Google Scholar zoekresultaten voor wetenschappelijke projecten: linked data & natural language processing

Dhr. Bart De Paepe

Scriptie voorgedragen tot het bekomen van de graad van
Professionele bachelor in de toegepaste informatica

Promotor: Dhr. Jan Claes
Co-promotor: Dhr. Milan Lamote
Academiejaar: 2024–2025
Eerste examenperiode

Departement IT en Digitale Innovatie .

HO
GENT

Woord vooraf

Aan de basis van deze bachelorproef ligt een JIRA-ticket dat al een tijdje oud is. Dat ticket was aangemaakt door het team van IMIS met de bedoeling om een bestaande procedure te verbeteren, doch zonder daar enige prioriteit aan te koppelen. Bij mijn zoektocht naar een onderwerp kwam ik al snel bij mijn werkgever, het VLIZ, terecht. JIRA werd erbij gehaald en het ticket waarvan sprake stak met kop en schouders boven de andere uit omwille van de toepasbaarheid, de mate van uitdaging en de haalbaarheid.

Wat volgde was een boeiende verdieping in de wereld van de academische literatuur, in het gigantische net van de web scraping, en niet in het minst van het eindeloze universum van de LLMs.

Ik wil in het bijzonder mijn manager Bart Vanhoorne bedanken die het mogelijk maakt om een omgeving te creëren waar deze vorm van werken kan bestaan.

Bijzondere dank ook voor mijn collega Milan Lamote voor die onuitputtelijke positiviteit en omdat die zonder aarzelen het co-promotorschap van deze bachelorproef aanvaardde.

Bijzondere dank ook voor mijn promotor Jan Claes voor het opzetten van een transparant kader voor deze bachelorproef.

Geen bijzondere maar niet minder oprechte dank voor mijn collega Fons Verheyde voor zijn enthousiasme bij het bespreken van deze bachelorproef.

Tenslotte aan het einde van deze bachelorproef, maar vooral aan het einde van deze opleiding, een eindeloze dank aan mijn echtgenote Lies Knockaert. Veel van de punten op mijn curiculum zijn onrechtstreeks ook haar verdienste.

Bart De Paepe, Sint-Baafs-Vijve, 1 mei 2025

Samenvatting

Zoekresultaten afkomstig van Google Scholar moeten aan het Integrated Marine Information System (IMIS) toegevoegd worden. Dit onderzoek bekijkt hoe dit proces volledig of grotendeels geautomatiseerd kan worden. Daarbij komen verschillende aspecten kijken die allemaal stuk voor stuk afzonderlijk behandeld worden.
Zo worden Google Scholar alerts gebruikt om continu nieuwe zoekresultaten te ontvangen. Er wordt uitvoerig toegelicht hoe een zoekopdracht aangemaakt moet worden en hoe daarvoor een melding ingesteld kan worden.
Google Scholar zoekresultaten komen in de vorm van een HTML pagina. HTML aan IMIS toevoegen is niet interessant. Er wordt dieper ingegaan hoe de lijst met zoekresultaten van de HTML pagina gescraped kan worden. Daarbij worden verschillende technieken uitgeprobeerd waaronder Large Language Models (LLMs). Uiteindelijk wordt een klassieke benadering gekozen die de HTML parset aan de hand van Beautiful Soup.
Na het scrapen zou de gevonden informatie in IMIS opgeslaan kunnen worden. Maar er zijn nog bijkomende stappen nodig om ervoor te zorgen dat enkel kwalitatieve gegevens naar IMIS vloeien. Behalve het feit dat Google Scholar de zoekresultaten selecteert voor de zoekopdracht en rangschikt volgens een bepaalde volgorde, zijn er verder geen criteria die aangeven hoe relevant een zoekresultaat is voor IMIS. Natural Language Processing biedt daar oplossingen voor. Er wordt een relevantiescore berekend van de mate waarin de zoekopdracht aanwezig is in het zoekresultaat. Algemeen wordt daarvoor aangenomen dat de frequentie van de zoekopdracht in het zoekresultaat rechtevenredig is met de relevantie van het resultaat voor IMIS.
Er mogen ook geen duplicaten in IMIS opgeslaan worden. Duplicaten kunnen gedetecteerd worden aan de hand van de Digital Object Identifier (DOI) van de publicatie. Die kan stapsgewijs opgezocht worden in de link van de publicatie, in Crossref, of op de webpagina van de publicatie. Maar er is geen garantie dat de DOI gevonden zal worden. Voor die gevallen moet er minstens een score gegeven worden of een publicatie al dan niet een duplicaat is. Dat kan gedaan worden aan de hand van “Semantic Search”. Daarvoor worden embeddings berekend voor alle titels in IMIS. Vervolgens wordt de embedding van de titel van het zoekresultaat daarmee vergeleken. De mate van gelijkenis is een score voor duplicaten van de publicatie.
Uiteindelijk worden al deze stappen geïntegreerd in een pijplijn voor het semi-automatisch toevoegen van publicaties aan IMIS. In het geval de relevantiescore voldoende is en

indien de DOI gevonden wordt, dan kan de publicatie zonder tussenkomst aan IMIS toegevoegd worden. Maar wanneer er geen DOI gevonden wordt, dan is er nog steeds een manuele stap nodig om te beslissen op basis van de duplicatenscore of de publicatie toegevoegd mag worden.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Lijst van codefragmenten

1

Inleiding

Het Vlaams Instituut voor de Zee (VLIZ) (“Vlaams Instituut voor de Zee”, 2024) is een pionier in zeekennis. Dit wetenschappelijk instituut gelegen in Oostende heeft onder andere een mandaat om een complete en geactualiseerde catalogus bij te houden van alle wetenschappelijke publicaties in de mariene sector. Al meer dan 20 jaar bouwt het Integrated Marine Information System (IMIS) aan deze catalogus die intussen beschikt over meerdere collecties van marien wetenschappelijk referentiemateriaal.
Naast wetenschappelijke literatuur zitten er ook collecties van mariene wetenschappelijke projecten (vb. het World Register of Marine Species (WoRMS) (“World Register of Marine Species”, 2024)) in het systeem. Het is belangrijk voor het VLIZ om te weten door hoeveel publicaties er naar een project verwezen wordt. Deze maatstaf geeft een indicatie van het draagvlak van elk project binnen de wetenschappelijke gemeenschap en is één van de belangrijkste criteria tijdens projectevaluaties. Daarom is het cruciaal om continu nieuwe publicaties waarin verwezen wordt naar die projecten op te zoeken. Daarvoor wordt op heden Google Scholar gebruikt die bekend staat als de meest uitgebreide en geactualiseerde index. Op die manier blijft IMIS up-to-date en zijn de projectreferenties steeds geactualiseerd.

1.1. Probleemstelling

Momenteel verloopt dit proces binnen het VLIZ grotendeels handmatig. De zoekresultaten, volgens een bepaalde zoekfilter per project, afkomstig van Google Scholar worden manueel gefilterd en de Digital Object Identifier (DOI) van de geselecteerde artikels wordt opgezocht. Vervolgens worden de basisgegevens van elk artikel zoals titel, auteurs, datum en uitgever opgevraagd op basis van de DOI in Crossref (“Crossref”, 2024). Met deze informatie wordt manueel beslist om het artikel al dan niet toe te voegen aan een collectie binnen IMIS.

Dit is een tijdrovend proces. Daarom is er vraag naar automatisatie die de zoekresultaten verwerkt en gestructureerd opslaat. Het systeem moet ook kunnen aangeven of de publicatie reeds in IMIS zit zodat er geen duplicaten opgeslaan worden. De beslissing om een artikel toe te voegen aan IMIS blijft nog altijd een manuele stap, maar naar verwachting moet dit sneller, efficiënter en accurater verlopen. Dat moet mogelijk zijn door de heterogene zoekresultaten om te zetten in gestructureerde informatie en duplicaten op te zoeken.

1.2. Onderzoeksvraag

De probleemstelling vraagt om gedeeltelijke automatisatie. Er wordt aangenomen dat er nog steeds een manuele beslissingsstap zal zijn. Toch streeft dit onderzoek naar zoveel mogelijk automatisatie waarbij het grootste deel van de publicaties automatisch aan IMIS toegevoegd worden en waarbij slecht voor een zo klein mogelijke rest van publicaties een manuele beslissing nodig is.
Bijgevolg luidt de centrale vraag die onderzocht moet worden: “Hoe kunnen de zoekresultaten van Google Scholar automatisch toegevoegd worden aan IMIS?” Dit omvat voornamelijk 2 problemen:

- Hoe kunnen de uiteenlopende zoekresultaten van Google Scholar omgezet worden in gestructureerde data?
- Zijn alle zoekresultaten uniek identificeerbaar zodat er geen duplicaten opgeslaan worden?

De uitgewerkte oplossing zal daarom met volgende aspecten rekening moeten houden:

- Hoe kunnen ook steeds nieuwe zoekresultaten van dezelfde zoekopdracht systematisch opgezocht worden?
- Kan er een score berekend worden hoe relevant elk zoekresultaat is ten opzichte van de zoekopdracht?
- Als er geen unieke identifier is, hoe wordt er dan gecontroleerd op duplicaten?

1.3. Onderzoeksdoelstelling

Het beoogde resultaat van het onderzoek is om het toevoegen van publicaties aan IMIS zoveel mogelijk te automatiseren:

- Een proof-of-concept van de meest geschikte methode om de Google Scholar zoekresultaten om te zetten in gestructureerde data.
- Een proof-of-concept van het proces dat een score voor elk zoekresultaat berekent.
- Een proof-of-concept van het opzoeken van duplicaten.

1.4. Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:
In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.
In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.
Dit wordt verder uitgewerkt in Hoofdstuk 4 voor het systematisch ontvangen van de meest recente publicaties. Hoofdstuk 5 gaat in op het omzetten van de Google Scholar zoekresultaten in gestructureerde data. Hoofdstuk **??** gaat over het berekenen van een score voor elk zoekresultaat. Hoofdstuk 7 onderzoekt hoe voor elk zoekresultaat een unieke identificatie kan gezocht worden. Hoofdstuk 8 bekijkt hoe voor elke publicatie de aanwezigheid van duplicaten opgespoord wordt.
In Hoofdstuk 9 tenslotte, worden alle aspecten samengebracht. Er wordt een besluit geformuleerd in welke mate de onderzoeksvragen beantwoord zijn door het onderzoek. Daarbij wordt ook vermeld wat er verder nog moet gebeuren in de toekomst om het resultaat te verbeteren.

2

Stand van zaken

Wereldwijde onuitputtelijke digitale indexen (waaronder Google Scholar), diep maatschappelijk verankerde sociale media, het Internet Of Things, de toenemende belangstelling in AI, zorgen tezamen voor een gigantische hoeveelheid aan online data. Met deze grote berg van informatie die beschikbaar is op het world wide web, is het bijzonder relevant voor organisaties om te begrijpen hoe ze deze data kunnen ontginnen teneinde er bruikbare informatie uit te filteren (Lotfi e.a., 2021). Een belangrijk begrip dat daarbij prominent op de voorgrond treedt, is 'web scraping' of 'web crawling'. «Web scraping is een geautomatiseerd data extractie proces van websites met gebruik van gespecialiseerde software» (Bhatt e.a., 2023). Maar web scraping is lang niet de enige techniek die gebruikt wordt om data van een webpagina te halen (Gray e.a., 2012). Andere technieken zijn het gebruik van API's, screen readers en het ontleden van online PDF documenten. Het voordeel van web scraping is dat de website niet over een API met ruwe data hoeft te beschikken. De HTML code waarmee de website opgebouwd is, wordt gebruikt door de scraper om de eigenlijke inhoud eruit te filteren.
Web scrapers kunnen gebaseerd zijn op verschillende technologieën zoals 'spidering' en 'pattern matching'. Daarnaast biedt de programmeertaal waarin ze geïmplenteerd zijn ook telkens andere mogelijkheden (Bhatt e.a., 2023). Lotfi e.a. (2021) onderscheidt web scrapers zowel op basis van hun toepassing (vb. medisch, social media, financieel, marketing, onderzoek) als op basis van hun methodiek (vb. copy & paste, HTML parsing, DOM parsing, HTML DOM, reguliere expressies, XPath, vertical aggregation platform, semantic annotation recognizing, computer vision web page analyzer). Hoewel de methodologie kan wijzigen, blijft het uitgangspunt van integriteit, correctheid en betrouwbaarheid van de data steeds van primordiaal belang (Lotfi e.a., 2021). Verder duidt Lotfi e.a. (2021) ook nog op het iteratieve karakter van een web scaper. Het programma wordt gevoed met een url, en zal dan op zijn beurt nieuwe urls zoeken op de pagina zodat die ook bezocht kunnen worden.

Singrodia e.a. (2019) vult bovenstaande kenmerken van web scraping nog verder aan met de systematische omzetting van ongestructureerde gegevens op webpagina's naar gestructureerde data in een databank. De gegenereerde data heeft aanleiding tot filtering of tot statistieken. Het biedt vele voordelen aangezien de data opgeschoond is en daardoor als vrij van fouten kan beschouwd worden. Andere voordelen zijn tijdswinst en centrale opslag wat verdere verwerking ten goede komt.
Daarenboven vergroot web scraping de snelheid en het volume van de data die verwerkt kan worden aanzienlijk, en vermindert het ook het aantal fouten die zouden optreden door menselijke verwerking (Bhatt e.a., 2023). Uiteindelijk zet web scraping de online informatie om in business intelligence afhankelijk van het uitgangspunt van de eindgebruiker.
Na web scrapers besproken te hebben, is het niet onbelangrijk even stil te staan bij de vraag of web scrapen wel legaal is? Volgens Castrillo-Fernández (2015) is daar geen straightforward antwoord op, de situatie is afhankelijk van het betrokken land. Maar over het algemeen zijn de meest reguleringen wel in het voordeel van web scraping. Men moet vooral opletten met het respecteren van het eigendomsrecht wanneer inhoud verworven wordt door scaping en vervolgens verder gebruikt wordt.
Tot slot van de stand van zake omtrent web scraping nog een pleidooi voor Python. Al is het perfect mogelijk om web scraping te ontwikkelen in Php, Java, en andere, toch biedt Python de meest gebruiksvriendelijke tools aan volgens Kumar en Roy (2023). «Beautiful Soup is een van de eenvoudigste bibliotheken om data te scrapen van websites. Een simpele find_all() in Beautiful Soup, is krachtig genoeg om de data uit het gehele document te doorzoeken. Daarna is de taak om de data te structureren. Dat kan in Python aan de hand van Pandas die in staat is om de data in een geordend formaat te presenteren.»
Om verder gebruik te maken van deze data moet deze eerst gestuctureerd opgeslaan worden. Mitchell (2015) reikt een aantal methodes aan waarop dit gedaan kan worden, steeds afhankelijk van het beoogde resultaat. De skills om al die data te beheren en ermee te interageren is zo mogelijk nog belangrijk dan het scrapen op zich. De data waarvan sprake is niet gekenmerkt door strikte relaties, maar stelt eerder een collectie semi gestructureerde data voor. Een document store database lijkt in dit geval het meest aangewezen. Lourenço e.a. (2015) vergelijkt de gangbare NoSQL databases en stelt MongoDB voor als een consistente [1] document store database.
De website in de spotlight voor dit onderzoek is Google Scholar (GS), de grootste bron van wetenschappelijke publicaties op heden. De beta versie verscheen in 2004 en sindsdien wordt het systeem voornamelijk door academici gebruikt om een persoonlijke bibliotheek aan te leggen tezamen met statistieken omtrent cita-

[1] alle clients zien steeds dezelfde data

ties en h-indexen [2]. Volgens Oh en Colón-Aguirre (2019) is er een symbiose tussen GS and de academische wereld waarbij academici gratis hun werk kunnen aanbieden op GS. GS op zijn beurt zorgt voor goede visibiliteit van dat werk door onder andere citaten, referenties en een oplijsting van gelijkaardige onderwerpen. In tegenstelling tot de baanbrekende vernieuwing die GS introduceerde, gaat het raadplegen van deze gegevens wel gepaard met een aantal hindernissen, in het bijzonder op grote schaal.

