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

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Samenvatting

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Lijst van codefragmenten

1

Inleiding

De inleiding moet de lezer net genoeg informatie verschaffen om het onderwerp te begrijpen en in te zien waarom de onderzoeksvraag de moeite waard is om te onderzoeken. In de inleiding ga je literatuurverwijzingen beperken, zodat de tekst vlot leesbaar blijft. Je kan de inleiding verder onderverdelen in secties als dit de tekst verduidelijkt. Zaken die aan bod kunnen komen in de inleiding (Pollefliet, 2011):

- context, achtergrond
- afbakenen van het onderwerp
- verantwoording van het onderwerp, methodologie
- probleemstelling
- onderzoeksdoelstelling
- onderzoeksvraag
- ...

1.1. Probleemstelling

Uit je probleemstelling moet duidelijk zijn dat je onderzoek een meerwaarde heeft voor een concrete doelgroep. De doelgroep moet goed gedefinieerd en afgelijnd zijn. Doelgroepen als “bedrijven,” “KMO’s”, systeembeheerders, enz. zijn nog te vaag. Als je een lijstje kan maken van de personen/organisaties die een meerwaarde zullen vinden in deze bachelorproef (dit is eigenlijk je steekproefkader), dan is dat een indicatie dat de doelgroep goed gedefinieerd is. Dit kan een enkel bedrijf zijn of zelfs één persoon (je co-promotor/opdrachtgever).

1.2. Onderzoeksvraag

Wees zo concreet mogelijk bij het formuleren van je onderzoeksvraag. Een onderzoeksvraag is trouwens iets waar nog niemand op dit moment een antwoord heeft (voor zover je kan nagaan). Het opzoeken van bestaande informatie (bv. "welke tools bestaan er voor deze toepassing?") is dus geen onderzoeksvraag. Je kan de onderzoeksvraag verder specifiëren in deelvragen. Bv. als je onderzoek gaat over performantiemetingen, dan

1.3. Onderzoeksdoelstelling

Wat is het beoogde resultaat van je bachelorproef? Wat zijn de criteria voor succes? Beschrijf die zo concreet mogelijk. Gaat het bv. om een proof-of-concept, een prototype, een verslag met aanbevelingen, een vergelijkende studie, enz.

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.
In Hoofdstuk 6, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2

Stand van zaken

Het huidige digitale tijdperk dat hoofdzakelijk gekenmerkt wordt door de toenemende belangstelling in AI, levert eveneens een gigantische hoeveelheid aan online data. Met deze grote berg van informatie die beschikbaar is op de pagina's van het world wide web, is het bijzonder relevant voor organisaties om te begrijpen hoe ze deze data kunnnen 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 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 zo uitvoerig web scrapers te bespreken, 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 hangbare 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 citaties en h-indexen [2]. Volgens Oh en Colón-Aguirre (2019) is er een symbiose tus-

[1] alle clients zien steeds dezelfde data

[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.

sen GS and de academische wereld waarbij academici gratis hun werk kunnen aabieden op GS. GS op zijn beurt zorgt voor goede visibiliteit van dat werk door onder andere citaten, referenties en 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.

Dit hoofdstuk bevat je literatuurstudie. De inhoud gaat verder op de inleiding, maar zal het onderwerp van de bachelorproef *diepgaand* uitspitten. De bedoeling is dat de lezer na lezing van dit hoofdstuk helemaal op de hoogte is van de huidige stand van zaken (state-of-the-art) in het onderzoeksdomein. Iemand die niet vertrouwd is met het onderwerp, weet nu voldoende om de rest van het verhaal te kunnen volgen, zonder dat die er nog andere informatie moet over opzoeken (Pollefliet, 2011).
Je verwijst bij elke bewering die je doet, vakterm die je introduceert, enz. naar je bronnen. In LaTeX kan dat met het commando `\textcite{}` of `\autocite{}`. Als argument van het commando geef je de "sleutel" van een "record" in een bibliografische databank in het BibLaTeX-formaat (een tekstbestand). Als je expliciet naar de auteur verwijst in de zin (narratieve referentie), gebruik je `\textcite{}`. Soms is de auteursnaam niet expliciet een onderdeel van de zin, dan gebruik je `\autocite{}` (referentie tussen haakjes). Dit gebruik je bv. bij een citaat, of om in het bijschrift van een overgenomen afbeelding, broncode, tabel, enz. te verwijzen naar de bron. In de volgende paragraaf een voorbeeld van elk.
Knuth (1998) schreef een van de standaardwerken over sorteer- en zoekalgoritmen. Experten zijn het erover eens dat cloud computing een interessante opportuniteit vormen, zowel voor gebruikers als voor dienstverleners op vlak van informatietech-

Figuur 2.1: Voorbeeld van invoegen van een figuur. Zorg altijd voor een uitgebreid bijschrift dat de figuur volledig beschrijft zonder in de tekst te moeten gaan zoeken. Vergeet ook je bronvermelding niet!

```
    import pandas as pd
    import seaborn as sns

    penguins = sns.load_dataset('penguins')
    sns.relplot(data=penguins, x="flipper_length_mm",
        y="bill_length_mm", hue="species")
```

Codefragment 2.1: Voorbeeld van het invoegen van een codefragment.

nologie (Creeger, 2009).
Let er ook op: het `cite`-commando voor de punt, dus binnen de zin. Je verwijst meteen naar een bron in de eerste zin die erop gebaseerd is, dus niet pas op het einde van een paragraaf.

Kolom 1	**Kolom 2**	**Kolom 3**
α	β	γ
A	10.230	a
B	45.678	b
C	99.987	c

Tabel 2.1: Voorbeeld van een tabel.

3

Methodologie

3.1. Google Scholar zoekopdracht

GS biedt een aantal filtermogelijkheden om de meest relevante resultaten te bekomen voor de zoekopdracht. De filter moet manueel geconfigureerd worden voor elke zoekopdracht en vervolgens kan er een e-mail alert ingesteld worden per zoekopdracht.

3.2. Parsen van Google Scholar zoekresultaat

Er moet een geautomatiseerd systeem komen dat

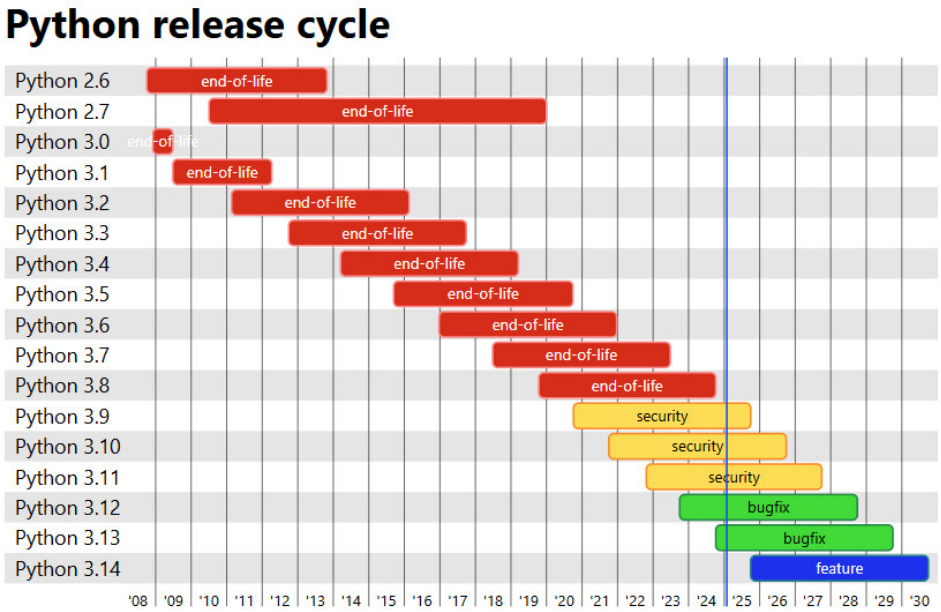
- de binnenkomende e-mail alerts in een mailbox leest
- vervolgens de zoekresultaten in de body parset
- tenslotte voor elk zoekresultaat de overeenkomstige link bezoekt teneinde de Digital Object Indentifier (DOI) op te zoeken.

