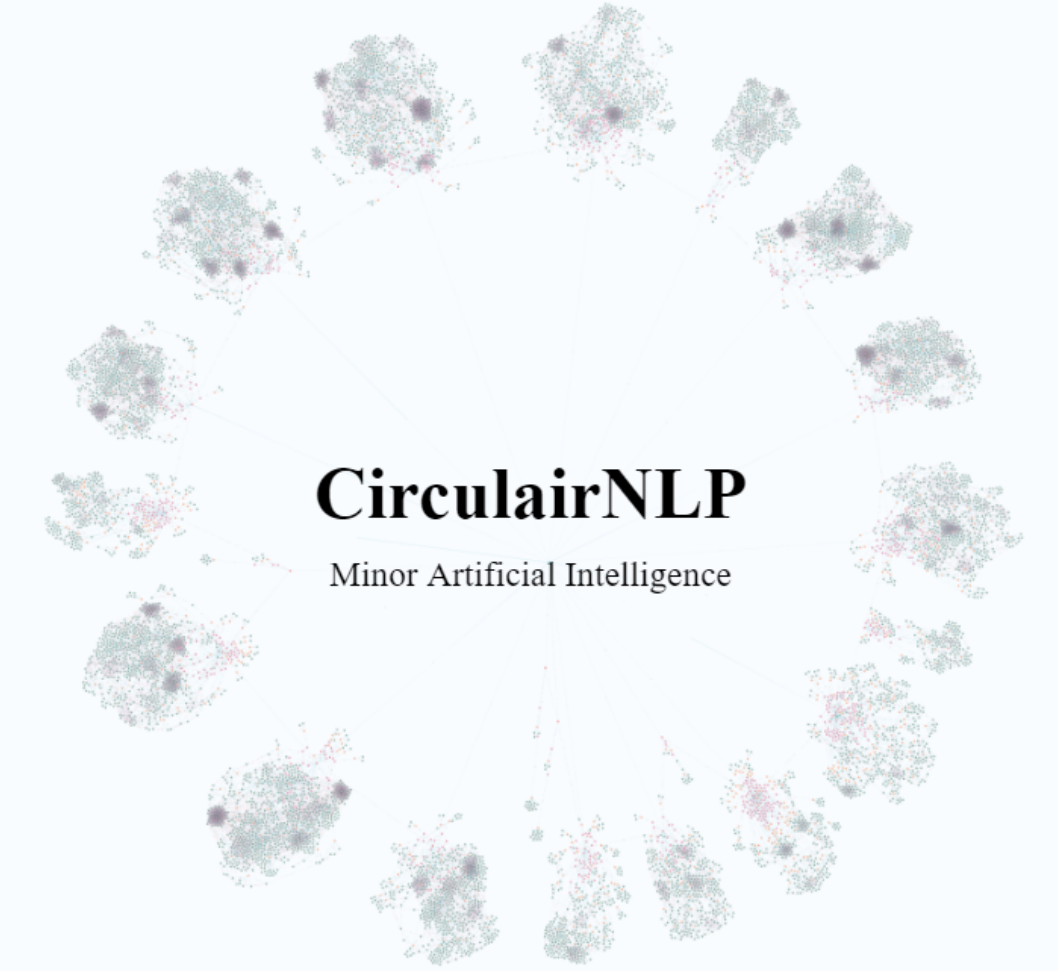


CirculairNLP

Minor Artificial Intelligence

by

Axel Frederiks (2202630),
Jonah Siemers (2202907),
Menno Rompelberg (2204333),
Oscar Theelen (2204154)

19/04/2024

lectoraat Data Intelligence

supervisor(s)

Pascal Slaats
Jeroen Baijens
Shannen Dolls

Zuyd Hogeschool

Abstract

Bedrijven worden nu verplicht om jaarlijks hun impact op circulariteit in het milieu te documenteren. Dit gebeurt meestal in de vorm van een PDF en om deze documenten efficiënt te analyseren, is een opdracht geformuleerd: "Ontwikkel een programma dat zoveel mogelijk informatie uit een PDF kan halen en op basis daarvan kan bepalen of de auteur streeft naar circulariteit." Door dit te realiseren worden bedrijven die zich inzetten voor het milieu beschermt en bedrijven die dit niet doen blootgesteld.
Middels de cycli binnen de Design Science Research methode van Hevner, is het fundament voor deze opdracht gelegd. Onderzoek laat zien dat PDF's lezen zelfs voor de grote bedrijven in deze sector nog steeds een probleem is. Het, in dit document beschreven, artefact is de eerste tool die een hiërarchische structuur uit een PDF correct kan opstellen. Deze data wordt daarna opgeslagen in een Neo4j database.
Opvolgende projecten zullen op basis van deze nieuwe technologie verder streven naar het bereiken van een betrouwbaar eindproduct.