Web scraping is 1 van de technieken die aangewend worden om dit probleem te omzeilen. Meerdere studies ((PRATIBA e.a., 2018),(Rafsanjani, 2022),(Amin e.a., 2024),(Sulistya e.a., 2024)) gebruiken bovenstaande concepten van web scrapers en gestructureerde opslag om automatisch data te ontleden van GS aan de hand van gebruiksvriendelijke interfaces. Er zijn onderling steeds wel verschillen tussen de gebruikte tools en het gekozen formaat die beide afhankelijk zijn van het beoogde resultaat. Maar het omliggend kader is steeds hetzelfde met gebruik van web scraping en gestructureerde opslag.

Het is evident dat HTML pagina's verwerkt kunnen worden door web scrapers, maar daarnaast zijn ongeveer 70% van de feiten die op het internet gepresenteerd worden, verkregen uit PDF-documenten (Singrodia e.a., 2019). Dit verklaart de noodzaak om zowel HTML pagina's als PDF documenten te scrapen.

Yang e.a. (2017) beschrijft welke HTML elementen belangrijk zijn bij het ontleden van een GS pagina en Rahmatulloh en Gunawan (2020) toont hoe deze kunnen gemapt worden naar de custom code van de scraper.

Al deze gegevens op zich zijn waardevol, maar het is pas door ze te valideren aan de hand van Crossref dat ze ook betrouwbaar worden. Crossref is een onafhankelijke organisatie die instaat voor het beheren van de unieke links naar wetenschappelijke artikels (Hendricks e.a., 2020). Daarvoor wordt de DOI gebruikt die tezamen met alle metadata van een wetenschappelijke publicatie in Crossref bijgehouden wordt. Crossref werkt voor en door alle geregistreerde uitgevers en streeft voortdurend technologische innovatie na teneinde het publicatieproces van wetenschappelijke artikels te verbeteren.

[2] De h-index van een wetenschappelijk onderzoeker komt overeen met de grootste h van het aantal publicaties die minstens h keer geciteerd zijn in ander werk.

3

Methodologie

Om nieuwe publicaties die gerelateerd zijn aan wetenschappelijke projecten toe te voegen aan IMIS, moet er natuurlijk een signaal zijn wanneer dergelijke publicaties beschikbaar zijn. Indexen van academische literatuur zijn daarvoor een geschikte bron en zoals eerder beschreven is Google Scholar een interessant optie. Het is de bedoeling om enkel nieuwe resultaten te ontvangen en bijgevolg vervalt dus de optie om publicaties op te zoeken aan de hand van de Google Scholar zoekpagina. De zoekpagina houdt geen gegevens bij van vorige queries en daarom is het zeer aannemelijk dat dezelfde resultaten zullen voorkomen bij opeenvolgende zoekopdrachten. Google Scholar laat echter toe om meldingen aan te maken voor een bepaalde zoekopdracht. Die stuurt automatisch nieuwe resultaten door per e-mail onder de vorm van de Google Scholar SERP [1]. Het opstellen van een zoekopdracht en het aanmaken van een alert wordt verder uitgewerkt in Hoofdstuk 4.
Google Scholar meldingen zijn e-mails in HTML formaat. Ze bevatten inhoud die overeenkomt met de Google Scholar SERP overeenkomstig de zoekopdracht. De SERP bevat een vaste structuur. Het is een lijst met zoekresultaten die telkens dezelfde elementen bevatten:

- titel
- link naar de webpagina van de publicatie
- auteurs
- tijdschrift
- jaartal
- abstract van de publicatie of een fragment ervan

[1] Search Engine Result Page

Die HTML moet omgezet worden in gestructureerde data door middel van HTML scraping technieken:

- HTML scraping door een LLM [2]
  - online model
    - OpenAI
    - Generieke procedure onafhankelijk van het model
  - lokaal model
- HTML scraping door het parsen van de DOM
  - Beautiful Soup
  - SerpAPI

Web scraping wordt verder uitgewerkt in Hoofdstuk 5.
Elk zoekresultaat is op basis van het algoritme van Google Scholar gematched met de zoekopdracht. Maar wil dat daarom ook zeggen dat de publicatie interessant is voor IMIS? Aan de hand van Natural Language Processing (NLP) wordt een score berekend van de relevantie van het zoekresultaat voor de overeenkomstige collectie in IMIS. NLP wordt verder uitgewerkt in Hoofdstuk 6.
Om te weten of een publicatie echt nieuw is, moet ze ondubbelzinnig geïdentificeerd kunnen worden. Titel, auteurs, tijdschrift, enz. maken een publicatie echter niet uniek. Er kunnen namelijk variante benamingen voorkomen van titels, auteurs, enz. Voor literatuur is het de DOI die een publicatie uniek maakt. Voor elk zoekresultaat wordt er op zoek gegaan naar de DOI aan de hand van een stapsgewijze procedure:

- DOI opzoeken in de link naar de webpagina van de publicatie
- DOI opzoeken in Crossref op basis van de titel
- DOI opzoeken op de webpagina van de publicatie
  - dit kan een HTML pagina zijn
  - dit kan een PDF document zijn
  - dit kan een HTML pagina zijn met een embedded PDF document

Het opzoeken van de DOI wordt verder uitgewerkt in Hoofdstuk 7.
Dan resteert de vraag of de publicatie echt nieuw is, of dat ze reeds aan IMIS toegevoegd werd? Het antwoord daarop is afhankelijk van de beschikbare informatie die tijdens de voorgaande stappen gevonden werd.

- De DOI is gevonden: er kan met 100% zekerheid opgezocht worden of de publicatie reeds in IMIS zit of niet.

[2] Large Language Model

- De DOI is niet gevonden: omwille van de variante benamingen kan niet met volledige zekerheid opgezocht worden of een publicatie reeds in IMIS zit of niet. Wel kan door middel van semantic search op basis van de titel de waarschijnlijkheid berekend worden dat de publicatie reeds in IMIS zit.

Semantic search wordt verder uitgewerkt in Hoofdstuk 8.

4

Google Scholar alert

4.1. Instellen van de zoekopdract & aanmaken van het alert

Om op de hoogte te blijven van nieuwe publicaties kan een index van academische literatuur gebruikt worden. Zoals eerder beschreven is Google Scholar daarvoor een interessante bron.
Het is de bedoeling om alleen nieuwe publicaties te ontvangen die nog niet eerder opgezocht werden. Bijgevolg wordt er beter geen gebruik gemaakt van de Google Scholar zoekpagina, omdat die dezelfde publicaties kan tonen bij opeenvolgende opzoekingen. Daarentegen zijn er Google Scholar alerts. Dat zijn meldingen onder de vorm van automatische e-mails waarin de nieuwe publicaties opgelijst staan. Aangezien de meldingen steeds gekoppeld zijn aan een account, houdt het systeem rekening met de zoekgeschiedenis en zijn de zoekresultaten incrementeel.
Bijlage Google Scholar zoekopdracht geeft een uitvoerige uiteenzetting over het opmaken van een zoekopdracht en het instellen van een melding.
In het kader van deze opdracht werd een nieuw account **google-scholar@marineinfo.org** aangemaakt, als een gedeeld account waar meerdere gebruikers toegang tot hebben. Vervolgens werd een Google account aangemaakt met hetzelfde e-mailadres.
Vervolgens werden meerdere zoekopdrachten opgemaakt telkens voor een bepaald project, en werden deze geäctiveerd als meldingen.
Bijvoorbeeld om relevante publicaties over het VLIZ te vinden, worden volgende zoektermen gebruikt:

- Vlaams Instituut voor de Zee
- Vlaams Instituut van de Zee
- Flanders Marine Institute
- VLIZ

- Simon Stevin
- R/V Simon Stevin
- RV Simon Stevin
- Marine Station Ostend
- Mariene Station Oostende

Vanaf dat moment zal Google Scholar voor elke zoekopdracht dagelijks e-mails met nieuwe zoekresultaten sturen naar het e-mailadres van het account, zolang dat de melding geäctiveerd is.

5

Web scraping

5.1. Inleiding

De e-mails afkomstig van Google Scholar bevatten een lijst met zoekresultaten. Het formaat van de e-mail is HTML en de opmaak van de lijst is gelijkaardig aan die van de Google Scholar resultaten pagina. Dit wordt algemeen benoemd als een SERP en is voor iedereen die vertrouwd is met het internet herkenbaar als de lijst met zoekresultaten van Google (zie figuur 5.1).
De Google Scholar SERP heeft een vaste structuur, namelijk een lijst met zoekresultaten bestaande uit:

- titel
- link naar de webpagina van de publicatie
- auteurs
- naam van het tijdschrift
- jaartal
- abstract of fragment van het abstract

Bovenstaande gegevens moeten uit de SERP gefilterd worden zodat ze opgeslaan kunnen worden voor verder gebruik in de volgende stappen.
Informatie uit HTML pagina's halen is algemeen gekend onder de naam "web scraping" of "web crawling". Deze techniek geniet momenteel veel aandacht omdat hij de gebruiker in staat stelt om veel data te verzamelen. Die data dient dan weer als brandstof voor AI. Het punt is dat in die context meerdere technieken bestaan om dezelfde job toe doen:

- web scraping met gebruik van een LLM [1]

[1] Large Language Model

Big Five personality traits and university students' academic performance: A meta-analysis

S Chen, ACK Cheung, Z Zeng - Personality and Individual Differences, 2025
Meta-analytical evidence on the linkage between university students' academic performance and personality **traits**, considering cultural and contextual factors, remains limited. This meta-analysis reports comprehensive assessment of the ...

Measuring personality Traits: Simulations Exploring (Mis) Alignment between methods and measurands

P Durkee - Journal of Research in Personality, 2025
... Measuring personality **traits** requires that measurement methods align with intended measurands. There are two broad classes of personality **trait** ... The simulations highlight challenges in capturing personality **trait** measurands and offer ...

Landscape modification and species traits shape seasonal wildlife community dynamics within an arid metropolitan region

JD Haight, SJ Hall, JS Lewis - Landscape and Urban Planning, 2025
... For this study, we evaluated how landscape characteristics and species **traits** influenced habitat use (ie, patch occupancy, persistence, and ... However, habitat use further varied according to species **traits**, with larger-bodied species exhibiting ...

[HTML] Consequences of Dietary Olive Leaf Powder Supplementation on Growth Performance, Carcass Traits, Blood Biochemical Parameters and Gut Microbiota in Broilers

MH Negm, AK Aldhalmi, EA Ashour, LA Mohamed... - Poultry Science, 2025
This experiment examined the potential of olive leaf powder (OLP) as a natural growth enhancer in broiler feed and its influences on growth performance, carcass characteristics, blood parameters, and intestinal bacterial count. A total of 210 one-day-old ...

[HTML] ... -scale integration of meta-QTL and genome-wide association study identifies genomic regions and candidate genes for photosynthetic efficiency traits in bread wheat

M Chen, T Chen, L Yun, Z Che, J Ma, B Kong, J Long... - BMC Genomics, 2025
... efficiency, a complex **trait**, through the ... in **trait** coverage, demonstrating significant advantages in data size and **trait** resolution dimensions. The increasing and intensive application of MQTL has effectively revealed the complex genetic ...

[HTML] Comprehensive evaluation of sugar beet varieties based on genotype× yield× trait (GYT) in different environments

D xinwang, L yuhang, H xiaohang, L yanli - Scientific Reports, 2025
... The aim of this study was to assess the agronomic **traits** and genotype by yield × **trait** (GYT) interactions of sugar beet across ... **trait** combinations was analyzed using the GYT biplot technique. Principal components PC1 and PC2 accounted for ...

[HTML] A Unified Method for Detecting Phylogenetic Signals in Continuous, Discrete, and Multiple Trait Combinations

L Yao, Y Yuan - Ecology and Evolution, 2025
... We used d phylo to represent phylogenetic distance and d **trait** to represent **trait** distance, and d **trait** was calculated using Gower's distance, which allows our method to handle any type of **trait**. Considering three species i, j, and k, if d ij phylo < ...

... sequence repeat (SSR) markers for Chinese yam (Dioscorea opposita Thunb.) and genetic diversity and association analysis using phenotypic and genotypic traits

Q Zhang, H Tian, S Guo, Y Wang, S Pei, C Wu, Q Wang... - Genetic Resources and ..., 2025
Chinese yam (Dioscorea opposita Thunb.) has a long history of cultivation and application, and is an important medicinal and edible plant in China. However, the lack of molecular markers limits the efficiency and accuracy of the genetic breeding ...

[HTML] Aboveground-canopy and belowground-root-trait correlations contribute

```
<h3 style=̈font-weight:normal;margin:0;font-size:17px;line-height:20px;̈><span
style=̈font-size:11px;font-weight:bold;color:1a0dab;vertical-align:2pẍ>[HTML]</span>
<a href=”https://scholar.google.be/scholar_url?url=https://www.sciencedirect.com/science/article/pii/
GcCSieoPqLGJyAM&amp;scisig=AFWwaebeqxEetR78Fz7JpxPtT7ui&amp;oi=scholaralrt&amp;hist=
kY6yEQWAvLJNk68&amp;html=&amp;pos=0&amp;folt=kw-
topc̈lass=g̈se_alrt_titles̈tyle=̈font-size:17px;color:1a0dab;line-height:22pẍ>The role
of narcissistic personality <b>traits </b>in bullying behavior in adolescence–A
systematic review and meta-analysis</a></h3>
```

Codefragment 5.1: HTML fragment van de titel van een publicatie.