Daarvoor moet er custom code geschreven worden en bijhorend moet de vraag gesteld worden welke technology stack er gebruikt zal worden? Daarbij zijn 2 aspecten belangrijk:

- Wat is de bestaande technology stack van de klant en kan de te ontwikkelen software daarin ondergebracht worden?
- Welke technologie is het meest geschikt om het gestelde probleem op te lossen?

Voor het eerste punt zijn er bij de klant 2 pijlers:

- Alle websites en datasystemen zijn geschreven in Php, al dan niet gebruik makend van het Symfony framework. Voor hun data steunen ze voornamelijk op



Figuur 3.1: Python release cycle.("Status of Python versions", 2025)

SqlServer en op PostgreSQL. Voor hun user interface steunen ze hoofdzakelijk op Twig en op NextJS.

- Veel data processing scripts draaien in hun eigen docker container. Ze gebruiken vooral R en Python als programmeertaal.

Onze opdracht past het best binnen de tweede pijler. Het is niet opportuun om een webserver te belasten met het systematisch herhalen van een opdracht die los staat van de websites die hij host. De GS alerts zijn op zich ook data die in hun eigen container verwerkt zullen worden.
Voor het tweede punt is de technologie zeer duidelijk. Python (maar ook R) beschikken over hele grote bibliotheken die toelaten om uiteenlopende soorten data te verwerken.
Het programmeerwerk zal uitgevoerd worden in Python. Op het moment van schrijven is Python versie 3.13.1 de meest recente stabiele versie.

Release	Released	Active Support	Security Support	Latest
3.13	3 months and 4 weeks ago (07 Oct 2024)	Ends in 1 year and 8 months (01 Oct 2026)	Ends in 4 years and 9 months (31 Oct 2029)	3.13.1 (03 Dec 2024)
3.12	1 year and 4 months ago (02 Oct 2023)	Ends in 1 month and 4 weeks (02 Apr 2025)	Ends in 3 years and 9 months (31 Oct 2028)	3.12.8 (03 Dec 2024)
3.11	2 years and 3 months ago (24 Oct 2022)	Ended 10 months ago (01 Apr 2024)	Ends in 2 years and 9 months (31 Oct 2027)	3.11.11 (03 Dec 2024)
3.10	3 years and 4 months ago (04 Oct 2021)	Ended 1 year and 10 months ago (05 Apr 2023)	Ends in 1 year and 9 months (31 Oct 2026)	3.10.16 (03 Dec 2024)
3.9	4 years ago (05 Oct 2020)	Ended 2 years and 8 months ago (17 May 2022)	Ends in 9 months (31 Oct 2025)	3.9.21 (03 Dec 2024)
~~3.8~~	5 years ago (14 Oct 2019)	Ended 3 years and 9 months ago (03 May 2021)	Ended 3 months and 4 weeks ago (07 Oct 2024)	~~3.8.20 (06 Sep 2024)~~

Figuur 3.2: End of life date Python.(“End of life date Python”, 2025)

De geparsete e-mail alerts worden opgeslagen in een MongoDB database, dit is een NoSQL document store databank. Een RabbitMQ broker zorgt ervoor dat voor elke e-mail iedere stap in de flow doorlopen wordt. Tenslotte wordt alles ondergebracht in zijn eigen Docker container die binnen het project met elkaar communiceren.

4

Google Scholar zoekopdracht

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

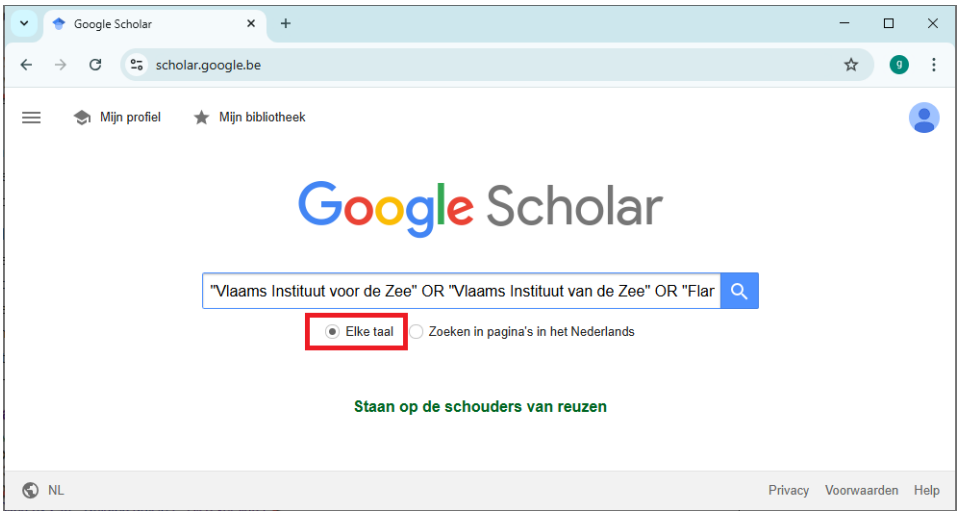
4.1.1. Basis zoeken

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:

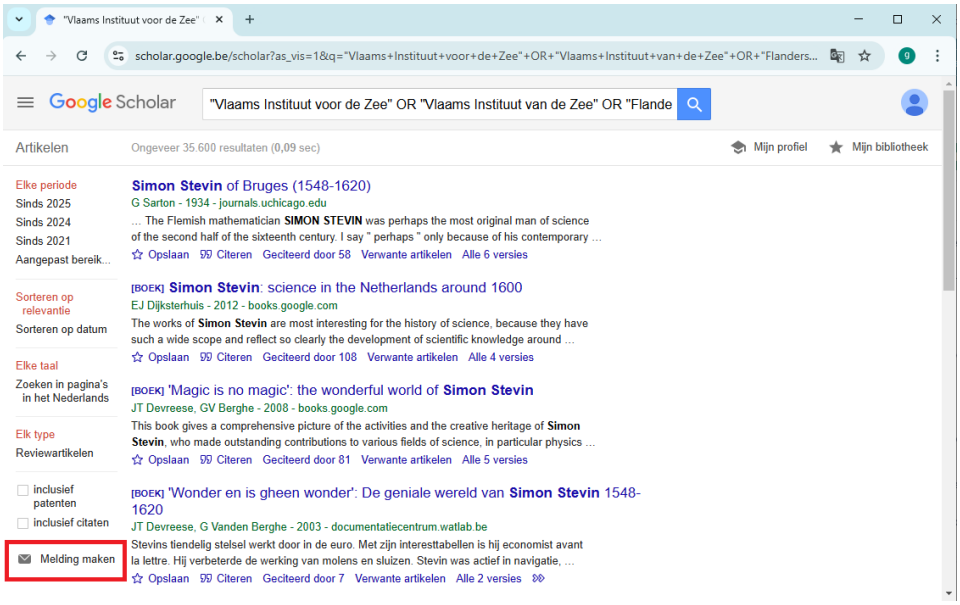
- **Elke periode**: Dit is de default filter zodat alle resultaten getoond worden ongeacht hun publicatiedatum.
- **Sinds jaar**: Hierbij worden enkel resultaten gefilterd die sinds het gespecifiëerde jaar gepubliceerd werden.
- **Aangepast bereik**: Hierbij worden enkel resultaten gefilterd waarvan 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 4.1: Google Scholar user interface voor het basis zoeken van publicaties op basis van de ingevoerde zoektermen.