1

Introductie

Aanleiding

In de huidige wereld ligt het onderwerp milieugebonden circulariteit steeds meer in het centrum van de aandacht. Aangezien bedrijven door deze aandacht aan circulariteit hier ook mee te maken krijgen, moeten zij rapportages opstellen waarin ze vertellen over hun activiteiten op dit gebied. Velen doen dit goed, echter zijn er een aantal die minder aandacht schenken aan de modulariteit en duidelijkheid van hun documenten. Dit veroorzaakt het probleem dat de documenten van deze bedrijven niet makkelijk op circulariteit te beoordelen zijn door een computer. Voor een mens is het zeer lastig en tijdrovend om honderden of wel duizenden van dit soort documenten, die soms wel eens oplopen tot wel 150 pagina's, te beoordelen.
Op weg naar een betere circulariteit in deze wereld zou het grootschalig en contextueel analyseren van deze documenten door computers een stap zijn in de beoogde richting. Dit project zorgt dat bedrijven die zich inzetten voor het milieu beschermt worden en bedrijven die dit niet doen inhoudelijk gecontroleerd. Vanuit Pascal Slaats en Jeroen Baijens is daarom een verzoek gedaan om hier een oplossing voor te vinden en te maken.

Doelstelling

De doelstelling van dit project is om de eerste stap te zetten in de richting van het interpreteren van PDF bestanden, door PDF's gestructureerd op te stellen in een database en daarop bewerkingen uit te voeren om koppelingen te leggen binnen tekst, zodat er een fundament wordt gelegd voor verdere analyse.

[1] Zie J - Requirements in de bijlagen

2

Theoretisch kader

Waarom is het lezen van PDF's zo moeilijk voor een computer?

Om de lay-out van het bestand te behouden, slaat PDF gegevens op exacte posities op en soms zelfs niet als tekst, maar als afbeelding. Op een blog van Gradient worden onder andere de volgende uitdagingen en problemen genoemd: "Inconsistent Layout, Lack of Structural Metadata, Complex Layouts and Formats, Quality Issues, Multimodal Content" [9]. Ook wordt in de documentatie van de Apache PDF tool Tika gezegd: "Note that as of 2023, we still see papers at research conferences on extracting structural elements (including tables) from PDFs – THIS IS NOT A SOLVED PROBLEM. And, humorously, this kind of task is sometimes called "PDF remediation"." [3] Al deze uitdagingen zorgen ervoor dat pdf extractie heel lastig is.

Wat is OCR?

Optical Character Recognition, oftewel OCR, is een vorm van Artificial Intelligence waarbij een getrainde agent karakters herkent in een afbeelding. OCR systemen zijn relatief effectief in het herkennen van tekst, maar kunnen moeite hebben met diepteverschillen tussen oppervlakten[2].

Wat is een graph database?

Een graaf database heeft veel weg van de veelgebruikte relationele database. Echter zijn er duidelijke verschillen tussen de twee. In een relationele database dienen relaties tussen gegevens te worden gedefinieerd door middel van sleutels(beschrijvingen van verbindingen). Daarentegen worden er in een graaf database relaties gelegd tussen gegevens door een directe verbinding, de database is dus echt te beschouwen als een graaf met nodes en randen.[15]

Wat is coreferencing?

Coreferentie verwijst naar het fenomeen waarbij een persoonlijk voornaamwoord (zoals "hij", "zij", "het", etc.) in een zin terugverwijst naar een eerdere entiteit (bijvoorbeeld een persoon of een object) in dezelfde tekst. Om dit soort relaties tussen entiteiten en de bijbehorende persoonlijke voornaamwoorden te begrijpen, wordt coreference-resolution toegepast, dit is een vorm van Natural Language Processing (NLP). Coreference-resolution algoritmen proberen clusters van woorden te identificeren die naar dezelfde entiteit verwijzen.[11]

Lemmatisatie en stemming

Stemming en lemmatisatie zijn beide een techniek die wordt gebruikt om woorden te normaliseren. Stemming vereenvoudigt een woordvariant door affixen te verwijderen, "rennen" wordt "ren". Lemmatisatie doet ditzelfde maar herkent ook dat "ren" een daadwerklijk woord is. Het kan dus zijn dat stemming niet-bestaande woorden creëert.