  - online model
  - lokaal model
- web scraping met verwerking van de DOM
  - Beautiful Soup
  - SerpAPI

5.2. Web scraping met gebruik van een LLM

De meest gekende vorm van web scraping gebruikt de DOM structuur om er de inhoud uit te filteren. Daar komt dus niets van AI bij kijken. Het probleem met die aanpak is dat de custom code die de HTML structuur verwerkt, sterk afhankelijk is van de HTML zelf. Bijvoorbeeld wanneer de titel van een publicatie tussen <h3> tags staat die gekenmerkt worden door een class=”gse_alrt_title”(zie codefragment 5.1), dan zal de custom code specifiek filteren op die DOM elementen. Maar wanneer de structuur van de HTML wijzigt om welke reden dan ook, dan zal de web scraper niet langer werken zolang de custom code niet werd aangepast.
Daarom begint dit hoofdstuk met meer recente technieken die gebaseerd zijn op AI en die niet strikt afhankelijk zijn van de DOM structuur.

5.2.1. Web scraping met OpenAI

LLMs zijn bijzonder goed in het beantwoorden van vragen. OpenAI is voor het brede publiek beter gekend door zijn chatbot ChatGPT. Maar hoe goed is OpenAI in het parsen van een webpagina? OpenAI biedt ook een API (“OpenAI developer platform”, 2025) aan waarmee gebruikers opdrachten kunnen sturen naar een model. De opdracht/vraag in kwestie is: “Geef de titels, orginele links, auteurs, naam van de tijdschriften, jaartal van de publicaties en tekstfragmenten van hetvolgende Google Scholar zoekresultaat.”
Het online artikel “Web scraping experiment with AI (Parsing HTML with GPT-4 and GPT-4o)” (2025) beschrijft stap voor stap hoe de gevraagde gegevens verkregen kunnen worden met het “gpt-4-1106-preview” model van OpenAI. De code op git-

```
        # Remove all occurrences of content between <head> and </head>
        body_text = re.sub(r'<head.*?>.*?</head>', '', body_text,
            flags=re.DOTALL)
        # Remove all occurrences of content between <script> and
            </script>
        body_text = re.sub(r'<script.*?>.*?</script>', '', body_text,
            flags=re.DOTALL)
        # Remove all occurrences of content between <style> and
            </style>
        body_text = re.sub(r'<style.*?>.*?</style>', '', body_text,
            flags=re.DOTALL)
```

Codefragment 5.2: Codefragment voor het opkuisen van de HTML.

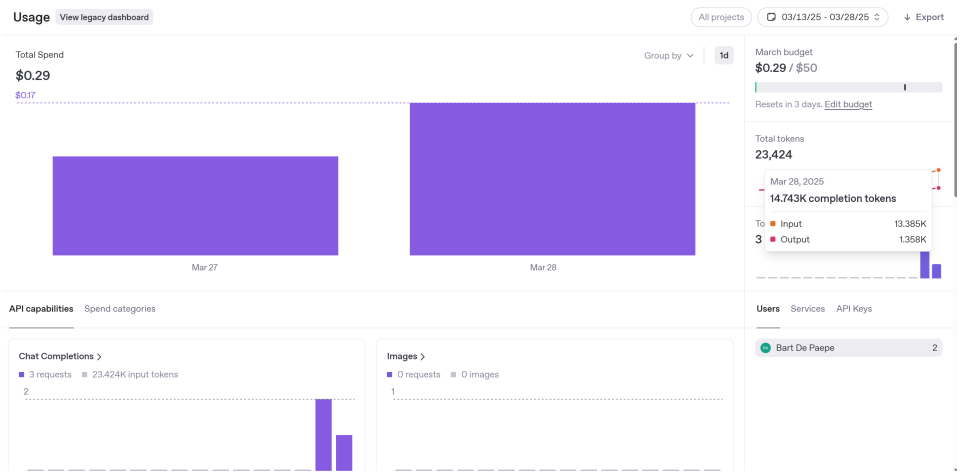
```
        messages=[
        {"role": "system",
            "content": "You are a master at scraping Google Scholar
                results data. Scrape top 10 organic results data
                from Google Scholar search result page."},
        {"role": "user", "content": body_text}
        ],
```

Codefragment 5.3: Codefragment voor het opstellen van een prompt.

hub “Web scraping Google Scholar SERP using OpenAI” (2025) maakt hiervan een implementatie aangepast voor de Google Scholar alerts.
De HTML wordt eerst ontdaan van overbodige tags zodat alleen de inhoud behouden blijft zoals te zien is in codefragment 5.2 De prompt voor het model is te zien in codefragment 5.3. De verwachte parameters die het model moet zoeken zijn te zien in codefragment 5.4.
Het is vereist om een account te registreren bij OpenAI en om dit account te crediteren. Vervolgens kan er met de API gewerkt worden en wordt er betaald naargelang het verbruik. Een test met 1 Google Scholar alert met 10 zoekresultaten heeft een prijs van 0,17$ zoals te zien is in figuur 5.2. Het model gebruikte daarvoor 14743 tokens. De test was succesvol. Het model vond de gevraagde parameters voor elk van de 10 zoekresultaten. Toch houdt het onderzoek naar web scraping hier niet op. Het resultaat tot zover is immers betalend en dat is niet noodzakelijk de beste oplossing voor het probleem. Bovendien werkt deze oplossing enkel met OpenAI modellen wat weinig flexibiliteit toelaat.

```
        "function": {
            "name": "parse_data",
            "description": "Parse organic results from Google Scholar
                SERP raw HTML data nicely",
            "parameters": {
                'type': 'object',
                'properties': {
                    'data': {
                        'type': 'array',
                        'items': {
                            'type': 'object',
                            'properties': {
                                'title': {'type': 'string'},
                                'original_url': {'type': 'string'},
                                'authors': {'type': 'string'},
                                'year_of_publication': {'type':
                                    'integer'},
                                'journal_name': {'type': 'string'},
                                'snippet': {'type': 'string'}
                            }
                        }
                    }
                }
            }
        }
```

Codefragment 5.4: Codefragment voor het opstellen van de zoekopties.



Figuur 5.2: OpenAI dashboard.

```
        class FieldDefinition(BaseModel):
        """Define the fields to extract from the webpage."""

        name: str = Field( ... , description="The desired name for this
            field.")
        type: Literal["str", "int", "float", "bool", "list"]
```

Codefragment 5.5: Codefragment voor het opstellen van een Pydantic Field.

5.2.2. Web scraping met Mirascope en Anthropic

De vorige scraper was zeer specifiek geschreven voor OpenAI. De code zou niet werken voor een model van een andere provider. Daarom wordt er verder gezocht naar een meer generieke methode die niet afhankelijk is van de provider van het model. Mirascope "LLM abstractions that aren't obstructions" (2025) is een gespecialiseerde code library om op uniforme wijze met verschillende LLMs te werken. Het online artikel "Extracting Structured Data from Websites with Mirascope, Anthropic, and Streamlit" (2025) beschrijft stap voor stap hoe de gevraagde gegevens verkregen kunnen worden aan de hand van Mirascope en het Anthropic Claude model. Net zoals OpenAI biedt Anthropic ook een API ("Using the API Getting started", 2025) aan voor development. Er moet eveneens een account geregistreerd worden bij Anthropic maar hun free tier is uitgebreider dan dat van OpenAI zodat het account voor dit experiment niet gecrediteerd moest worden.
De code op github "Web scraping Google Scholar SERP using Anthropic" (2025) maakt hiervan een implementatie aangepast voor de Google Scholar alerts. De parameters die opgezocht moeten worden, dienen eerst als velden en schema gedeclareerd te worden. Dit is te zien in codefragmenten 5.5 en 5.6. Met deze structuur van veld en schema kan vervolgens voor elke opdracht een gepast model gemaakt worden aan de hand van de code in codefragment 5.7 Dan kan er geparsed worden met de resultaten volgens het gevraagde schema zoals getoond wordt in codefragment 5.8.
Ook deze test was succesvol. Het model vond de gevraagde parameters voor elk van de 10 zoekresultaten. Toch houdt het onderzoek naar web scraping ook hier niet op. De oplossing tot zover heeft nog steeds een beperkte free tier en meerdere opdrachten zouden ook aanleiding geven tot een betalende service. Daarnaast zijn tot zover beide oplossingen ook afhankelijk van een online service wat in bepaalde gevallen niet wenselijk kan zijn (bijvoorbeeld wanneer de klant een on-premise oplossing wil).

```
class SchemaGenerator(AnthropicExtractor[list[FieldDefinition]]):
"""Generate a schema based on a user query."""

api_key = settings.anthropic_api_key

extract_schema: Type[list] = list[FieldDefinition]

prompt_template = """
Call your tool with field definitions based on this query:
{query}
"""

query: str
```

Codefragment 5.6: Codefragment voor het opstellen van een Pydantic Schema.

```
    def generate_schema(self) -> None:
"""Sets `extract_schema` to a schema generated based on `query`."""
field_definitions = SchemaGenerator(query=self.query).extract()
model = create_model(
"ExtractedFields",
__doc__=DEFAULT_TOOL_DOCSTRING,
**{
    field.name.replace(" ", "_"): (field.type, ...)
    for field in field_definitions
},
)
self.extract_schema = list[model]
```

Codefragment 5.7: Codefragment voor het opstellen van een Pydantic Model.

```
        @computed_field
        @property
        def webpage_content(self) -> str:
        """Returns the text content of the webpage found at `url`."""
        soup = BeautifulSoup(self.html_text, "html.parser",
            from_encoding="utf-8")
        text = soup.get_text()
        for link in soup.find_all("a"):
        text += f"\n{link.get('href')}"
        return text
```

Codefragment 5.8: Codefragment voor het parsen.

docker exec -it ollama ollama pull llama3:8b docker exec -it ollama ollama run llama3

Codefragment 5.9: Ollama

5.2.3. Web scraping met een lokaal model

Om niet afhankelijk te zijn van een online service moet het model lokaal gehost worden. "Ollama: Get up and running with large language models." (2025) is een platform om LLMs te hosten op je eigen systeem. Eenmaal geïnstalleerd kan een model naar keuze gedownload en uitgevoerd worden op de lokale machine. Dit kan bijvoorbeeld aan de hand van Docker zoals te zien is codefragment 5.9. Het is daarbij aan de gebruiker om te kiezen welk model het best geschikt is voor het probleem, maar ook om de vereisten van het model en de beschikbare hardware op elkaar af te stemmen. Er is een groot aanbod van beschikbare modellen, maar hieronder staan de 2 modellen die gebruikt werden voor dit onderzoek.

- LLaMA — geschikt voor een brede waaier aan toepassingen, LLaMA blinkt uit in het begrijpen en het genereren van tekst die de menselijke taal nabootst ("Introducing Meta Llama 3: The most capable openly available LLM to date", 2025).
- Gemma — biedt een lichtere versie van een model dat ook geschikt is voor web scraping ("Gemma3: Powerful AI Within Everyone's Reach", 2025).

Ieder van deze modellen is beschikbaar in verschillende versies van grootte van het model. Een groot model heeft meer kennis en zal beter op de vraag kunnen antwoorden. Anderzijds zal de verwerking trager verlopen bij een groot model en zal een groot model veel meer resources nodig hebben om te kunnen werken. Daarom wordt in eerste instantie met een klein llama3:8b model (4.7GB) gewerkt dat kan uitgevoerd worden op een doorsnee laptop. Het online artikel "Scrape-graph AI Tutorial; Scrape websites easily with LLaMA AI" (2025) beschrijft stap voor

```
graph_config = {
    "llm": {
        "model": "ollama/gemma3:27b",  # Specifies the large language model to use
        "temperature": 0,          # Temperature controls randomness; 0 makes it deterministic
        "format": "json",          # Output format is set to JSON
        "base_url": "http://localhost:11434",  # Base URL where the Ollama server is running
        "model_tokens": 128000,
    },
    "embeddings": {
        "model": "ollama/nomic-embed-text",  # Specifies the embedding model to use
        "temperature": 0,                    # Keeps the generation deterministic
       "base_url": "http://localhost:11434",  # Base URL for the embeddings model server
    },
    "verbose": True,  # Enables verbose output for more detailed log information
}
```

Codefragment 5.10: Codefragment voor het configureren van Scrapegraph.

stap hoe webpagina's gescraped kunnen worden aan de hand van ScrapegraphAI en het LlaMa3 model. "ScrapeGraphAI: You Only Scrape Once" (2025) is een library die LLMs en grafen combineert om webpagina's te scrapen.
De code op github "Web scraping Google Scholar SERP using Llama3" (2025) maakt hiervan een implementatie aangepast voor de Google Scholar alerts. De code veronderstelt dat ollama draait op het systeem en dat het LlaMa3 model erop uitgevoerd wordt. Codefragment 5.10 connecteert de scraping graph met het model. De prompt voor het model is te zien in codefragment 5.11.
De test was niet succesvol. Het model vond geen enkele van de gevraagde parameters voor geen enkele van de 10 zoekresultaten. In eerste instantie werd gedacht dat het geselecteerde model te zwak is voor deze taak omdat er voor een klein model gekozen werd. Daarom werd dezelfde test uitgevoerd op een krachtiger systeem met 40GB RAM met het gemma3:27b model (17GB). Maar ook deze test was niet succesvol: de opdracht bleef gedurende 1 uur lopen zonder resultaat, waarna het commando manueel onderbroken werd.

```
        smart_scraper_graph = SmartScraperGraph(
        prompt="You are an AI assistant specialized in processing
            Google Scholar search engine result pages and returning
            structured JSON data. Always provide your response as
            valid, well-formatted JSON without any additional text
            or comments. Focus on extracting and organizing the most
            relevant information from Google Scholar search engine
            result pages, including title, original url, authors,
            name of journal, year of publication, snippet.",  # The
            AI prompt specifies what to extract
        source=body_text,  # URL of the website from which to scrape
            data
        config=graph_config,  # Uses predefined configuration
            settings
        )

        # Execute the scraping process
        result = smart_scraper_graph.run()

```

Codefragment 5.11: Codefragment voor het opstellen van een prompt.

Figuur 5.3: HTML structuur van de GS alert.

5.3. Web scraping via het parsen van de DOM

Tot zover is er nog geen bruikbaar resultaat. Het gebruik van een online model is betalend en kan niet gewenst zijn door de klant. Anderzijds leidt een lokaal model voorlopig niet tot een goed resultaat. Daarom wordt er teruggegrepen naar de meer klassieke benadering van web scraping waarbij de HTML structuur ontleed wordt aan de hand van custom code. De Google Scholar SERP is een gestructureerde HTML pagina. Voor het parsen wordt er gebruik gemaakt van deze structuur. Elk afzonderlijk resultaat heeft een H3-tag met als klasse 'gse_alrt_title' waarin de titel staat. De korte tekst van het zoekresultaat staat in een DIV-tag met als klasse 'gse_alrt_sni'. Tussen de titel en het snippet staan de auteurs, de uitgever en het jaartal gegroepeerd in een DIV-tag zoals te zien is in figuur 5.3.