Figuur 4.2: 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.

4.1.2. Geävanceerd zoeken

Google Scholar heeft ook een meerdere geävanceerde zoekopties:

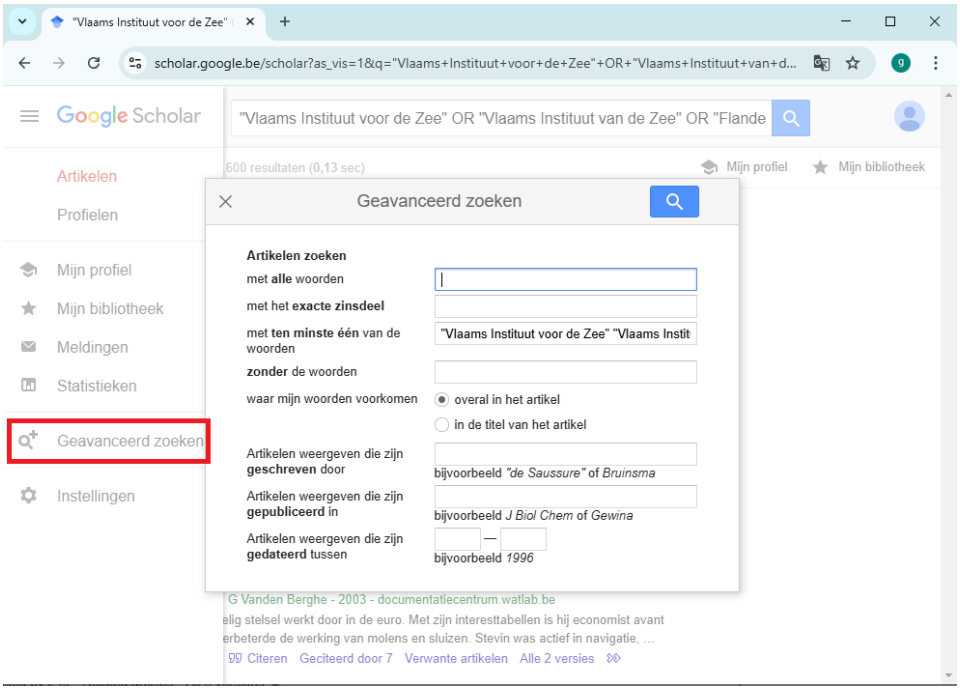
- **Zoek artikels met alle termen**: Combineert zoektermen. Zoekt publicaties die alle termen bevatten.
- **Zoek artikels met de exacte zoekterm**: Zoekt publicaties waar de zoekterm of zin exact in terug te vinden is.
- **Zoek artikels met op zijn minst 1 van de zoektermen**: Zoekt publicaties waar 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 4.3: 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:

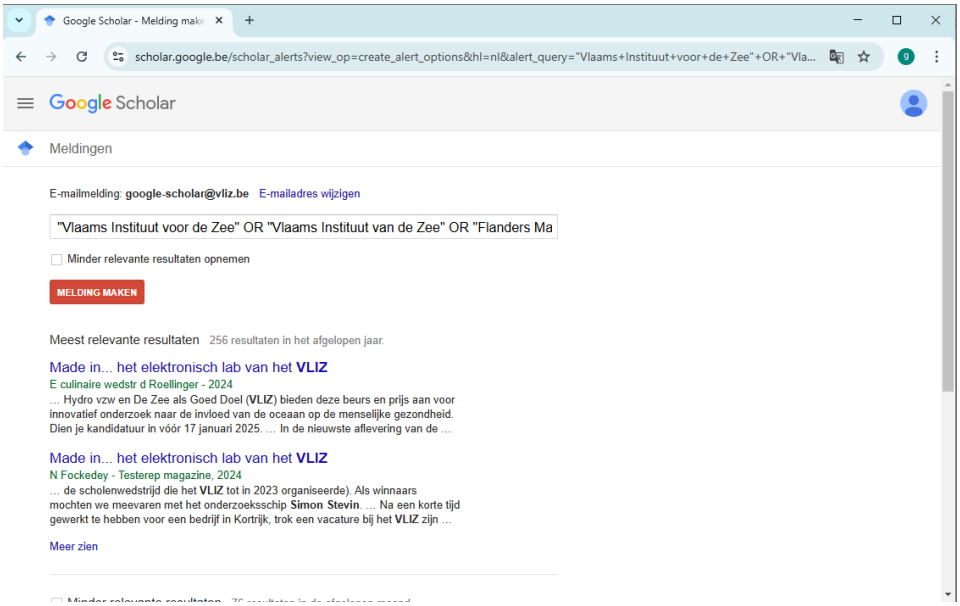
- **intitle**: De zoekresultaten bevatten de opgegeven zoekterm in de titel.
- **intext**: De zoekresultaten bevatten de opgegeven zoekterm in de tekst.
- **author**: De zoekresultaten bevatten de opgegeven auteur.
- **source**: De zoekresultaten bevatten de opgegeven uitgever.

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.

4.2. 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 4.4: Google Scholar user interface voor het aanmaken van een e-mail alert voor de ingevoerde zoekopdracht.

klikken op 'Melding maken' zoals getoond op figuur 4.2. 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.

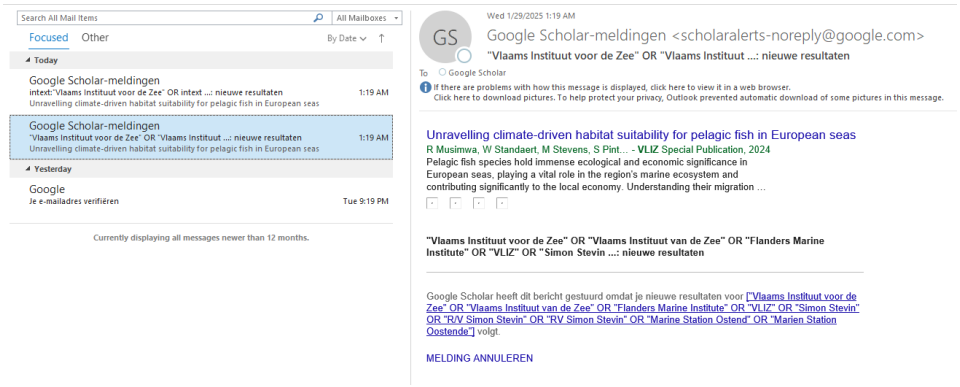
4.3. Opstellen van een zoekopdracht

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

5

Parsen van Google Scholar zoekresultaten

5.1. Doelstelling

De GS alert geeft aanleiding tot e-mails met de meest recente zoekresultaten. Voor elke e-mail en voor elk zoekresultaat wordt de DOI opgezocht. Daarvoor worden 3 stappen doorlopen:

- E-mail lezen
- Body van de e-mail parsen
- DOI opzoeken in de link die aanwezig is in het geparsete resultaat

5.1.1. E-mail lezen

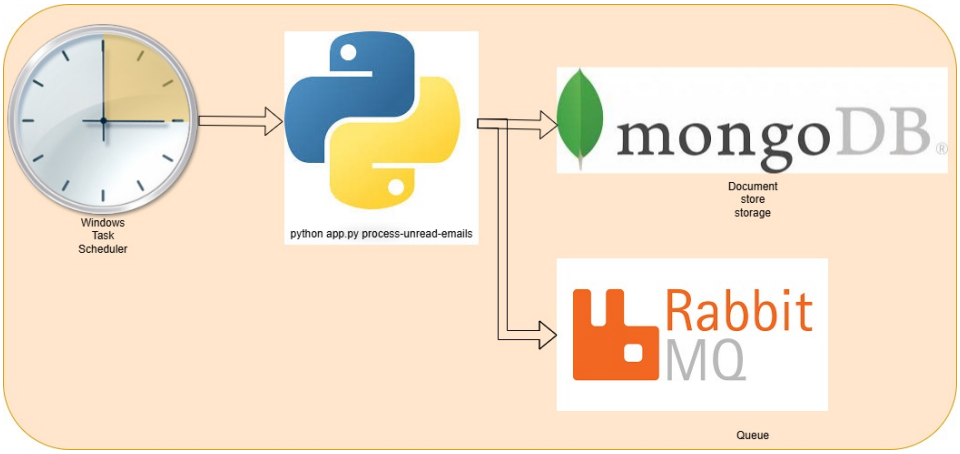
Op gezette tijdstippen wordt de inbox van het account google-scholar@marineinfo.org geopend op zoek naar ongelezen e-mails. Elke e-mail wordt gelezen en de informatie van de e-mail wordt opgeslaan. De e-mail wordt verplaatst van de inbox naar een werkmap overeenkomstig het onderwerp van de e-mail. Het onderwerp komt overeen met de zoekopdracht.

5.1.2. Body van de e-mail parsen

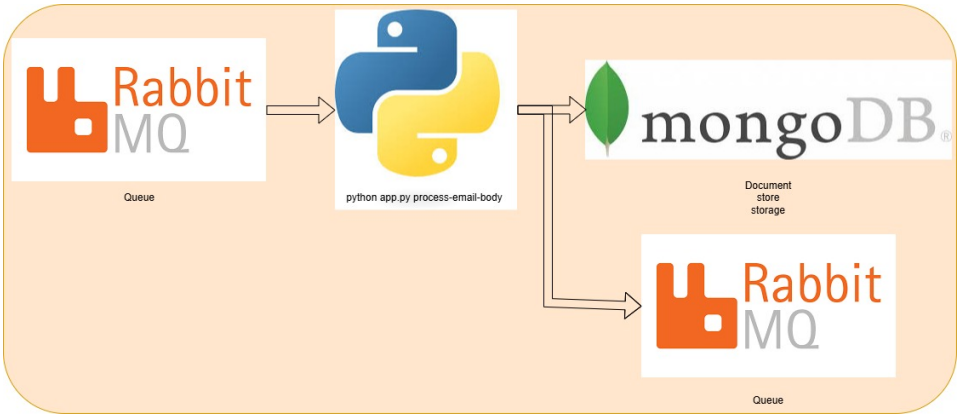
Elke e-mail bevat 1 of meerdere zoekresultaten afkomstig van GS. In deze stap wordt elk zoekresultaat uit de body van de e-mail gefilterd en afzonderlijk opgeslaan. De informatie van het zoekresultaat wordt opgeslaan.

5.1.3. DOI opzoeken in de link van het geparsete resultaat

Het GS zoekresultaat zelf bevat geen DOI. Maar elk zoekresultaat heeft wel een weblink die werd opgeslaan in de vorige stap. Hier wordt een request gestuurd

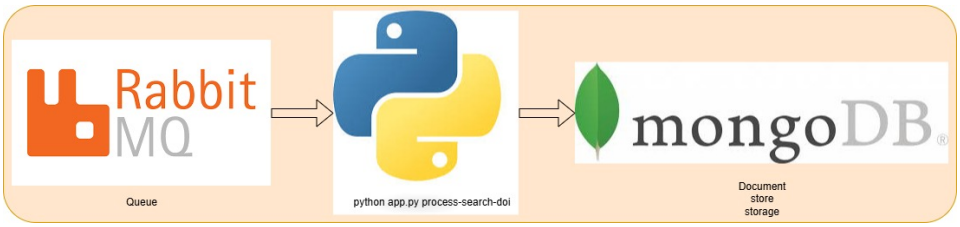


Figuur 5.1: Flow e-mail lezen.



Figuur 5.2: Flow e-mail body parsen.

naar de link. Het response wordt vervolgens verwerkt met als doel om het DOI erin op te zoeken. De informatie van het DOI wordt opgeslaan.



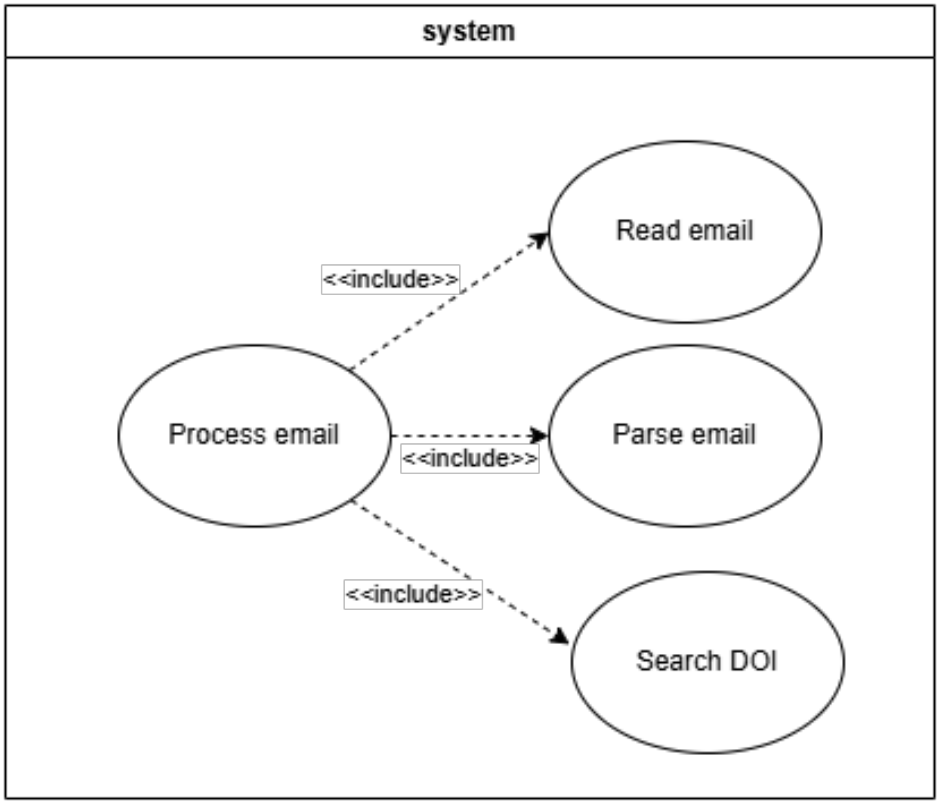
Figuur 5.3: DOI opzoeken in de link van het geparsete resultaat.