[2] Zie I - OCR voorbeelden in de bijlagen

3

Methode

3.1. Fase-indeling

Tijdens dit project is gewerkt met de Design Science Research(DSR) methodiek. Deze kent drie verschillende fasen: de relevance cycle, de rigor cycle en de design/build cycles. Tijdens de relevance cycle zijn bij de opdrachtgevers de requirements[3] van dit project achterhaald. Vervolgens zijn de rigor cycles en design/build cycles door elkaar heen gelopen aangezien de complexiteit van dit project dit vereiste. Er hebben dus in principe vier cycli (voor elk deel in de opdracht één) van onderzoek en praktisch werk tussen elkaar plaatsgevonden. Alle documentatie is vastgelegd in een onderzoeksbestand [20], de applicatie en een testbestand [21].

3.2. Grounding

Een belangrijk onderdeel in de DSR-methodiek is de rigor cycle, hieronder valt grounding; Om een ruwe schets van het bestaande landschap op het gebied van het lezen van PDF's te creëren is er een tabel[4] opgesteld waarin eigenschappen van bestaande tools tegen elkaar worden afgewogen, hieruit is PyMuPDF de beste op de meeste gebieden. Verder is er met een quick literature scan onderzocht welke vormen van coreferencing tools zouden kunnen werken voor dit project door empirisch te testen welke wel en niet werken. Hieruit zijn een aantal opties naar voren gekomen: SpaCy, NeuralCoref, AllenNLP, fast-coref. Door limitaties/deprecated onderdelen van sommige tools is er uiteindelijk gekozen voor fast-coref om de coreferencing te verzorgen. Neo4j zal gebruikt worden als database aangezien dit als een randvoorwaarde was gesteld. Er is onderzoek gedaan naar de werking van Neo4j door de geschreven documentatie van Neo4j zelf door te spitten en door te werken via trial and error. Aan de hand van de bevinding uit het testen met Neo4j is geconcludeerd dat er geen ander advies nodig is voor de benodigde functionaliteiten.

3.3. Design/Build

Binnen de design/build cycles zijn er drie hoofdonderdelen verwerkt. File data extraction, File data storage en Coreferencing. Dit zijn tevens ook de drie functionaliteiten die naar voren komen in het product. Tijdens de design/build fases is er gewerkt via github als sourcecontrol. De design/build fases strekten over het gehele project met een tijdsspanne van grofweg 10 weken waarin om de zoveel weken de focus lag op een ander onderdeel na overleg met de opdrachtgever.

3.4. File data extraction

Een belangrijke afweging die is gemaakt bij data extractie is het tussentijds weergeven van data. Het tussentijds opslaan van resultaten in JSON bestanden geeft overzichtelijk weer per bestand wat iedere functie voor een effect heeft. Zo kunnen vervolgstudies een specifieke functie aanpassen en het verschil op alleen die resultaten vergelijken. Resultaten opslaan in een JSON bestand splitst de file data extraction en file data storage op, waardoor je de functie kunt aanpassen of een andere opslagmogelijkheid kunt testen. Dit biedt flexibiliteit en maakt het mogelijk om verschillende opslagopties te verkennen.

3.5. File data storage

Voor dit project worden relaties tussen datapunten veel gebruikt. "For intensive data relationship handling, graph databases improve performance by several orders of magnitude."[12], dit bevestigt dat de gegeven randvoorwaarde (het gebruik van Neo4j) goed bruikbaar is om geleverde data uit de File data extraction, efficiënt en overzichtelijk op te slaan. Binnen Neo4j zijn relaties namelijk als kern-component geïntergreed en worden er in dit geval zeer grote hoeveelheden van deze aangelegd. Ook is te visualiseren hoe de structuur eruit ziet door gebruik te maken van de User Interface van Neo4j. Er is veel werk gestoken in het maken, testen, evalueren en het opnieuw maken van onderdelen binnen deze database.