Meerdere specifieke libraries laten toe om HTML te parsen, maar de meest gekende bibliotheek is Beautiful Soup ("Beautiful Soup", 2025). Door de naam van de HTML-tag en de gebruikte klasse (gse_alrt_title, en gse_alrt_sni) mee te geven aan Beautiful Soup, worden de overeenkomstige elementen in het HTML-document weergegeven. Beautiful Soup laat ook toe om te navigeren in het document zodat het element met de auteurs, de uitgever en het jaartal gemakkelijk gevonden kan worden.

De code op github "Web scraping Google Scholar SERP using Beautiful Soup" (2025) maakt hiervan een implementatie aangepast voor de Google Scholar alerts. De selectie van tags aan de hand van Beautiful Soup is te zien in codefragment 5.12 De navigatie tussen tags aan de hand van Beautiful Soup is te zien in codefragment 5.13 De volledige uitwerking is te zien in bijlage Parse Google Scholar zoekopdracht. De test met een SERP met 10 zoekresultaten was geslaagd. De gevraagde parameters worden gevonden voor elk van de 10 resultaten.

```
        ...
        soup = BeautifulSoup(body_text, "html.parser",
            from_encoding="utf-8")
        all_titles = soup.find_all("a", {"class": "gse_alrt_title"})
        all_snippets = soup.find_all("div", {"class":
            "gse_alrt_sni"})

        ...

        for i in range(0, len(all_titles)):
        title = all_titles[i].get_text()
        snippet = all_snippets[i].get_text()
        try:
        data = self.parse_search_result(email_id, all_titles[i],
            all_snippets[i])
        ...
```

Codefragment 5.12: Codefragment voor het selecteren van tags aan de hand van Beautiful Soup.

```
        ...
        media_type = title.find_previous()
        if media_type.name.lower() == "span":

        ...

        author_publisher_year = snippet.find_previous()
        author_publisher_year = author_publisher_year.get_text()
        ...
```

Codefragment 5.13: Codefragment voor het navigeren tussen tags aan de hand van Beautiful Soup.

5.4. Meest geschikte Web scraping

Tenslotte moet ook nog SerpAPI (“Scrape Google and other search engines from our fast, easy, and complete API.”, 2025) vermeld worden. Tijdens het onderzoek naar web scraping kwam dit platform meermaals naar voren, zelfs voor de toepassingen die gebruik maken van AI. SerpAPI biedt specifieke oplossingen voor elke mogelijke search engine, waaronder ook Google Scholar. Alleen focust hun product telkens op de online search engine en niet op alerts waardoor de zoekresultaten niet incrementeel zijn.

Tot besluit van dit hoofdstuk wordt er dus verder gewerkt met het parsen van de zoekresultaten aan de hand van Beautiful Soup. De grote tekortkoming dat de code afhankelijk is van de HTML wordt verzacht doordat die HTML ook weerspiegeld wordt in de code. Aanpassingen zouden dus begrijpbaar en beperkt moeten blijven.

Op een dag gaat de code dus breken en het zou goed zijn mocht men hiervan op de hoogte gebracht worden. Daarvoor kan er een e-mail notificatie geïmplementeerd worden die een bericht stuurt wanneer Beautiful Soup de gewenste tags in het HTML document niet langer vindt.

6

Natural Language Processing

6.1. Inleiding

Google Scholar heeft feitelijk al gezocht naar publicaties die relevant zijn voor onze zoekopdracht. De sortering van de zoekresultaten is op basis van de omvang van de integrale tekst, het aanzien van het tijdschrift en van de auteurs en het aantal citaties (inclusief de gedateerdheid ervan). Op die manier volgt Google Scholar de manier van werken binnen de academische wereld.
Maar dit laat nog steeds ruimte om voor elk zoekresultaat een score af te leiden die aangeeft hoe relevant de publicatie is voor IMIS. De zoekopdracht voor het VLIZ bijvoorbeeld gebruikt als 1 van de trefwoorden “Simon Stevin”, zijnde het wetenschappelijk vaartuig van het VLIZ. Dat betekent dat Google Scholar ook zoekresultaten zal geven van publicaties die verwijzen naar “Simon Stevin”, de vlaamse geleerde uit de 16de eeuw. Het is interessant om die artikels te kunnen markeren met een lagere score ten opzichte van publicaties die het trefwoord “VLIZ” bevatten.
Daar bestaan Natural Language Processing (NLP) technieken voor. NLP is een verzamelnaam van een hele groep toepassingen en algorithmes die tekst omzetten in informatie. Deze zijn zeer divers en daarom ook onderverdeeld in verschillende deelgebieden. Het bepalen van de relevantie van een tekst in functie van een trefwoord valt eerder onder het deelgebied van de “Natural Language Understanding”.

6.2. Relevantiescore

Het uitgangspunt is om een score te berekenen door te tellen hoe vaak een trefwoord voorkomt in de tekst. De frequentie is dan rechtevenredig met de relevantie van die tekst. Een eenvoudige techniek voor het bepalen van de frequentie is aan de hand van een “Bag of Words” (BoW). Dat is een tabel met een rij voor elke tekst en een kolom voor elk uniek woord. De cellen tonen het aantal keer dat het

	I	like	this	article	medium	data
I like this article	1	1	1	1	0	0
I like medium	1	1	0	0	1	0
I like data	1	1	0	0	0	1

Figuur 6.1: Bag of Words.

$$score = tf * idf \tag{6.1}$$

where

$$tf = term\ frequency\ (see\ above) \tag{6.2}$$

$$idf_t = log\left(\frac{N}{df_t}\right) \tag{6.3}$$

$$N = total\ number\ of documents \tag{6.4}$$

$$df_t = the\ number\ of\ documents\ in\ which\ term\ t\ occurs \tag{6.5}$$

Codefragment 6.1: term frequency-inverse document frequency

woord voorkomt in de tekst. Dit is te zien in figuur 6.1. Toch is de frequentie op zich nog niet de beste waardemeter voor een teksts. Er kunnen namelijk woorden heel vaak voorkomen die op zich weinig vertellen over het onderwerp. Om daaraan tegemoet te komen bestaat er een variant van de BoW, die in plaats van gewoon te tellen de “term frequency-inverse document frequency” (Tf-Idf) geeft. De formules zijn te zien in codefragment 6.1 en het resultaat in figuur 6.2. Maar de relevantie van een publicatie kan niet afhangen van het aantal zoekresultaten, daarom valt de “inverse document frequency” weg. In de plaats zijn er andere parameters om de relevantie te benaderen.

- De topic-sentence ratio: Een ratio van het trefwoord count ten opzichte van het totaal aantal zinnen.
- De topic-noun ratio: Een ratio van het trefwoord count ten opzichte van het totaal aantal zelfstandige naamwoorden.

Aan de hand van die parameters kan een relevantiescore berekend worden zoals

	I	like	this	article	medium	data
I like this article	0,35	0,35	0,35	0,35	0	0
I like medium	0,35	0,35	0,35	0	0,35	0
I like data	0,35	0,35	0,35	0	0	0,35

Figuur 6.2: term frequency-inverse document frequency.

$$score = log(topiccount * topic - sentenceratio * topic - nounratio) \quad (6.6)$$

Codefragment 6.2: relevantiescore

te zien in codefragment 6.2. De code op github “Natural Language Processing of Google Scholar search results” (2025) maakt hiervan een implementatie aangepast voor de Google Scholar alerts.

6.3. Tekstverwerking

De BoW geeft wel aanleiding tot tabellen met zeer grote dimensies. Hoe meer teksten er zijn, des te uitgebreider zal de woordenschat worden. Daarom wordt de BoW altijd voorafgegaan door tekstverwerking die stopwoorden [1] verwijdert en lemmatisering [2].
Voornaamwoorden kunnen verwijzen naar het trefwoord. Het is dus interessant om ze te vervangen door het trefwoord zelf zodat ze mee een invloed hebben op de score. “Coreference resolution” is daar de geschikte NLP techniek voor. In een eerste fase worden alle gerelateerde voornaamwoorden en zelfstandige naamwoorden opgezocht. In een tweede fase worden alle gevonden voornaamwoorden vervangen door hun bijhorende zelfstandig naamwoord. Daarvoor gebruikt deze techniek een coreference model dat bestaat uit opeenvolgende paren van tokens met een bijhorende coreference score. Het opzoeken van paren is te zien in codefragment 6.3.

[1] Stopwoorden (of, a, the, in ,you, ...) komen vaak voor maar hebben geen toegevoegde waarde over het onderwerp.
[2] Lemmatisering zet woorden om naar hun basisvorm, maar houdt daarbij rekening met de context. Voorbeeld “caring” wordt omgezet in “care” en niet in “car”.

```
import spacy
...
    def __init__(self, db_service: DBService,
        logging_service: LoggingService):
self.db_service = db_service
self.logging_service = logging_service
self.nlp = spacy.load("en_coreference_web_trf")
...
    def coreference_resolution(self, text):
doc = self.nlp(text)
spans = doc.spans
span_array = []
self.logging_service.logger.debug(spans)
for spangroup in spans.values():
span_tuple = []
for span in spangroup:

    self.logging_service.logger.debug(text[span.start_char:span.end_char])
span_tuple.append(text[span.start_char:span.end_char])
span_array.append(span_tuple)
return span_array
```

Codefragment 6.3: Coreference resolver

7

Linked data

7.1. Inleiding

De web scraping leverde gegevens zoals titel, auteurs, naam van tijdschrift en jaartal van de publicatie op. Deze informatie heeft echter geen garantie van uniciteit. Dat is te wijten aan mogelijke varianten (zoals bijvoorbeeld afkortingen) in de titel, auteurs en naam van tijdschriften.
Nochtans moet een publicatie uniek geïdentificeerd kunnen worden om uit te sluiten dat er reeds duplicaten in IMIS zitten. De DOI vormt de unieke indentifier van elke publicatie. Er zal getracht worden om deze voor elke publicatie op te zoeken. Hiervoor worden 3 afzonderlijke bronnen geraadpleegd in chronologische volgorde en zolang de DOI niet gevonden werd:

- In de URL van de link naar de originele publicatie
- Aan de hand van een opzoeking in Crossref op basis van de titel
- Op de originele webpagina van de publicatie

7.2. De DOI opzoeken in de URL

De web scraping vond ook de URL naar de webpagina van de oorspronkelijke publicatie. Empirisch valt het op dat veel van die URLs opgebouwd zijn met de DOI van de publicatie (voorbeeld in codefragment 7.1).
Alle DOIs hebben dezelfde structuur: ze beginnen met het cijfer 10, gevolgd door een punt en 4 tot 9 cijfers, daarna volgt een slash. Verder kan elke willekeurige opeenvolging van letters, cijfers, speciale tekens en slashes voorkomen. De volgende

```
https://pubs.acs.org/doi/full/10.1021/acsami.4c21991
```

Codefragment 7.1: Originele URL van de publicatie

lijst reguliere expressies is aanbevolen om DOIs op te zoeken “DOIs and matching regular expressions” (2025):

- `/^10.\d4,9/[-._;()/:A-Z0-9]+$/i`
- `/^10.1002/[^\s]+$/i`
- `/^10.\d4/\d+-\d+X?(\d+)\d+<[\d\w]+:[\d\w]*>\d+.\d+.\w+;\d$/i`
- `/^10.1021/\w\w\d++$/i`
- `/^10.1207/[\w\d]+\&\d+_\d+$/i`

Indien 1 van deze reguliere expressies matcht met de URL, dan is de DOI gevonden. De omzetting in code is te zien in codefragment 7.2

7.3. De DOI opzoeken in Crossref

Indien de voorgaande stap geen DOI opleverde, dan wordt hier op basis van de titel een opzoeking van de DOI gedaan in Crossref. Daarvoor beschikt Crossref over een API (“Tips for using the Crossref REST API”, 2025). De documentatie leert dat ondoordachte requests kunnen leiden tot zeer langdurige queries en/of ongewenste resultaten. Concreet wordt er afgeraden om meer dan 2 velden op te nemen in de query van een sample, of om meer dan 2 resultaten te vragen in het geval van een matching. Verder wordt voor matching aangeraden om te zoeken aan de hand van de bibliografische gegevens zoals te zien in codefragment 7.3. Maar Google Scholar alerts geven geen bibliografie, daarom wordt enkel de titel verder gebruikt. Het is niet nodig om de API rechtstreeks te bevragen aangezien meerdere onafhankelijke Python libraries daar een wrapper voor geschreven hebben:

- Crossref Commons for Python (“Crossref Commons for Python”, 2025)
- Habanero (“Habanero”, 2025)
- Crossrefapi (“Crossref API”, 2025)

Al deze bibliotheken bieden dezelfde tools en presteren gelijkaardig. Zonder bijzondere reden, behalve dat Crossref Commons ontwikkeld wordt door Crossref zelf, wordt er met Crossref Commons for Python gewerkt. De code om een sample op te vragen op basis van de titel is bijzonder compact zoals te zien in 7.4 Indien de Crossref API een resultaat geeft, dan is de DOI gevonden.