5.2. Analyse

5.2.1. Use Cases

Use Case Read email	GS stuurt e-mails naar google-scholar@marineinfo.org. Deze e-mails moeten automatisch verwerkt worden.
primary actor	-
stakeholders	-
pre-condities	onverwerkte e-mail
post-condities	verwerkte e-mails
normaal verloop	1. systeem opent e-mail 2. systeem leest afzender 3. systeem evalueert afzender (DR-email) 4. systeem leest datum / tijdstip 5. systeem leest onderwerp 6. systeem leest body 7. systeem slaat deze gegevens op 8. systeem verplaatst e-mail van inbox naar mail-box folder op basis van onderwerp
alternatief verloop	3A. de afzender is verkeerd (DR-email) 3A1. systeem verplaatst e-mail van inbox naar SPAM folder
domeinregels	DR-email: afzender moet zijn: scholaralerts-noreply@google.com

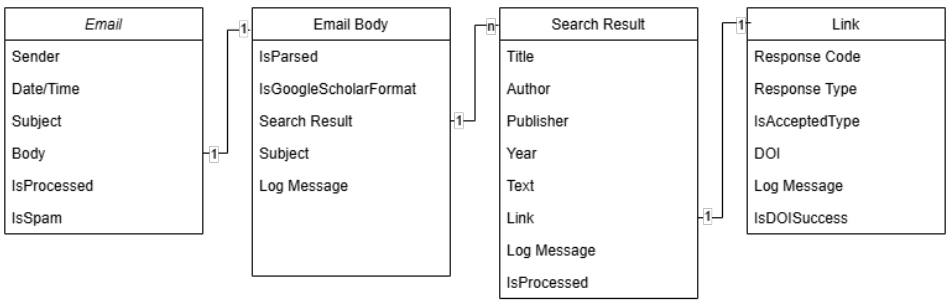
Tabel 5.1: GS stuurt e-mails naar google-scholar@marineinfo.org. Deze e-mails moeten automatisch verwerkt worden.



Figuur 5.4: Use case diagram.

5.2.2. Domeinmodel

De 3 bovenstaande stappen geven aanleiding tot 4 objecten die de kern van het domeinmodel uitmaken en de nodige properties bevatten om de tussenresultaten bij te houden alsook de uiteindelijke DOI.



Figuur 5.5: Domeinmodel.

Use Case Parse email	De body van een e-mail wordt ontleed en de informatie wordt gestructureerd opgeslaan.
primary actor	-
stakeholders	-
pre-condities	onverwerkte body van e-mail
post-condities	verwerkte body van e-mail
normaal verloop	1. systeem ontvangt de body van een e-mail 2. systeem ontleedt de body en zoekt volgende onderdelen (DR-Google-Scholar formaat) a. lijst i. titel (DR-uniek) ii. auteur iii. uitgever iv. jaartal v. tekst (DR-uniek) vi. link (DR-uniek) b. onderwerp 3. systeem slaat bovenstaande informatie gestructureerd op
alternatief verloop	2A. de verwachte informatie wordt niet gevonden (DR-Google-Scholar formaat) 2A1. systeem logt het probleem 2A2. use case eindigt zonder bereiken van de post-conditie 2B. de body is onleesbaar 2B1. systeem logt het probleem 2B2. use case eindigt zonder bereiken van de post-conditie 2C. titel en/of tekst en/of link zijn niet uniek (zijn reeds eerder opgeslaan) 2C1. systeem logt het voorval 2C2. systeem gaat verder met stap 2 van het normale verloop
domeinregels DR-uniek	DR-Google-Scholar-formaat

Tabel 5.2: De body van een e-mail wordt ontleed en de informatie wordt gestructureerd opgeslaan.

Use Case Search DOI	Zoek een DOI in een tekstueel document.
primary actor	-
stakeholders	-
pre-condities	systeem beschikt over een online-url
post-condities	het DOI is gevonden
normaal verloop	1. systeem stuurt een request naar de link 2. systeem ontvangt response 3. systeem bepaalt type van de response 4. systeem systeem kiest de juiste tools in functie van het type (DR-type) 5. systeem zoekt DOI in de response 6. systeem slaat het gevonden DOI op
alternatief verloop	2A. geen response 2A1. het systeem logt het probleem 2A2. use case eindigt zonder bereiken van de post-conditie 4A. type is niet ondersteund (DR-type) 4A1. het systeem logt het probleem 4A2. use case eindigt zonder bereiken van de post-conditie 5A. DOI wordt niet gevonden 5A1. het systeem logt het probleem 5A2. use case eindigt zonder bereiken van de post-conditie 5B. meerdere DOIs worden gevonden 5A1. het systeem logt het probleem 5A2. use case eindigt zonder bereiken van de post-conditie
domeinregels	DR-type: pdf, html

Tabel 5.3: Zoek een DOI in een tekstueel document.

5.3. Implementatie

5.3.1. E-mail lezen

Meerdere specifieke Python libraries ondersteunen alle mogelijke bewerkingen met e-mail:

- **imaplib - IMAP4 protocol client ("imaplib - IMAP4 protocol client", 2025)**: Legt een verbinding met een IMAP server en implementeert de functionaliteiten van het IMAP [1] protocol om e-mails op te halen.("Internet Message Access Protocol, Uit Wikipedia, de vrije encyclopedie", 2025)
- **email - An email and MIME handling package ("email - An email and MIME handling package", 2025)**: Laat toe om emails te verwerken adhv. 3 modules.
  - **message**: Alle bewerkingen met een e-mail.
  - **parser**: Om een e-mail om te zetten in tekst.
  - **generator**: Om tekst om te zetten in een e-mail.

De gelezen e-mail wordt opgeslaan in een document store database ("Document-oriented database", 2025). Dit type databank is geschikt voor het bijhouden van dynamische data die als geheel (als één document) opgeslaan wordt. MongoDB ("MongoDB", 2025) is de meest gekende technologie van dit type databank. Pymongo ("PyMongo", 2025) is de specifieke python library die de verbinding met de Mongodb server mogelijk maakt.
Het custom script dat de e-mail leest wordt opgeroepen door de Windows task scheduler. Click_ ("Click„■, 2025) laat toe om meerdere commando's te definiëren in hetzelfde script.
Tenslotte wordt er gebruik gemaakt van de Dependency Injector ("Dependency Injector", 2025) om al deze services door middel van dependency injection ("Dependency injection", 2025) ter beschikking te stellen aan het script.
Aan het einde van het process dat de e-mail leest, wordt de body van de e-mail in een wachtrij geplaatst zodat deze opgepikt wordt door de volgende stap.

5.3.2. Body van de e-mail parsen

De body van de e-mail komt overeen met de Search Engine Result Page (SERP) en 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 5.6.

[1] Internet Message Access Protocol

Figuur 5.6: HTML structuur van de GS alert.

Meerdere specifieke libraries laten toe om HTML en XML 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 gevonden informatie wordt opgeslaan in een document store database. Aan het einde van het proces dat de body parset, wordt de link van elk zoekresultaat in een wachtrij geplaatst zodat deze kan opgepikt worden door de volgende stap.

