstein" --phrase "quantum theory"

# Retrieve a BibTeX entry for that quantum theory paper:
scholar.py -c 1 -C 17749203648027613321 --citation bt

# Retrieve five articles written by Einstein after 1970 where the title
# does not contain the words "quantum" and "theory":
scholar.py -c 5 -a "albert einstein" -t --none "quantum theory" --after 1970"""

    fmt = optparse.IndentedHelpFormatter(max_help_position=50, width=100)
    parser = optparse.OptionParser(usage=usage, formatter=fmt)
    group = optparse.OptionGroup(parser, 'Query arguments',
                         'These options define search query arguments and
parameters.')
    group.add_option('-a', '--author', metavar='AUTHORS', default=None,
                     help='Author name(s)')
    group.add_option('-A', '--all', metavar='WORDS', default=None, dest='allw',
                     help='Results must contain all of these words')
    group.add_option('-s', '--some', metavar='WORDS', default=None,
                     help='Results must contain at least one of these words. Pass
arguments in form -s "foo bar baz" for simple words, and -s "a phrase, another
phrase" for phrases')
    group.add_option('-n', '--none', metavar='WORDS', default=None,
                     help='Results must contain none of these words. See -s|--some re.
formatting')
    group.add_option('-p', '--phrase', metavar='PHRASE', default=None,
                     help='Results must contain exact phrase')
    group.add_option('-t', '--title-only', action='store_true', default=False,
                     help='Search title only')
```

```
    group.add_option('-P', '--pub', metavar='PUBLICATIONS', default=None,
                     help='Results must have appeared in this publication')
    group.add_option('--after', metavar='YEAR', default=None,
                     help='Results must have appeared in or after given year')
    group.add_option('--before', metavar='YEAR', default=None,
                     help='Results must have appeared in or before given year')
    group.add_option('--no-patents', action='store_true', default=False,
                     help='Do not include patents in results')
    group.add_option('--no-citations', action='store_true', default=False,
                     help='Do not include citations in results')
    group.add_option('-C', '--cluster-id', metavar='CLUSTER_ID', default=None,
                     help='Do not search, just use articles in given cluster ID')
    group.add_option('-c', '--count', type='int', default=None,
                     help='Maximum number of results')
    parser.add_option_group(group)

    group = optparse.OptionGroup(parser, 'Output format',
                                 'These options control the appearance of the results.')
    group.add_option('--txt', action='store_true',
                     help='Print article data in text format (default)')
    group.add_option('--txt-globals', action='store_true',
                     help='Like --txt, but first print global results too')
    group.add_option('--csv', action='store_true',
                     help='Print article data in CSV form (separator is "|")')
    group.add_option('--csv-header', action='store_true',
                     help='Like --csv, but print header with column names')
    group.add_option('--citation', metavar='FORMAT', default=None,
                     help='Print article details in standard citation format. Argument
Must be one of "bt" (BibTeX), "en" (EndNote), "rm" (RefMan), or "rw"
(RefWorks).')
    parser.add_option_group(group)

    group = optparse.OptionGroup(parser, 'Miscellaneous')
    group.add_option('--cookie-file', metavar='FILE', default=None,
                     help='File to use for cookie storage. If given, will read any existing
cookies if found at startup, and save resulting cookies in the end.')
    group.add_option('-d', '--debug', action='count', default=0,
                     help='Enable verbose logging to stderr. Repeated options increase
detail of debug output.')
    group.add_option('-v', '--version', action='store_true', default=False,
                     help='Show version information')
    parser.add_option_group(group)

    options, _ = parser.parse_args()

    # Show help if we have neither keyword search nor author name
    if len(sys.argv) == 1:
        parser.print_help()
        return 1
```

```
    if options.debug > 0:
        options.debug = min(options.debug, ScholarUtils.LOG_LEVELS['debug'])
        ScholarConf.LOG_LEVEL = options.debug
        ScholarUtils.log('info', 'using log level %d' % ScholarConf.LOG_LEVEL)

    if options.version:
        print('This is scholar.py %s.' % ScholarConf.VERSION)
        return 0

    if options.cookie_file:
        ScholarConf.COOKIE_JAR_FILE = options.cookie_file

    # Sanity-check the options: if they include a cluster ID query, it
    # makes no sense to have search arguments:
    if options.cluster_id is not None:
        if options.author or options.allw or options.some or options.none \
           or options.phrase or options.title_only or options.pub \
           or options.after or options.before:
            print('Cluster ID queries do not allow additional search arguments.')
            return 1

    querier = ScholarQuerier()
    settings = ScholarSettings()

    if options.citation == 'bt':
        settings.set_citation_format(ScholarSettings.CITFORM_BIBTEX)
    elif options.citation == 'en':
        settings.set_citation_format(ScholarSettings.CITFORM_ENDNOTE)
    elif options.citation == 'rm':
        settings.set_citation_format(ScholarSettings.CITFORM_REFMAN)
    elif options.citation == 'rw':
        settings.set_citation_format(ScholarSettings.CITFORM_REFWORKS)
    elif options.citation is not None:
        print('Invalid citation link format, must be one of "bt", "en", "rm", or
"rw".')
        return 1

    querier.apply_settings(settings)

    if options.cluster_id:
        query = ClusterScholarQuery(cluster=options.cluster_id)
    else:
        query = SearchScholarQuery()
        if options.author:
            query.set_author(options.author)
        if options.allw:
            query.set_words(options.allw)
        if options.some:
            query.set_words_some(options.some)
        if options.none:
```

```
            query.set_words_none(options.none)
        if options.phrase:
            query.set_phrase(options.phrase)
        if options.title_only:
            query.set_scope(True)
        if options.pub:
            query.set_pub(options.pub)
        if options.after or options.before:
            query.set_timeframe(options.after, options.before)
        if options.no_patents:
            query.set_include_patents(False)
        if options.no_citations:
            query.set_include_citations(False)

    if options.count is not None:
        options.count = min(options.count, ScholarConf.MAX_PAGE_RESULTS)
        query.set_num_page_results(options.count)

    querier.send_query(query)

    if options.csv:
        csv(querier)
    elif options.csv_header:
        csv(querier, header=True)
    elif options.citation is not None:
        citation_export(querier)
    else:
        txt(querier, with_globals=options.txt_globals)

    if options.cookie_file:
        querier.save_cookies()

    return 0

if __name__ == "__main__":
    sys.exit(main())
```

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Business Intelligence voor academische onderzoekers

Richting: **master in de toegepaste economische wetenschappen: handelsingenieur in de beleidsinformatica**
Jaar: **2015**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Theunissen, Sofie

Datum: **22/08/2015**