7.4. De DOI opzoeken op de webpagina van de publicatie

Indien de voorgaande stap geen DOI opleverde, dan wordt het DOI gezocht op de webpagina van de publicatie. In de meeste gevallen is dat op de website van de uitgever van het tijdschrift. Op die pagina staan altijd de titel, auteurs, naam

```
        def search_in_text(text, link):
        # find using regex
        patterns = get_patterns()
        doi_result = None
        while (doi_result is None) and (len(patterns) > 0):
        doi_result = re.search(patterns.pop(), text, re.IGNORECASE)

        if doi_result is not None:
        # update DOI
        doi = doi_result.group(0)
        link.doi = doi
        link.is_doi_success = True
        link.log_message = "DOI successfully retrieved"

        ...

        def get_patterns():
        # https://www.crossref.org/blog/dois-and-matching-regular-
            expressions/
        # /^10.\d{4,9}/[-._;()/:A-Z0-9]+$/i
        # /^10.1002/[^\s]+$/i
        # /^10.\d{4}/\d+-
            \d+X?(\d+)\d+<[\d\w]+:[\d\w]*>\d+.\d+.\w+;\d$/i
        # /^10.1021/\w\w\d++$/i
        # /^10.1207/[\w\d]+\&\d+_\d+$/i

       patterns = [r"10.1207/[\w\d]+\&\d+_\d+", r"10.1021/\w\w\d++",
        r"10.\d{4}/\d+-\d+X?(\d+)\d+<[\d\w]+:[\d\w]*>\d+.\d+.\w+;\d",
            r"10.1002/[^\s]+",
        r"10.\d{4,9}/[-._;()/:A-Z0-9]+"]
        return patterns
```

Codefragment 7.2: Codefragment voor het opzoeken van DOI in de URL van de publicatie.

http://api.crossref.org/works?query.bibliographic="Toward a Unified Theory of High-Energy Metaphysics, Josiah Carberry 2008-08-13"rows=2

Codefragment 7.3: Query op basis van de bibliografie

```
                try:
        filter = {}
        queries = {'query.title': title}
        response = crossref_commons.sampling.get_sample(size=2,
            filter=filter, queries=queries)
```

Codefragment 7.4: Codefragment voor opvragen van een sample op basis van de titel aan Crossref.

van het tijdschrift, jaartal en abstract van de publicatie. Soms staat ook de DOI op die pagina. In sommige gevallen is de integrale tekst van de publicatie hier beschikbaar als HTML of als PDF document.
Het formaat van de pagina kan verschillende zijn:

- Een gewone HTML pagina
- Een PDF document
- Een webpagina met een embedded PDF document

Voor elk van de 3 gevallen is er een andere verwerking nodig:

- In het geval van een HTML pagina wordt de inhoud geparsed met Beautiful Soup net zoals dat eerder gebeurde voor de SERP. Vervolgens wordt een DOI opgezocht in de inhoud door middel van de reguliere expressies uit hoofdstuk 7.2.
- In het geval van een PDF document is er een extra tussenstap nodig. De inhoud moet gelezen worden met gebruik van PyMuPDF (“PyMuPDF is a high-performance Python library for data extraction, analysis, conversion manipulation of PDF (and other) documents.” 2025). Daarna wordt er ook gezocht aan de hand van de reguliere expressies. Voorbeeld in codefragment 7.5.
- In het geval van een embedded PDF is de inhoud niet onmiddellijk beschikbaar. De gebruiker moet als het ware nog een handeling verrichten (vb. op een knop klikken) alvorens toegang te krijgen tot de inhoud. Dat gaat niet voor een geäutomatiseerd script, maar door middel van Selenium (“Selenium with Python”, 2025) kan de gebruikersinteractie nagebootst worden zodat de embedded content toch automatisch gedownload wordt. Voorbeeld in codefragment 7.6. Eenmaal gedownload kan het bestand gewoon geopend worden en doorzocht worden naar een DOI op dezelfde manier als voor PDF documenten.

Er is geen garantie dat de DOI op de webpagina van de publicatie gevonden wordt. Anderzijds is het ook mogelijk dat er meerdere verschillende DOIs gevonden worden. Voor een mens is het vaak evident om te weten welke DOI dan juist is, maar

```
            def search_in_pdf(pdf, link):
            doc = pymupdf.Document(stream=pdf)
            # Extract all Document Text
            text = chr(12).join([page.get_text() for page in doc])
            patterns = get_patterns()
            doi_result = None
            while (doi_result is None) and (len(patterns) > 0):
            pattern = re.compile(patterns.pop(), re.IGNORECASE)
            doi_result = pattern.search(text)

            if doi_result is not None:
            #update DOI
            doi = doi_result.group(0)
            link.doi = doi
            link.is_doi_success = True
            link.log_message = "DOI successfully retrieved"
```

Codefragment 7.5: Codefragment voor het openen van een online pdf.

voor geautomatiseerd script is daar geen context voor.
Er kan dus besloten worden dat de DOI met grote zekerheid achterhaald kan worden op basis van de URL of adhv. Crossref. In het geval van een opzoeking op de webpagina van de publicatie is de vindkans een pak kleiner.

```
            ...
            url = link.location_replace_url

            options = webdriver.ChromeOptions()

            download_folder =
                os.path.join(str(Path(__file__).parent.parent.parent.parent),
                "online_pdf")

            logging_service.logger.debug(f"Download folder:
                {download_folder}")

            profile = {
                "plugins.plugins_list": [{"enabled": False, "name":
                    "Chrome PDF Viewer"}],
                "download.default_directory": download_folder,
                "download.extensions_to_open": "",
                "download.prompt_for_download": False,
                "download.directory_upgrade": True,
                "plugins.always_open_pdf_externally": True
            }
            options.add_experimental_option("prefs", profile)
            options.add_argument("start-maximized") # open Browser in
                maximized mode
            options.add_argument("disable-infobars") # disabling
                infobars
            options.add_argument("--disable-extensions") # disabling
                extensions
            options.add_argument("--disable-gpu") # applicable to
                windows os only
            options.add_argument("--disable-dev-shm-usage") #
                overcome limited resource problems
            options.add_argument("--no-sandbox") # Bypass OS security
                model
            options.add_argument("--headless")

            logging_service.logger.debug("Downloading file from link:
                {}".format(link.location_replace_url))
            print("Downloading file from link:
                {}".format(link.location_replace_url))

            driver = webdriver.Chrome(options=options)
            driver.get(url)

            logging_service.logger.debug("Status: Download
                Complete.")
            print("Status: Download Complete.")
```

8

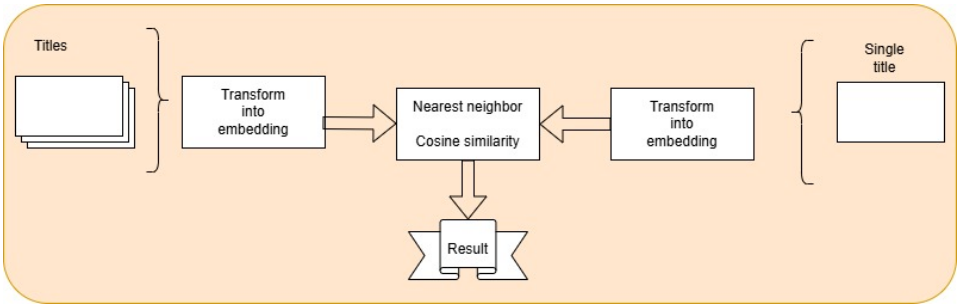
Semantic search

8.1. Inleiding

Alle publicaties in IMIS moeten uniek zijn. Op basis van voorgaande stappen is dat met 100% zekerheid te bepalen in het geval de DOI van de publicatie gevonden werd. Als de DOI niet gevonden werd, kan er niet met zekerheid gesproken worden, maar er kan wel een score berekend worden dat de publicatie reeds in IMIS zit. Dat is mogelijk aan de hand van semantic search.
Op basis van de titel van het artikel wordt een embedding berekend. Vervolgens wordt die vergeleken met alle embeddings van de titels in IMIS om zo tot een score te komen van de gelijkenis tussen beide. Dit wordt voorgesteld in figuur 8.1
Indien de score hoog is dan is er veel waarschijnlijkheid dat de publicatie reeds in IMIS zit. In het geval van een lage score is de waarschijnlijkheid eerder klein.

8.2. Chroma

Embeddings worden opgeslaan in een vector databank. Er zijn talloze producten beschikbaar:



Figuur 8.1: Semantic search.

```
        ...
        chroma_client = chromadb.Client()
        # https://docs.trychroma.com/docs/collections/configure
        self.collection = chroma_client.create_collection(
        name="my_collection",
        metadata={
            "hnsw:space": "cosine"
        }
        )
        ...
```

Codefragment 8.1: Codefragment voor het connecteren met Chroma.

- Pinecone
- Chroma
- Weaviate
- Qdrant
- Milvus
- Vespa
- SingleStore
- Redis
- Elastic Stack
- Mongo
- ...
- (Every database that can store an n-array of numbers)

Chroma ("Chroma", 2025) is een gebruiksvriendelijke database die toelaat om semantic search lokaal te testen zonder extra kosten, zonder account, en zonder installatie van andere software.
Het online artikel "How to Use Chroma to Build Your First Similarity Search" (2025) beschrijft stap voor stap hoe een semantic search met Chroma uitgevoerd kan worden. De code op github "Semantic search using Chroma" (2025) maakt hiervan een implementatie aangepast voor de Google Scholar alerts. De connectie met de Chroma database is te zien in codefragment 8.1. Vervolgens worden alle titels uit IMIS opgevraagd en worden de overeenkomstige embeddings berekend aan de hand van de code in codefragment 8.2 Dan kan een titel opgezocht worden zoals

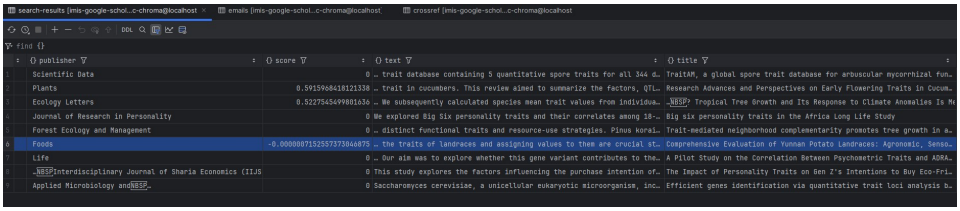
```
        def initialize_embeddings(self):
    result = re-
        quests.get(``https://www.vliz.be/nl/imis?count=2000&module=ref&searchs
    publications = result.json()
    documents = []
    ids = []
    count = 1
    for publication in publications:
    #self.logging_service.logger.debug(publication)
    documents.append(publication['StandardTitle'])
    ids.append(f"id{count}")
    count+=1

    self.collection.add(
    documents=documents,
    ids=ids
    )
```

Codefragment 8.2: Codefragment voor het berekenen van de embeddings.

```
        def do_semantic_search(self, title):
    results = self.collection.query(
    query_texts=[title],  # Chroma will embed this for you
    n_results=2  # how many results to return
    )
    return results['distances'][0][0]
```

Codefragment 8.3: Codefragment voor het opzoeken van een titel.



Figuur 8.2: Chroma resultaat.

```
        ...
        self.model =
            SentenceTransformer("nomic-ai/nomic-embed-text-v1",
            trust_remote_code=True)
        ...
```

Codefragment 8.4: Codefragment voor het configureren van het model met MongoDB.

getoond wordt in codefragment 8.3.
De test was succesvol. Bij wijze van proef werd de titel van 1 van de zoekresultaten toegevoegd aan de lijst van titels uit IMIS. Het resultaat is te zien in figuur 8.2. Deze oplossing voegt Chroma toe als database naast MongoDB die nodig is voor de Web scraping. Het zou beter zijn indien de embeddings ook in MongoDB opgeslaan kunnen worden. Dat wordt verder onderzocht.

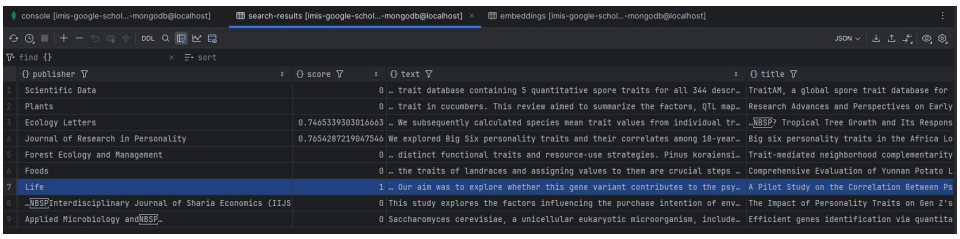
8.3. MongoDB

Atlas Vector Search ("Vector Search", 2025) laat toe om de embeddings tezamen op te slaan met de originele data waar ze van afgeleid zijn.
De online documentatie "Atlas Vector Search Quick Start" (2025) beschrijft stap voor stap hoe een semantic search met Atlas Vector Search lokaal uitgevoerd kan worden. De code op github "Semantic search using MongoDB" (2025) maakt hiervan een implementatie aangepast voor de Google Scholar alerts. De selectie van het model voor de embeddings is te zien in codefragment 8.4. Vervolgens worden alle titels uit IMIS opgevraagd en worden de overeenkomstige embeddings berekend aan de hand van de code in codefragment 8.5 Vervolgens wordt de lijst met embedding geïndexeerd zoals getoond wordt in codefragment 8.6. Dan kan een titel opgezocht worden zoals getoond wordt in codefragment 8.7.
De test was succesvol. Bij wijze van proef werd de titel van 1 van de zoekresultaten toegevoegd aan de lijst van titels uit IMIS. Het resultaat is te zien in figuur 8.3. Deze oplossing breidt de bestaande MongoDB databank uit met embeddings. Voor artikels zonder indentifcatie kan een score berekend worden die duidt op duplicaten in IMIS.

```
        def initialize_embeddings(self):
    docs = []
    data = []
    embeddings = []
    result = requests.get(IMIS)
    publications = result.json()
    for publication in publications:
    data.append(publication['StandardTitle'])
    embed-
        dings.append(self.get_embedding(publication['StandardTitle']))

    for i, (embedding, title) in enumerate(zip(embeddings,
        data)):
    doc = {
        "_id": i,
        "title": title,
        "embedding": embedding,
    }
    docs.append(doc)
    self.db_service.set_collection("embeddings")
    self.db_service.insert_many(docs)
```

Codefragment 8.5: Codefragment voor het berekenen van de embeddings.



Figuur 8.3: Mongodb resultaat.

```
            def create_search_index(self):
        search_index_model = SearchIndexModel(
        definition={
            "fields": [
            {
                "type": "vector",
                "path": "embedding",
                "similarity": "dotProduct",
                "numDimensions": 768
            }
            ]
        },
        name="vector_index",
        type="vectorSearch"
        )
        self.db_service.set_collection("embeddings")
       self.db_service.create_search_index(model=search_index_model)
```

Codefragment 8.6: Codefragment voor het indexeren van de embeddings.

```
        def do_semantic_search(self, title):
    query_embedding = self.get_embedding(title)
    pipeline = [
    {
        "$vectorSearch": {
            "index": "vector_index",
            "queryVector": query_embedding,
            "path": "embedding",
            "exact": True,
            "limit": 5
        }
    },
    {
        "$project": {
            "_id": 0,
            "title": 1,
            "score": {
                "$meta": "vectorSearchScore"
            }
        }
    }
    ]
    self.db_service.set_collection("embeddings")
    result = self.db_service.aggregate(pipeline)
```

Codefragment 8.7: Codefragment voor het opzoeken van een titel.