5.3.3. DOI opzoeken in de link van het geparsete resultaat

Met de link die werd opgeslaan in de vorige stap wordt er nu een request gestuurd. Daar zijn meerdere methodes mogelijk maar Requests “Requests is a simple, yet elegant, HTTP library.” (2025) is de meest courante. De headers van het request worden zo opgesteld dat de server van GS niet ziet dat het hier om een geäutomatiseerde request gaat. Daardoor bestaat het risico dat GS request van deze client bant.

De url die in het zoekresultaat van GS zit, verwijst zelf ook naar de GS website: Daarin valt te zien dat de eigenlijke url van het zoekresultaat verpakt zit binnen het GS adres. Inderdaad, het response heeft een HTTP 302 status code en redirect het de request naar de eigenlijke url. Het verwerkende script kan de uiteindelijke url opzoeken in de javascript van het response aan de hand van de **location.replace** instructie. Met de url uit de **location.replace** kan er nu opnieuw een request gestuurd worden. Het response dat daar teruggestuurd wordt is verschillend voor

headers = ”User-Agent”: ”Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.94 Safari/537.36”, ”Accept”: ”text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8”, ”Accept-Encoding”: ”gzip, deflate, br”, ”Accept-Language”: ”en-US,en;q=0.9,lt;q=0.8,et;q=0.7,de;q=0.6”,

Codefragment 5.1: Voorbeeld headers.

https://scholar.google.be/scholar_url?url=https://www.nature.com/articles/s41598-024-83657-0&hl=nl&sa=X&d=11469596634523662 58&ei=tRGsZ4eaLIC96rQP29mI6AY&scisig=AFWwakY6yEQWAvLJNk68&html=&pos=0&folt=kw-top

Codefragment 5.2: Embedded url.

```
        pattern = r”location\.replace\(['\”]([^'\”]+)['\”]\)”
        match = re.search(pattern, js_code)
        if match:
        location_replace_url = match.group(1)
        self.logging_service.logger.debug(f”Extracted
            location.replace URL for search result link:
            {location_replace_url}”)
        else:
        link.log_message = ”No location.replace url found for search
            result link”
        self.logging_service.logger.debug(”No location.replace url
            found for search result link”)
```

Codefragment 5.3: Codefragment voor het vinden van de location.replace.

```
            doc = pymupdf.Document(stream=pdf)
            # Extract all Document Text
            text = chr(12).join([page.get_text() for page in doc])
```

Codefragment 5.4: Codefragment voor het openen van een online pdf.

elke zoekresultaat. Het is dus niet mogelijk om het DOI voor elk zoekresultaat op dezelfde manier te achterhalen. Daarom wordt er een stapsgewijze werkwijze gehanteerd, van zodra het DOI gevonden is worden de volgende stappen overgeslaan:

- **het DOI opzoeken in de link**: De link zelf bevat soms zelf reeds het DOI. Dit wordt getoets aan de hand van enkele reguliere expressies.
- **het DOI opzoeken in de content** Het antwoord komt overeen met een html-pagina of een pdf-document. Andere return types worden buiten beschouwing gelaten. De inhoud van het antwoord wordt gelezen en met gebruik van dezelfde reguliere expressie kan de aanwezigheid van het DOI opgespoord worden. Voor pdf-documenten is er een extra tussenstap nodig: de inhoud moet gelezen worden met gebruik van ("PyMuPDF is a high-performance Python library for data extraction, analysis, conversion manipulation of PDF (and other) documents." 2025).
- **het DOI opzoeken in de embedded content**: Als tot zover er nog steeds geen DOI gevonden is, dan kan het zijn dat het response een embedded document bevat. Op een website kan embedded content ontsloten worden door erop te klikken. Dat gaat hier niet. Door middel van ("Selenium with Python", 2025) kan de embedded content automatisch gedownload worden. Eenmaal gedownload kan het bestand gewoon geopend worden en doorzocht worden naar een DOI op dezelfde manier als voor pdf-documenten in de voorgaande stap.

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

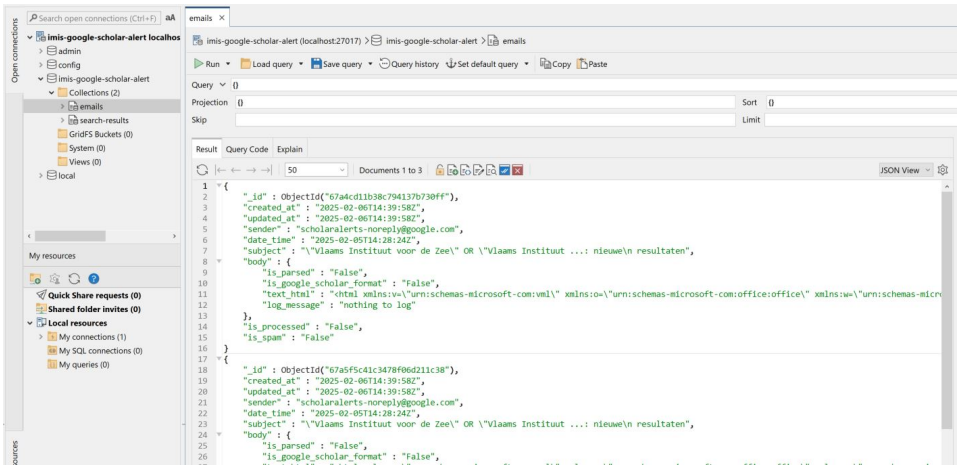
- **/**^{1}0.4, 9/[−.;()/ : A − Z0 − 9]+**/i**
- **/**^{1}0.1002/[]+**/i**
- **/**^{1}0.4/ + − +X?(+) + < []+ : []* > +. + .+;**/i**
- **/**^{1}0.1021/ + +**/i**
- **/**^{1}0.1207/[] + & + +**/i**

Het DOI wordt opgeslaan in de databank.

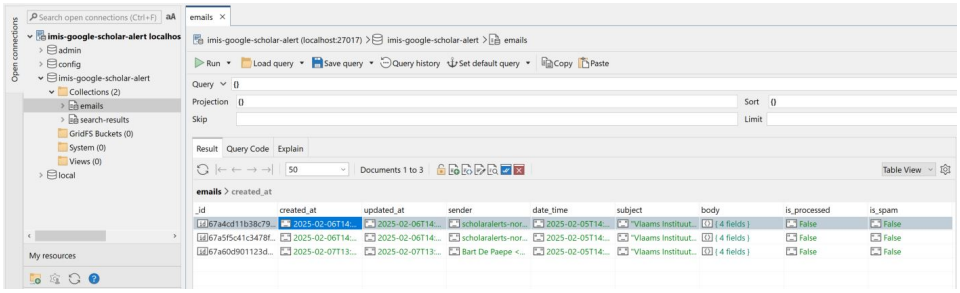
5.4. Resultaat

5.4.1. E-mail lezen

Het resultaat van deze stap is eerder een tussenresultaat waarbij verschillende onderdelen van de e-mail zoals datum, onderwerp, en andere opgeslaan worden in de databank. Deze opgeslaan informatie vormt de bron voor de verdere verwerking in stap 2.