9

Conclusie

De IMIS collecties van het VLIZ kunnen voortaan semi-automatisch uitgebreid worden door het onderzoek van deze bachelorproef.
Google Scholar alerts zijn een automatische en incrementele bron van gegevens die zullen blijven stromen zolang de melding bestaat.
Web scraping technieken met LLMs verdienen bijzondere aandacht. Online modellen laten toe om een SERP te parsen en bovendien zijn ze bestand tegen interne veranderingen van de SERP. Na dit onderzoek moet er verder getest worden met verschillende lokale modellen op verschillende systemen. Het succes van de online modellen doet toch verwachten dat er ook lokaal een slaagkans moet zijn. Tot dan wordt er gewerkt met Beautiful Soup. Aangezien de structuur van de SERP vast ligt is dit zeker geen slechte oplossing. De code van Beautiful Soup en de HTML structuur gaan hand in hand zodat wijzigingen van die laatste naar verwachting vrij eenvoudig op te vangen zijn in de code. Belangrijk daarbij is dat er een notificatiesysteem voorzien wordt dat afgaat wanneer er zo een wijziging zou optreden.
Het berekenen van een relevantiescore op basis van de frequentie van de zoekopdracht in het zoekresultaat geeft zeker aanleiding tot een nieuwe indicator, maar de mate waarin die effectief ook voorspelt of een publicatie interessant is voor IMIS is zeker vatbaar voor discussie. Verder onderzoek moet dus ook blijven inzetten op de “Natural Language Understanding” om de waarde van een publicatie voor IMIS beter te berekenen.
Het opsporen van de DOI van elke publicatie volgens een stapsgewijze procedure is efficiënt doch niet feilloos. Er blijft een minderheid van publicaties waarvoor de DOI niet bestaat of niet gevonden kan worden. Verder onderzoek moet vooral inzetten op het vinden van de juiste DOI op de webpagina van de publicatie wanneer daar meerdere DOIs aanwezig zijn. Dat is een eenvoudige taak voor een mens, maar dat is het niet voor een computer.
Tenslotte helpt het opzoeken van de gelijkenis tussen de titel van een publicatie

en alle titels in IMIS aan de hand van “Semantic Search” wanneer een manuele beslissing moet genomen worden om een publicatie toe te voegen aan IMIS. Bij een volledige match zal er geen twijfel zijn, maar bij een gedeeltelijke match, bijvoorbeeld in het geval van variaties in de titel, moet er beter onderzocht worden wat de threshold is om te beslissen om een publicatie aan IMIS toe te voegen.

10

Google Scholar zoekopdracht

10.1. Zoeken in Google Scholar ("Google Scholar Guide", 2025)

10.1.1. Basis zoeken

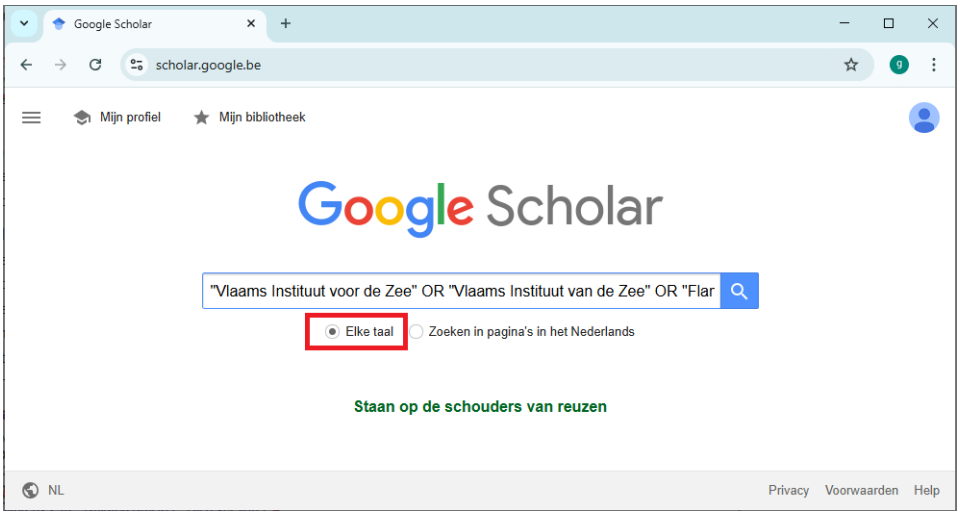
Google Scholar biedt een vertrouwde gebruikersinterface met een inputveld waar de relevante zoektermen ingevuld moeten worden. Default wordt er in elke taal gezocht, maar de gebruiker kan dit beperken tot zijn eigen taal.
Wanneer een zoekopdracht verzonden wordt, dan is er een antwoord binnen de 3 seconden. Het resultaat kan vervolgens verder gefilterd worden:

- **Elke periode**: Dit is de default filter zodat alle resultaten getoond worden ongeacht hun publicatiedatum.
- **Sinds jaar**: Hierbij worden enkel resultaten gefilterd die sinds het gespecifiëerde jaar gepubliceerd werden.
- **Aangepast bereik**: Hierbij worden enkel resultaten gefilterd waarvan de publicatiedatum binnen het gespecifiëerde bereik ligt.
- **Sorteren op relevantie**: Dit is de default filter tezamen met 'Elke periode' die de resultaten sorteert op basis van hun belangrijkheid.[1]
- **Sorteren op datum**: Hierbij worden de resultaten gesorteerd op publicatiedatum.
- **Reviewartikelen**: Hierbij worden enkel state of the art publicaties gefilterd.[2]

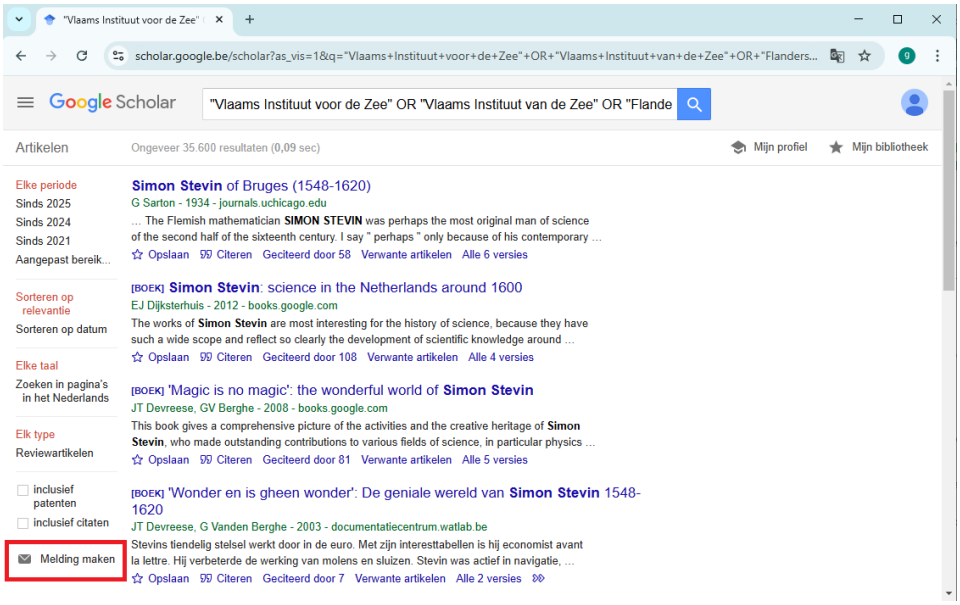
Elk resultaat kan verder uitgediept worden:

[1] De relevantie van elke publicatie wordt in de eerste plaats bepaald door het aantal citaties ((Beel & Gipp, 2009))

[2] Een reviewartikel ondergaat een systematische review door een groep van experten volgens de op dat moment geldende 'State of the art' ((Sataloff e.a., 2021))



Figuur 10.1: Google Scholar user interface voor het basis zoeken van publicaties op basis van de ingevoerde zoektermen.



Figuur 10.2: Google Scholar zoekresultaten op basis van een zoekopdracht.

- **Geciteerd door**: Een oplijsting van publicaties die zelf het artikel citeren. Dit kan leiden tot andere relevante artikels.
- **Verwante artikelen**: Andere artikels in hetzelfde thema. Dit kan leiden tot andere relevante artikels.
- **Alle versies**: Alternatieve locaties waar dezelfde informatie kan teruggevonden worden. Dit kan leiden tot een breder beeld van organisaties, instituten en uitgevers.

10.1.2. Geävanceerd zoeken

Google Scholar heeft ook een meerdere geävanceerde zoekopties:

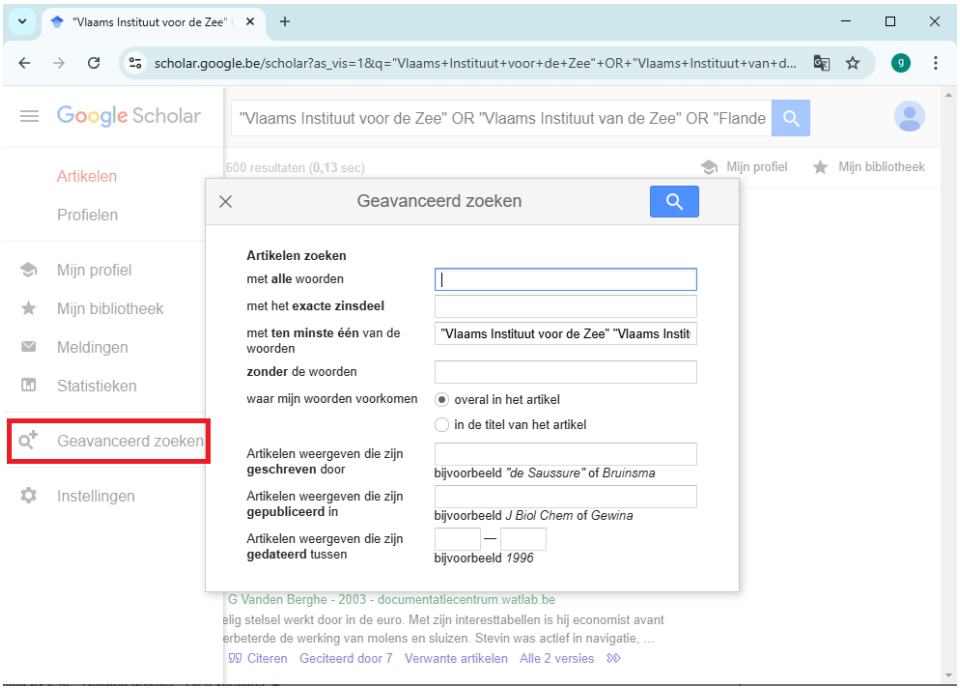
- **Zoek artikels met alle termen**: Combineert zoektermen. Zoekt publicaties die alle termen bevatten.
- **Zoek artikels met de exacte zoekterm**: Zoekt publicaties waar de zoekterm of zin exact in terug te vinden is.
- **Zoek artikels met op zijn minst 1 van de zoektermen**: Zoekt publicaties waar alle of minstens 1 van de zoektermen in voorkomen.
- **Zoek publicaties zonder de zoektermen**: Matcht publicaties waar geen enkele van de zoektermen in voorkomen.

Voor alle bovenstaande filters kan ingesteld worden of er enkel in de titel of overal in de tekst mag gezocht worden.
Daarnaast zijn er 3 bijkomende geävanceerde filters:

- **Zoek artikels op basis van auteurs**: Zoekt publicaties die geschreven zijn door een bepaalde auteur.
- **Zoek artikels op basis van de uitgever**: Zoekt publicaties die uitgegeven zijn door een bepaalde uitgever.
- **Zoek artikels op basis van publicatiedatum**: Zoekt publicaties die gepubliceerd zijn tussen 2 opgegeven datums.

Alle geävanceerde filters kunnen verder gespecifieerd worden door middel van logische operatoren: (AND, OR, NOT, AROUND).

- **AND**: Zoekt beide zoektermen in de publicatie.
- **OR**: Zoekt 1 of beide zoektermen in de publicatie.
- **NOT**: Sluit ongewenste tekst uit van het zoekresultaat.
- **AROUND**: Zoekt zoektermen in de ingestelde nabijheid van de opgegeven zoekterm.



Figuur 10.3: Google Scholar user interface voor het geävanceerd zoeken van publicaties op basis van de ingevoerde zoektermen en filters.

Alle geävanceerde filters kunnen verder gespecifieerd worden door middel van hulpwoorden:

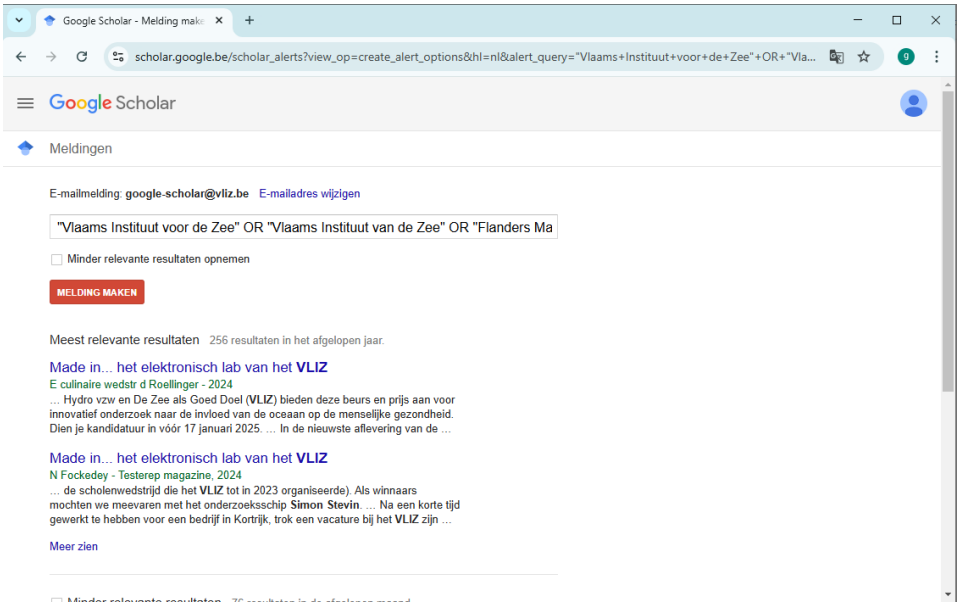
- **intitle**: De zoekresultaten bevatten de opgegeven zoekterm in de titel.
- **intext**: De zoekresultaten bevatten de opgegeven zoekterm in de tekst.
- **author**: De zoekresultaten bevatten de opgegeven auteur.
- **source**: De zoekresultaten bevatten de opgegeven uitgever.

Alle geävanceerde filters kunnen verder gespecifieerd worden door middel van enkele leestekens:

- **aanhalingstekens (“”)**: De zoekresultaten bevatten de exacte tekst tussen aanhalingstekens.
- **liggend streepje (A-B)**: Om aan te tonen dat 2 zoektermen sterk verbonden zijn.
- **liggend streepje (A -B)**: Om de tweede zoekterm uit te sluiten van de resultaten.

10.2. E-mail alerts

Het is mogelijk om de zoekresultaten te personaliseren. Voor elke zoekopdracht die wordt aangemaakt, kan een overeenkomstige alert ingesteld worden door te



Figuur 10.4: Google Scholar user interface voor het aanmaken van een e-mail alert voor de ingevoerde zoekopdracht.

klikken op 'Melding maken' zoals getoond op figuur 10.2. Het volstaat om het e-mailadres in te vullen naar waar de alerts verstuurd moeten worden. Dit genereert een verificatie e-mail en na bevestiging is de alert geäctiveerd.
Vanaf dan worden nieuwe publicaties die voldoen aan de filtercriteria systematisch doorgestuurd naar het e-mailadres. Overeenkomstig werd een nieuw account **google-scholar@marineinfo.org** aangemaakt, als een gedeeld account waar meerdere gebruikers toegang tot hebben. Vervolgens werd een Google account aangemaakt met hetzelfde e-mailadres.

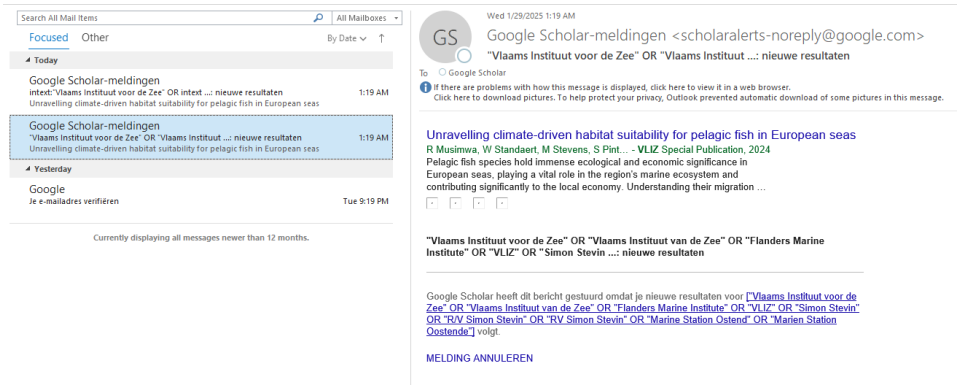
10.3. Opstellen van een zoekopdracht

Om relevante publicaties over het VLIZ te vinden, worden volgende zoektermen gebruikt:

- Vlaams Instituut voor de Zee
- Vlaams Instituut van de Zee
- Flanders Marine Institute
- VLIZ
- Simon Stevin
- R/V Simon Stevin
- RV Simon Stevin
- Marine Station Ostend
- Mariene Station Oostende

De zoekopdracht moet resultaten geven wanneer minstens 1 of meerdere zoektermen voorkomen in de titel of de tekst van het artikel. Dit kan bereikt worden door de zoektermen tussen aanhalingstekens te schrijven en door ze te verbinden met de OR operator.

```
``Vlaams Instituut voor de Zee''
                OR
``Vlaams Instituut van de Zee''
                OR
``Flanders Marine Institute''
                OR
``VLIZ''
                OR
``Simon Stevin''
                OR
``R/V Simon Stevin''
                OR
``RV Simon Stevin''
                OR
``Marine Station Ostend''
                OR
``Mariene Station Oostende''
```



Figuur 10.5: Google Scholar e-mail alert met de nieuwe resultaten sinds het aanmaken van de melding en sinds de vorige melding.

A

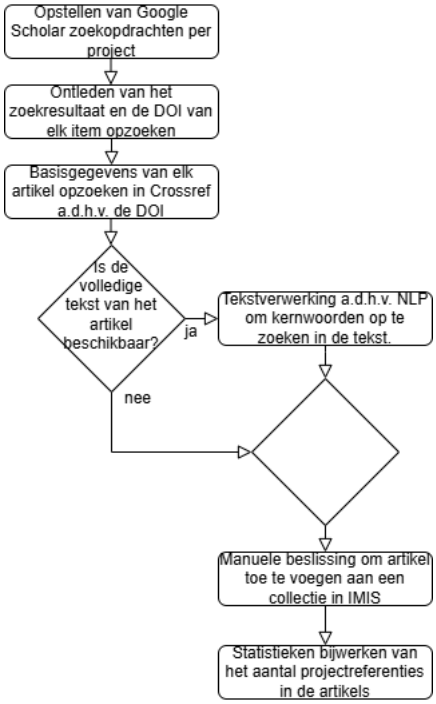
Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1. Inleiding

Het Vlaams Instituut voor de Zee (VLIZ) ("Vlaams Instituut voor de Zee", 2024) is een pionier in zeekennis. Dit wetenschappelijk instituut gelegen in Oostende heeft onder andere een mandaat om een complete en geactualiseerde catalogus bij te houden van alle wetenschappelijke publicaties in de mariene sector. Al meer dan 20 jaar bouwt het Integrated Marine Information System (IMIS) aan deze catalogus die intussen beschikt over meerdere collecties van marien wetenschappelijk referentiemateriaal.
Naast wetenschappelijke literatuur zitten er ook collecties van mariene wetenschappelijke projecten in het systeem. Het is belangrijk voor het VLIZ om te weten door hoeveel publicaties er naar een project (vb. het World Register of Marine Species (WoRMS) ("World Register of Marine Species", 2024)) verwezen wordt. Deze maatstaf geeft een indicatie van het draagvlak van elk project binnen de wetenschappelijke gemeenschap en is één van de belangrijkste criteria tijdens projectevaluaties. Daarom is het cruciaal om continu nieuwe publicaties waarin verwezen wordt naar die projecten op te zoeken. Daarvoor wordt op heden Google Scholar gebruikt die bekend staat als de meest uitgebreide en geactualiseerde index. Op die manier blijft IMIS up-to-date en zijn de projectreferenties steeds geactualiseerd.
Momenteel verloopt dit proces binnen het VLIZ grotendeels handmatig. De zoekresultaten, volgens een bepaalde zoekfilter per project, afkomstig van Google Scholar worden manueel gefilterd en de Digital Object Identifier (DOI) van de geselecteerde artikels wordt opgezocht. Vervolgens worden de basisgegevens van elk arti-

kel zoals titel, auteurs, datum en uitgever opgevraagd op basis van de DOI in Crossref ("Crossref", 2024). Met deze informatie wordt manueel beslist om het artikel al dan niet toe te voegen aan een collectie binnen IMIS.
Dit is een tijdrovend proces. Daarom is er vraag naar automatisatie die de zoekresultaten verwerkt en gestructureerd opslaat. De beslissing om een artikel toe te voegen aan IMIS blijft nog altijd een manuele stap, maar naar verwachting moet dit sneller, efficiënter en accurater verlopen. Dat moet mogelijk zijn doordat de heterogene zoekresultaten omgezet worden in gestructureerde informatie.



Figuur A.1: Chronologische oplijsting van de uit te voeren stappen

De centrale vraag die opgelost moet worden is: 'Hoe kunnen de uiteenlopende zoekresultaten van Google Scholar automatisch verwerkt worden om de DOI van elk item op te zoeken?'. Daarbij moet ook rekening gehouden worden met duplicaten, relevantie en diverse media types voor elk item van het zoekresultaat.
Om bovenstaande vraag te verduidelijken, moeten er een paar deelvragen gesteld worden:

- De zoekresultaten zijn afhankelijk van de zoekopdracht. In welke mate beïnvloeden kleine variaties van de zoekopdracht de resultaten?
- De zoekresultaten worden standaard gesorteerd op relevantie door Google Scholar, maar komt dat overeen met onze verwachtingen?
- De zoekopdracht kan het aantal resultaten beperken naar wens. Wat is de optimale threshold?

Om bovenstaande vraag op te lossen, moeten er een paar deelvragen beantwoord worden:

- Op welke manier kan de Google Scholar zoekopdracht op regelmatige basis automatisch uitgevoerd worden?
- Hoe worden de zoekresultaten van Google Scholar het best geparsed?
- Zoals gesteld zijn de zoekresultaten divers van aard. Kan het resultaat mits kwaliteitscontrole verbeterd (gefilterd) worden?
- De DOI zelf ontbreekt in het resultaat. Hoe zal de DOI voor elk item opgezocht worden?

Verder moet er gekeken worden welke databanktechnologie het meest geschikt is om de verwerkte informatie van elk item op te slaan. Tenslotte moet er ook nagedacht worden hoe het manuele proces met gebruik van de gestructureerde informatie het best softwarematig ondersteund kan worden. De volledige pijplijn zal geïmplementeerd worden als een Proof of Concept (PoC) die kan getest worden op basis van echte zoekresultaten. Als vervolg op dit onderzoek kan dan overgegaan worden tot het aanpassen en uitrollen van het systeem in functie van de feedback.

A.2. Literatuurstudie

IMIS (Haspeslagh & Vanden Berghe, 2024) heeft als doel het mariene onderzoek in Vlaanderen te coördineren. Die rol wordt vervuld door op te treden als centraal kenniscentrum voor en door de mariene sector in Vlaanderen. De taken van IMIS gaan verder dan louter het uitbouwen van collecties met publicaties, ze omvatten ook referenties naar wetenschappers en naar projecten. In deze bachelorproef zijn de publicaties en de wetenschappelijke projecten belangrijk.
Google Scholar is een zoekmachine die wetenschappelijke literatuur indexeert (“Google Scholar, Uit Wikipedia, de vrije encyclopedie”, 2024). Google Scholar laat toe om te gaan zoeken op sleutelwoorden, titel, auteur, domein en combinaties van deze (Noruzi, 2005). Op die manier zijn er enkel zoekresultaten die overeenkomen met onze zoekcriteria.
Er zijn verschillende kenmerken die bijdragen aan het succes van Google Scholar ten opzichte van de andere grote collecties. Google Scholar is beschikbaar zonder kosten, het bevat de grootste bibliografische collectie ter wereld, en de data is afkomstig van zowel publieke als niet publieke bronnen (Aguillo, 2011).
Google Scholar wordt meer en meer gebruikt als een bibliometrische tool om informatie te verzamelen over de impact van citaties van en naar afzonderlijke artikels (Moed e.a., 2016). Araújo e.a. (2021) legt uit hoe wetenschappelijke artikels andere publicaties citeren. Dat doen ze door middel van bibliografische referenties naar andere documenten doorheen de tekst, en een uitgewerkte lijst van referenties

aan het einde van het artikel. Een index van citaties zoals in Google Scholar, is een databank die deze referenties tussen documenten opslaat.
In dergelijk digitaal netwerk van referenties, moeten de afzonderlijke entiteiten blijvend, betrouwbaar en onderscheidbaar geïdentificeerd kunnen worden. Chandrakar (2006) legt uit hoe de DOI voldoet aan deze criteria en de standaard identificatie is geworden van intellectuele eigendom op het internet.
Aan de hand van de bekomen DOI op basis van de zoekresultaten kan alle informatie met betrekking tot het artikel opgevraagd worden. In sommige gevallen is zelfs een pdf-versie beschikbaar. Vanuit ons standpunt zijn pdf-versies van wetenschappelijke artikels een vorm van ongestructureerde informatie. Automatisch berekende voorstellingen van de inhoud kunnen bijdragen om het zoekproces te vereenvoudigen. Dat is ook het uitgangspunt van het werk van (Jehangir e.a., 2023). Natural language processing (NLP) is van groot belang voor het verwerken van deze ongestructureerde data. Met behulp van NLP kunnen er conclusies en samenvattingen gemaakt worden. Binnen het domein van NLP, staat Named Entity Recognition (NER) centraal voor het opstellen van structuur op basis van ongestructureerde tekst (Pakhale, 2023). Palshikar (z.d.) en Jehangir e.a. (2023) lijsten de verschillende NER technieken op die kunnen gebruikt worden. Koning e.a. (2005) tenslotte past NER toe om taxonomische entiteiten op te zoeken in een doorlopende tekst. Dit lijkt dusdanig ookt geschikt voor het opzoeken van projectentiteiten in een artikel. Schäfer en Kiefer (2011) merkt verder ook terecht op dat wetenschappers vandaag geconfronteerd worden met een dagelijkse vloedgolf van nieuwe publicaties. Daarenboven worden ook meer en meer oude artikels gedigitaliseerd. Zowel de verwerking van de platte tekst als de vloedgolf aan publicaties geven aanleiding tot het toepassen van NLP voor het beheren van wetenschappelijke artikels (Schäfer & Kiefer, 2011).

A.3. Methodologie

Ons onderzoek wordt opgesplitst in 7 afzonderlijke fases:

- opstellen van de meest accurate zoekopdrachten in Google Scholar
- parsen van de zoekresultaten om de DOI van elk artikel op te zoeken
- basisgegevens van elke publicatie opvragen in Crossref op basis van de DOI
- ontwikkelen van een tekstverwerkingsproces op basis van NER
- ontwerpen van de databank voor de opslag van de gestructureerde data
- installeren van een pijplijn die de afzonderlijke stappen combineert
- ontwerpen van een gebruikersinterface die de gestructureerde data toont

Achtereenvolgend wordt er nu dieper ingegaan op de verschillende fases. In de context van dit voorstel, wordt elke stap nog niet volledig uitgewerkt, maar is het de bedoeling om een richting aan te geven. Een uitvoerige beschrijving van alle taken zal aan bod komen in de bachelorproef.
De eerste fase gaat over het correct opstellen van de Google Scholar zoekopdrachten voor de verschillende projecten met als doel om als resultaat de meest relevante artikels te verkrijgen.
De tweede fase komt overeen met de hoofdopdracht van ons onderzoek, namelijk een systeem ontwikkelen dat de heterogene zoekresultaten kan parsen om de DOI op te zoeken. Meerdere zaken komen hier aan bod zoals ten eerste het aggregeren van zoekresultaten en groeperen per titel. Vervolgens het opzoeken van de DOI via de linked data, of via Crossref, of op de full-text versie van het artikel. Python lijkt de voor de hand liggende programmeertaal voor deze taak waarbij het draait om dataverwerking.
In de derde fase zal er met de API van Crossref gewerkt worden om de basisgegevens van een artikel op te vragen aan de hand van de DOI. Ook hier zal Python gebruikt worden.
De vierde fase onderzoekt of er met behulp van NER bruikbare informatie gefilterd kan worden uit de pdf-versie van een artikel. Op die manier kunnen de basisgegevens verrijkt worden om de manuele moderatie accurater te maken. Een zoektocht op het internet levert snel een achttal oplossingen op die ruwe tekst verwerken tot gestructureerde informatie:

- ScienceParse (“Science Parse parses scientific papers (in PDF form) and returns them in structured form.” 2024)
- ScienceParse API

  (“science-parse-api 1.0.1”, 2024)
- Grobid (“GROBID: a machine learning software for extracting information from scholarly documents”, 2024)
- Sci-pdf parser (“Python PDF parser for scientific publications: content and figures”, 2024)
- Paper AI (“Semantic search and workflows for medical/scientific papers”, 2024)
- Paper ETL (“ETL processes for medical and scientific papers”, 2024)
- Delph-in (“DELPH-IN Overview”, 2024)
- TaxonGrab (“TaxonGrab”, 2024)

Om onze specifieke business logica te implementeren, wordt gewerkt met één oplossing, of een combinatie van meerdere oplossingen, eventueel met gebruik van

een geschikte NER methode. Ook hier is Python de gekozen programmeertaal voor de dataverwerking.
De vijfde fase evalueert welke databanktechnologie het best geschikt is om onze gestructureerde data in op te slaan. Daar zijn bijvoorbeeld relationele, grafische en document store databanken.
De zesde fase focust zich op de automatisatie van het gehele proces: van zoekresultaat tot opgeslagen gestructureerde data. Er zal een pijplijn opgesteld worden die de voortgang van elke stap observeert en de status van het hele proces update-tet. Dit wordt gebouwd met Symfony rekening houdend met de technology stack binnen het VLIZ.
De zevende en laatste fase tenslotte moet ervoor zorgen dat alle voorgaande stappen ook opbrengen voor de eindgebruiker. De opgeslagen gestructureerde data wordt ontsloten via een API (API Platform) en een custom front-end (NextJS) applicatie zorgt ervoor dat deze overzichtelijk voorgesteld wordt.