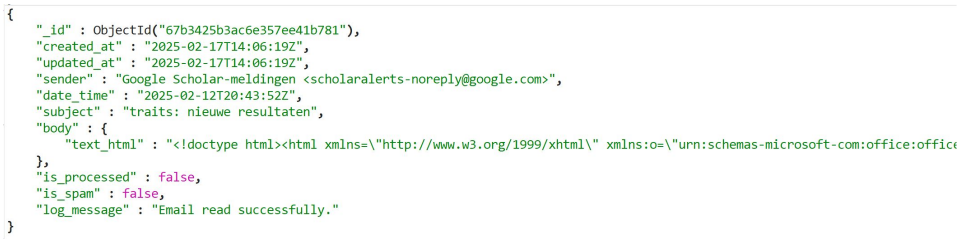


Figuur 5.7: Databank JSON view.



Figuur 5.8: Databank table view.

Een overzicht van de gegevens die opgeslaan worden zijn te zien in figuur 5.9



Figuur 5.9: Databank structuur e-mail.

- **_id**: De unieke ID van de record.
- **created_at**: De datum en het tijdstip waarop de e-mail in de databank werd opgeslaan.
- **updated_at**: De datum en het tijdstip waarop de e-mail in de databank het laatst werd gewijzigd.
- **sender**: De afzender van de e-mail.
- **date_time**: De datum en het tijdstip waarop de e-mail verstuurd werd.
- **subject**: Het onderwerp van de e-mail.
- **body**:
  - **text_html**: De body van de e-mail.
- **is_processed**: Een vlag die aanduidt of de body van de e-mail verwerkt werd.
- **is_spam**: Een vlag die aanduidt of de e-mail niet van GS afkomstig is.
- **log_message**: De logberichten tijdens het lezen van de e-mail worden hier bewaard.

5.4.2. Body van de e-mail parsen

Het resultaat van deze stap is eerder een tussenresultaat waarbij voor alle zoekresultaten in de body van de e-mail, verschillende onderdelen zoals titel, snippet, auteurs, uitgever, jaartal, en link opgeslaan worden in de databank. Deze opgeslaan informatie vormt de bron voor de verdere verwerking in stap 3. De gegevens van de e-mail worden bijgewerkt om aan te duiden dat de body verwerkt is, zoals te zien is in figuur 5.10.


```
{
    "_id" : ObjectId("67b5cf80ba5d6fa0d55d98eb"),
    "created_at" : "2025-02-19T12:33:04Z",
    "updated_at" : "2025-02-19T12:33:44Z",
    "sender" : "Google Scholar-meldingen <scholaralerts-noreply@google.com>",
    "date_time" : "2025-02-11T19:12:53Z",
    "subject" : "traits: nieuwe resultaten",
    "body" : {
        "text_html" : "<!doctype html><html xmlns=\"http://www.w3.org/1999/xhtml\" xmlns:o=\"urn:schemas-microsoft-
        "is_parsed" : true,
        "is_google_scholar_format" : true,
        "log_message" : "Body successfully parsed. "
    },
    "is_processed" : true,
    "is_spam" : false,
    "log_message" : "Email read successfully."
}
```


Figuur 5.10: Databank structuur body.

- **_id**: De unieke ID van de record.
- **created_at**: De datum en het tijdstip waarop de e-mail in de databank werd opgeslaan.

- **updated_at**: De datum en het tijdstip waarop de e-mail in de databank het laatst werd gewijzigd.
- **sender**: De afzender van de e-mail.
- **date_time**: De datum en het tijdstip waarop de e-mail verstuurd werd.
- **subject**: Het onderwerp van de e-mail.
- **body**:
  - **text_html**: De body van de e-mail.
  - **is_parsed**: Een vlag die aanduidt of het parsen van de body succesvol was.
  - **is_google_scholar_format**: Een vlag die aanduidt of de body overeenkomt met de verwachte html structuur.
  - **log_message**: De logberichten tijdens het parsen van de body worden hier bewaard.
- **is_processed**: Een vlag die aanduidt of de body van de e-mail verwerkt werd.
- **is_spam**: Een vlag die aanduidt of de e-mail niet van GS afkomstig is.
- **log_message**: De logberichten tijdens het lezen van de e-mail worden hier bewaard.

Een overzicht van de gegevens die opgeslaan worden zijn te zien in figuur 5.11.


```
{
    "_id" : ObjectId("67b5cfa84c19ccd3d1df01d0"),
    "created_at" : "2025-02-19T12:33:44Z",
    "updated_at" : "2025-02-19T12:33:44Z",
    "email" : ObjectId("67b5cf80ba5d6fa0d55d98eb"),
    "title" : "Genetic Variability and Association of Morpho-Agronomic Traits Among Ethiopian Barley (Hordeum vu
    "author" : "A Zewodu, W Mohammed, E Shiferaw ",
    "publisher" : " Scientifica",
    "year" : " 2025",
    "text" : "… traits among 49 barley accessions. The experiment was conducted in 2021 using \r\na 7 × 7 simple
    "link" : {
        "url" : "https://scholar.google.be/scholar_url?url=https://onlinelibrary.wiley.com/doi/pdf/10.1155/sci5/
    },
    "log_message" : "Search result parsed successfully.",
    "is_processed" : false
}
```


Figuur 5.11: Databank structuur zoekresultaat.

- **_id**: De unieke ID van de record.
- **created_at**: De datum en het tijdstip waarop het zoekresultaat in de databank werd opgeslaan.
- **updated_at**: De datum en het tijdstip waarop het zoekresultaat in de databank het laatst werd gewijzigd.
- **email**: Referentie naar de overeenkomstige e-mail waar dit zoekresultaat in staat.

- **title**: Titel van het zoekresultaat.
- **author**: Auteur(s) van het zoekresultaat.
- **publisher**: Uitgever van het zoekresultaat.
- **year**: Jaar van het zoekresultaat.
- **text**: Snippet van het zoekresultaat.
- **link**:
  - **url**: De link naar de online bron van het zoekresultaat.
- **log_message**: De logberichten tijdens het verwerken van het zoekresultaat worden hier bewaard.
- **is_processed**: Een vlag die aanduidt of de online bron van het zoekresultaat verwerkt werd.

5.4.3. DOI opzoeken in de link van het geparsete resultaat

Het DOI, het resultaat van deze stap, is een unieke identifier voor elk artikel. Vanaf dit punt en voor alle volgende bewerkingen kan elk artikel ondubbelzinnig geïdentificeerd worden en kunnen de bijhorende gegevens op gezocht worden. Een overzicht van de gegevens die opgeslaan worden is te zien in figuur 5.12.


```
{
    "_id" : ObjectId("67b8621c2adedfb8d1920e85"),
    "created_at" : "2025-02-21T11:23:08Z",
    "updated_at" : "2025-02-21T11:23:31Z",
    "email" : ObjectId("67b85f7e139ca121611fcf11"),
    "title" : "Natural selection and adaptive traits in the Maniq, a nomadic hunter-gatherer society from Mainland Southeast Asia",
    "author" : "T Herzog, M Larena, W Kutanan, H Lukas, M Fieder… ",
    "publisher" : " Scientific Reports",
    "year" : " 2025",
    "text" : "Asia is home to diverse hunter-gatherer populations characterized by significant \r\nmorphological, anthropological,
    "link" : {
        "url" : "https://scholar.google.be/scholar_url?url=https://www.nature.com/articles/s41598-024-83657-0&hl=nl&sa=X&d=1146959
        "response_code" : NumberInt(200),
        "response_type" : "text/html",
        "is_accepted_type" : true,
        "DOI" : "10.1002/humu.22595;",
        "log_message" : "DOI successfully retrieved",
        "is_DOI_success" : true
    },
    "log_message" : "Search result parsed successfully.",
    "is_processed" : true
}
```


Figuur 5.12: Databank structuur link.