A.4. Verwacht resultaat, conclusie

Naar verwachting zal deze pijplijn de verwerking van de Google Scholar zoekresutaten versnellen. Op die manier draagt ons onderzoek bij aan de continue uitbreiding van de collecties van IMIS. Er wordt ook verwacht dat het aantal projectreferenties beter gemonitord zal worden. Zo helpt ons onderzoek bij het beter in kaart van brengen van het draagvlak voor de wetenschappelijke projecten van het VLIZ.

Bibliografie

Aguillo, I. F. (2011). Is Google Scholar useful for bibliometrics? A webometric analysis. *Scientometrics*, *91*(2), 343–351. https://doi.org/10.1007/s11192-011-0582-8

Amin, M. M., Sutrisman, A., & Dwitayanti, Y. (2024). Google Scholar Crawling for Constructing Research Database. In *Proceedings of the 7th FIRST 2023 International Conference on Global Innovations (FIRST-ESCSI 2023)* (pp. 331–337). Atlantis Press International BV. https://doi.org/10.2991/978-94-6463-386-3_36

Anthropic: using the API: Errors. (2025, april 1). https://docs.anthropic.com/en/api/errors

Araújo, P. C. d., Gutierres, R. C., & Hjørland, B. (2021). Citation Indexing and Indexes. *KNOWLEDGE ORGANIZATION*, *48*(1), 72–101. https://doi.org/10.5771/0943-7444-2021-1-72

Atlas Vector Search Quick Start. (2025, mei 1). https://www.mongodb.com/docs/atlas/atlas-vector-search/tutorials/vector-search-quick-start/

Beautiful Soup. (2025, maart 1). https://www.crummy.com/software/BeautifulSoup/

Beel, J., & Gipp, B. (2009). Google Scholar's Ranking Algorithm: An Introductory Overview. *12th Int. Conf. Scientometrics Informetrics (ISSI)*, 439–446.

Bhatt, C., Bisht, A., Chauhan, R., Vishvakarma, A., Kumar, M., & Sharma, S. (2023). Web Scraping Techniques and Its Applications: A Review. *2023 3rd International Conference on Innovative Sustainable Computational Technologies (CISCT)*, 1–8. https://doi.org/10.1109/cisct57197.2023.10351298

Castrillo-Fernández, O. (2015, december 1). *Web Scraping: Applications and Tools* (tech. rap.). ePSIplatform Topic Report No. 2015 / 10 , December 2015.

Chandrakar, R. (2006). Digital object identifier system: an overview. *The Electronic Library*, *24*(4), 445–452. https://doi.org/10.1108/02640470610689151

Chroma. (2025, mei 1). https://docs.trychroma.com/docs/overview/getting-started

Click. (2025, maart 1). https://click.palletsprojects.com/en/stable/

Crossref. (2024, november 1). https://www.crossref.org/

Crossref API. (2025, april 1). https://github.com/fabiobatalha/crossrefapi

Crossref Commons for Python. (2025, april 1). https://gitlab.com/crossref/crossref_commons_py

DELPH-IN Overview. (2024, november 1). https://github.com/delph-in/docs/wiki

Dependency injection. (2025, maart 1). https://en.wikipedia.org/wiki/Dependency_injection

Dependency Injector. (2025, maart 1). https://python-dependency-injector.ets-labs.org/index.html

Document-oriented database. (2025, maart 1). https://en.wikipedia.org/wiki/Document-oriented_database

DOIs and matching regular expressions. (2025, maart 1). https://www.crossref.org/blog/dois-and-matching-regular-expressions/

email - An email and MIME handling package. (2025, maart 1). https://docs.python.org/3.13/library/email.html

ETL processes for medical and scientific papers. (2024, november 1). https://github.com/neuml/paperetl

Extracting Structured Data from Websites with Mirascope, Anthropic, and Streamlit. (2025, mei 1). https://ploomber.io/blog/mirascope-url-extractor/

Gemma3: Powerful AI Within Everyone's Reach. (2025, mei 1). https://gemma3.org/

Google Scholar Guide. (2025, maart 1). https://library.acg.edu/how-to-guides/google-scholar/overview

Google Scholar, Uit Wikipedia, de vrije encyclopedie. (2024, november 1). https://nl.wikipedia.org/wiki/Google_Scholar

Gray, J., Bounegru, L., & Chambers, L. (2012, augustus). *The Data Journalism Handbook*.

GROBID: a machine learning software for extracting information from scholarly documents. (2024, november 1). https://github.com/kermitt2/grobid

Habanero. (2025, april 1). https://github.com/sckott/habanero

Haspeslagh, J., & Vanden Berghe, E. (2024, november 1). *IMIS: Integrated Marine Information System*. Vlaams Instituut voor de Zee. https://www.researchgate.net/publication/33549102_IMIS_Integrated_Marine_Information_System

Hendricks, G., Tkaczyk, D., Lin, J., & Feeney, P. (2020). Crossref: The sustainable source of community-owned scholarly metadata. *Quantitative Science Studies*, *1*(1), 414–427. https://doi.org/10.1162/qss_a_00022

How to Use Chroma to Build Your First Similarity Search. (2025, mei 1). https://iamajithkumar.medium.com/how-to-use-chroma-to-build-your-first-similarity-search-5c054bfd5add

imaplib - IMAP4 protocol client. (2025, maart 1). https://docs.python.org/3/library/imaplib.html

Internet Message Access Protocol, Uit Wikipedia, de vrije encyclopedie. (2025, maart 1). https://en.wikipedia.org/wiki/Internet_Message_Access_Protocol

Introducing Meta Llama 3: The most capable openly available LLM to date. (2025, mei 1). https://ai.meta.com/blog/meta-llama-3/

Jehangir, B., Radhakrishnan, S., & Agarwal, R. (2023). A survey on Named Entity Recognition — datasets, tools, and methodologies. *Natural Language Processing Journal*, *3*, 100017. https://doi.org/10.1016/j.nlp.2023.100017

Koning, D., Sarkar, I. N., & Moritz, T. (2005). TaxonGrab: Extracting Taxonomic Names From Text. *Biodiversity Informatics*, *2*(0). https://doi.org/10.17161/bi.v2i0.17

Kumar, S., & Roy, U. B. (2023). A technique of data collection. In *Statistical Modeling in Machine Learning* (pp. 23–36). Elsevier. https://doi.org/10.1016/b978-0-323-91776-6.00011-7

LLM abstractions that aren't obstructions. (2025, mei 1). https://mirascope.com/

Lotfi, C., Srinivasan, S., Ertz, M., & Latrous, I. (2021). Web Scraping Techniques and Applications: A Literature Review. In *SCRS CONFERENCE PROCEEDINGS ON INTELLIGENT SYSTEMS* (pp. 381–394). Soft Computing Research Society. https://doi.org/10.52458/978-93-91842-08-6-38

Lourenço, J. R., Cabral, B., Carreiro, P., Vieira, M., & Bernardino, J. (2015). Choosing the right NoSQL database for the job: a quality attribute evaluation. *Journal of Big Data*, *2*(1). https://doi.org/10.1186/s40537-015-0025-0

Mitchell, R. (2015). *Web Scraping with Python: Collecting Data from the Modern Web* (1st). O'Reilly Media, Inc.

Moed, H. F., Bar-Ilan, J., & Halevi, G. (2016). A new methodology for comparing Google Scholar and Scopus. *Journal of Informetrics*, *10*(2), 533–551. https://doi.org/10.1016/j.joi.2016.04.017

MongoDB. (2025, maart 1). https://www.mongodb.com/resources/basics/databases/document-databases

Natural Language Processing of Google Scholar search results. (2025, mei 1). https://github.com/bart-de-paepe/google-scholar-natural-language-processing/blob/main/app/src/services/natural_language_processing_service.py

Noruzi, A. (2005). Google Scholar: The New Generation of Citation Indexes. *Libri*, *55*(4). https://doi.org/10.1515/libr.2005.170

Oh, K., & Colón-Aguirre, M. (2019). A Comparative Study of Perceptions and Use of Google Scholar and Academic Library Discovery Systems. *College amp; Research Libraries*, *80*(6), 876–891. https://doi.org/10.5860/crl.80.6.876

Ollama: Get up and running with large language models. (2025, mei 1). https://github.com/ollama/ollama

OpenAI developer platform. (2025, mei 1). https://platform.openai.com/docs/overview

Pakhale, K. (2023). Comprehensive Overview of Named Entity Recognition: Models, Domain-Specific Applications and Challenges. https://doi.org/10.48550/ARXIV.2309.14084

Palshikar, G. K. (z.d.). Techniques for Named Entity Recognition: A Survey. In *Bioinformatics* (pp. 400–426). IGI Global. https://doi.org/10.4018/978-1-4666-3604-0.ch022

PRATIBA, D., M.S., A., DUA, A., SHANBHAG, G. K., BHANDARI, N., & SINGH, U. (2018). Web Scraping And Data Acquisition Using Google Scholar. *2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS)*, 277–281. https://doi.org/10.1109/csitss.2018.8768777

PyMongo. (2025, maart 1). https://pymongo.readthedocs.io/en/stable/

PyMuPDF is a high-performance Python library for data extraction, analysis, conversion manipulation of PDF (and other) documents. (2025, maart 1). https://pymupdf.readthedocs.io/en/latest/how-to-open-a-file.html#opening-remote-files

Python PDF parser for scientific publications: content and figures. (2024, november 1). https://github.com/titipata/scipdf_parser

Rafsanjani, M. R. (2022). ScrapPaper: A web scrapping method to extract journal information from PubMed and Google Scholar search result using Python. https://doi.org/10.1101/2022.03.08.483427

Rahmatulloh, A., & Gunawan, R. (2020). Web Scraping with HTML DOM Method for Data Collection of Scientific Articles from Google Scholar. *Indonesian Journal of Information Systems*, *2*(2), 95–104. https://doi.org/10.24002/ijis.v2i2.3029

Requests is a simple, yet elegant, HTTP library. (2025, maart 1). https://pypi.org/project/requests/

Sataloff, R. T., Bush, M. L., Chandra, R., Chepeha, D., Rotenberg, B., Fisher, E. W., Goldenberg, D., Hanna, E. Y., Kerschner, J. E., Kraus, D. H., Krouse, J. H., Li, D., Link, M., Lustig, L. R., Selesnick, S. H., Sindwani, R., Smith, R. J., Tysome, J. R., Weber, P. C., & Welling, D. B. (2021). Systematic and other reviews: Criteria and complexities. *World Journal of Otorhinolaryngology - Head and Neck Surgery*, *7*(3), 236–239. https://doi.org/10.1016/j.wjorl.2021.04.007

Schäfer, U., & Kiefer, B. (2011). Advances in Deep Parsing of Scholarly Paper Content. In *Advanced Language Technologies for Digital Libraries* (pp. 135–153). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-23160-5_9

Science Parse parses scientific papers (in PDF form) and returns them in structured form. (2024, november 1). https://github.com/allenai/science-parse

science-parse-api 1.0.1. (2024, november 1). https://pypi.org/project/science-parse-api

Scrape Google and other search engines from our fast, easy, and complete API. (2025, mei 1). https://serpapi.com/google-scholar-api

Scrapegraph AI Tutorial; Scrape websites easily with LLaMA AI. (2025, mei 1). https://www.scrapingbee.com/blog/scrapegraph-ai-tutorial-scrape-websites-easily-with-llama-ai/

ScrapeGraphAI: You Only Scrape Once. (2025, mei 1). https://github.com/ScrapeGraphAI/Scrapegraph-ai

Selenium with Python. (2025, maart 1). https://selenium-python.readthedocs.io/

Semantic search and workflows for medical/scientific papers. (2024, november 1). https://github.com/neuml/paperai

Semantic search using Chroma. (2025, mei 1). https://github.com/bart-de-paepe/google-scholar-semantic-chroma

Semantic search using MongoDB. (2025, mei 1). https://github.com/bart-de-paepe/google-scholar-semantic-mongodb

Singrodia, V., Mitra, A., & Paul, S. (2019). A Review on Web Scrapping and its Applications. *2019 International Conference on Computer Communication and Informatics (ICCCI)*. https://doi.org/10.1109/iccci.2019.8821809

Status of Python versions. (2025, maart 1). https://devguide.python.org/versions/

Sulistya, Y. I., Wardhana, A. C., Istighosah, M., & Riyandi, A. (2024). Automated Google Scholar Crawling with a Web-Based Tool for Publication Data Management. *Jurnal Teknologi Dan Sistem Informasi Bisnis*, *6*(4), 768–773. https://doi.org/10.47233/jteksis.v6i4.1604

TaxonGrab. (2024, november 1). https://sourceforge.net/projects/taxongrab/

Tips for using the Crossref REST API. (2025, mei 1). https://www.crossref.org/documentation/retrieve-metadata/rest-api/tips-for-using-the-crossref-rest-api/

Using the API Getting started. (2025, mei 1). https://docs.anthropic.com/en/api/getting-started

Vector Search. (2025, mei 1). https://www.mongodb.com/products/platform/atlas-vector-search

Vlaams Instituut voor de Zee. (2024, november 1). https://www.vliz.be

Web scraping experiment with AI (Parsing HTML with GPT-4 and GPT-4o). (2025, mei 1). https://serpapi.com/blog/web-scraping-and-parsing-experiment-with-ai-openai/

Web scraping Google Scholar SERP using Anthropic. (2025, mei 1). https://github.com/bart-de-paepe/google-scholar-anthropic/blob/main/app/src/services/parse_service.py

Web scraping Google Scholar SERP using Beautiful Soup. (2025, mei 1). https://github.com/bart-de-paepe/google-scholar-beautifulsoup/blob/main/app/src/services/parse_service.py

Web scraping Google Scholar SERP using Llama3. (2025, mei 1). https://github.com/bart-de-paepe/google-scholar-ollama/blob/main/app/src/services/parse_service.py

Web scraping Google Scholar SERP using OpenAI. (2025, mei 1). https://github.com/bart-de-paepe/google-scholar-openai/blob/main/app/src/services/parse_service.py

World Register of Marine Species. (2024, november 1). https://www.marinespecies.org

Yang, D., Zhang, A. N., & Yan, W. (2017). Performing literature review using text mining, Part I: Retrieving technology infrastructure using Google Scholar and APIs. *2017 IEEE International Conference on Big Data (Big Data)*, 3290–3296. https://doi.org/10.1109/bigdata.2017.8258313