- **_id**: De unieke ID van de record.
- **created_at**: De datum en het tijdstip waarop het zoekresultaat in de databank werd opgeslaan.
- **updated_at**: De datum en het tijdstip waarop het zoekresultaat in de databank het laatst werd gewijzigd.
- **email**: Referentie naar de overeenkomstige e-mail waar dit zoekresultaat in staat.

- **title**: Titel van het zoekresultaat.
- **author**: Auteur(s) van het zoekresultaat.
- **publisher**: Uitgever van het zoekresultaat.
- **year**: Jaar van het zoekresultaat.
- **text**: Snippet van het zoekresultaat.
- **link**:
  - **url**: De link naar de online bron van het zoekresultaat.
  - **location_replace_url**: De link naar de eigenlijke online bron van het zoekresultaat na redirect door GS.
  - **response_code**: De http status code van het response.
  - **response_type**: Het type document van het response.
  - **is_accepted_type**: Een vlag die aanduidt of het type van het response verder beschouwd zal worden.
  - **DOI**: De Digital Object Identifier.
  - **log_message**: De logberichten tijdens het verwerken van de link worden hier bewaard.
  - **is_DOI_succes**: Een vlag die aanduidt of de verwerking tijdens deze stap al dan niet succesvol was.
- **media_type**: Het type van de online bron zoals het aangegeven wordt in het zoekresultaat.
- **log_message**: De logberichten tijdens het verwerken van het zoekresultaat worden hier bewaard.
- **is_processed**: Een vlag die aanduidt of de online bron van het zoekresultaat verwerkt werd.

6

Conclusie

Curabitur nunc magna, posuere eget, venenatis eu, vehicula ac, velit. Aenean ornare, massa a accumsan pulvinar, quam lorem laoreet purus, eu sodales magna risus molestie lorem. Nunc erat velit, hendrerit quis, malesuada ut, aliquam vitae, wisi. Sed posuere. Suspendisse ipsum arcu, scelerisque nec, aliquam eu, molestie tincidunt, justo. Phasellus iaculis. Sed posuere lorem non ipsum. Pellentesque dapibus. Suspendisse quam libero, laoreet a, tincidunt eget, consequat at, est. Nullam ut lectus non enim consequat facilisis. Mauris leo. Quisque pede ligula, auctor vel, pellentesque vel, posuere id, turpis. Cras ipsum sem, cursus et, facilisis ut, tempus euismod, quam. Suspendisse tristique dolor eu orci. Mauris mattis. Aenean semper. Vivamus tortor magna, facilisis id, varius mattis, hendrerit in, justo. Integer purus.

Vivamus adipiscing. Curabitur imperdiet tempus turpis. Vivamus sapien dolor, congue venenatis, euismod eget, porta rhoncus, magna. Proin condimentum pretium enim. Fusce fringilla, libero et venenatis facilisis, eros enim cursus arcu, vitae facilisis odio augue vitae orci. Aliquam varius nibh ut odio. Sed condimentum condimentum nunc. Pellentesque eget massa. Pellentesque quis mauris. Donec ut ligula ac pede pulvinar lobortis. Pellentesque euismod. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent elit. Ut laoreet ornare est. Phasellus gravida vulputate nulla. Donec sit amet arcu ut sem tempor malesuada. Praesent hendrerit augue in urna. Proin enim ante, ornare vel, consequat ut, blandit in, justo. Donec felis elit, dignissim sed, sagittis ut, ullamcorper a, nulla. Aenean pharetra vulputate odio.

Quisque enim. Proin velit neque, tristique eu, eleifend eget, vestibulum nec, lacus. Vivamus odio. Duis odio urna, vehicula in, elementum aliquam, aliquet laoreet, tellus. Sed velit. Sed vel mi ac elit aliquet interdum. Etiam sapien neque, convallis et, aliquet vel, auctor non, arcu. Aliquam suscipit aliquam lectus. Proin tincidunt magna sed wisi. Integer blandit lacus ut lorem. Sed luctus justo sed enim.

Morbi malesuada hendrerit dui. Nunc mauris leo, dapibus sit amet, vestibulum et, commodo id, est. Pellentesque purus. Pellentesque tristique, nunc ac pulvinar adipiscing, justo eros consequat lectus, sit amet posuere lectus neque vel augue. Cras consectetuer libero ac eros. Ut eget massa. Fusce sit amet enim eleifend sem dictum auctor. In eget risus luctus wisi convallis pulvinar. Vivamus sapien risus, tempor in, viverra in, aliquet pellentesque, eros. Aliquam euismod libero a sem. Nunc velit augue, scelerisque dignissim, lobortis et, aliquam in, risus. In eu eros. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur vulputate elit viverra augue. Mauris fringilla, tortor sit amet malesuada mollis, sapien mi dapibus odio, ac imperdiet ligula enim eget nisl. Quisque vitae pede a pede aliquet suscipit. Phasellus tellus pede, viverra vestibulum, gravida id, laoreet in, justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer commodo luctus lectus. Mauris justo. Duis varius eros. Sed quam. Cras lacus eros, rutrum eget, varius quis, convallis iaculis, velit. Mauris imperdiet, metus at tristique venenatis, purus neque pellentesque mauris, a ultrices elit lacus nec tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent malesuada. Nam lacus lectus, auctor sit amet, malesuada vel, elementum eget, metus. Duis neque pede, facilisis eget, egestas elementum, nonummy id, neque.

A

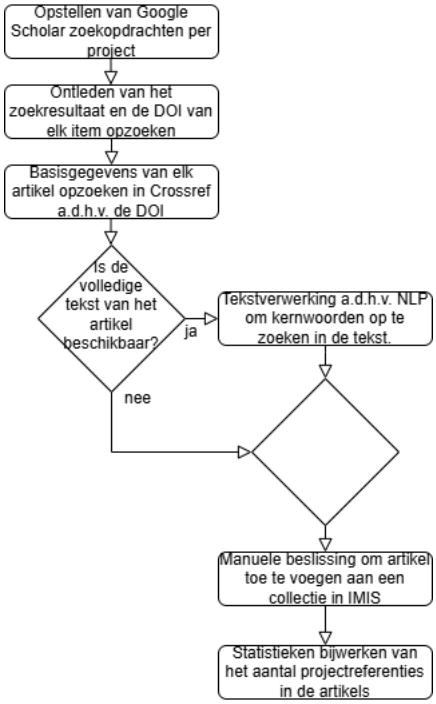
Onderzoeksvoorstel

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. 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

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

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

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

Creeger, M. (2009). CTO Roundtable: Cloud Computing. *Communications of the ACM*, *52*(8), 50–56.

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

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

End of life date Python. (2025, maart 1). https://endoflife.date/python

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

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

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

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

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

Knuth, D. E. (1998). *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. Addison Wesley Longman Publishing Co., Inc.

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

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

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

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

Pollefliet, L. (2011). *Schrijven van verslag tot eindwerk: do's en don'ts*. Academia Press.

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

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

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/

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

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