3.6. Coreferencing

Bij het maken van de coreferencing functionaliteit is er veel overwogen moeten worden. Efficiëntie en precisie waren nogal vraagstukken op dit gebied. Het analyseren van een zin gaat vrij snel met huidige technologie, maar daar gaat het om 1 paragraaf. Een PDF bestand bevat al gauw 100 paragrafen (afhankelijk van de werking van het File data extraction gedeelte). Hierdoor was het belangrijk om een zo efficiënt mogelijke oplossing te vinden. Uiteindelijk is er dan voor fast-coref gekozen wegens precisie en snelheid. Deze tool draait op een getraind huggingface model[19] en weet vrijwel altijd de correcte entiteiten uit een tekst te herkennen.

[3] Zie J - Requirements in de bijlagen
[4] Zie C - Libraries voor pdf's in python

4

Resultaten

4.1. Functionaliteiten van applicatie

Het geleverde product is in staat PDF bestanden om te zetten naar een hiërarchische JSON structuur. Deze JSON structuur wordt vervolgens geüpload naar een Neo4j database waar er bewerkingen op worden uitgevoerd die later toegelicht worden. In deze graaf database is de data visueel in te zien en is het op te roepen op basis van wat de gebruiker nodig heeft.

4.1.1. File data extraction

De verschillende functies van de gekozen pdf-reader zijn in te stellen in een functie die alle andere functionaliteiten aanstuurt. Meerdere functies hebben twee implementaties; de eerste bekijkt tekst structuur per pagina en de tweede bekijkt de structuur over de hele tekst.
Werkende functionaliteiten bevatten: inhoudsopgave- , tabel-, tekst-, afbeelding- en metadata-extractie. Verder zijn er functies om tekst te segmenteren op absolute structuur en geïnterpreteerde structuur middels reading order interpretatie, tekstopmaak segmentatie en garbage filtering[5].
De meeste functionaliteiten zijn gebaseerd op ingebouwde functies van PyMuPDF. De garbage filter kijkt in essentie alleen of een span een engels woord bevat. Hiervoor worden stemming en lemmatisatie gebruikt om meer woorden te herkennen. De bestandsstructuur groepeert spans die door de garbage filter komen op basis van lettertype en grootte en geeft ze een label ten opzichte van de paragraaf tekst wat het meest voorkomende teksttype is in het bestand of de pagina.
In sommige gevallen kan slechte output van afbeeldingen worden toegeschreven aan opmaakelementen, zoals ruis die wordt toegevoegd om de achtergrond van een pagina visueel interessanter te maken[11]. Andere problemen, zoals afbeeldingen die zijn opgeslagen als .jpx-bestanden of met kleurinversie, zijn bekende kwesties [1]. Het huidige programma slaagt er momenteel in 35,08%[12] van de gevallen in om afbeeldingen correct uit de bestanden te extraheren.

4.1.2. File data storage

Alle data verkregen uit het File data extraction gedeelte moet natuurlijk worden opgeslagen. Dit wordt verzorgd door het File data storage gedeelte. Dit gedeelte bestaat uit twee delen, upload en structure. Bij de upload wordt eerst de ruwe data (onbewerkt) uit een JSON bestand gehaald en ingevoegd in een Neo4j graaf database. Binnen het upload gedeelte wordt daarna ook een geïnterpreteerde en gefilterde documentstructuur geüpload naar de database. In het volgende gedeelte wordt deze dan ook gekoppeld aan de “absolute” data[6] die al eerder was ingevoerd in de database.
Nadat dit upload gedeelte voor alle betreffende bestanden is voltooid wordt de database gestructureerd waarbij alle absolute data een volgorde krijgt toegewezen zodat te achterhalen is waar alles precies vandaan komt. Hierbij wordt ook de geïnterpreteerde structuur[7] gekoppeld aan de “absolute” data. Door deze sequentie aan bewerkingen wordt er een achterhaalbare structuur gevormd. Naast deze datastructuren worden er ook entiteit[8] nodes door het programma aangemaakt waardoor teksten contextueel met elkaar vergeleken kunnen worden, dit wordt verzorgd door het Coreference gedeelte.

Alle aanpassingen, toevoegingen en verwijderingen worden uitgevoerd en opgeslagen middels Cypher queries. Het grote voordeel van Cypher is dat binnen een query direct gewerkt kan worden met variabelen[16]. Alle data wordt dus naar Neo4j geüpload en met elkaar verbonden, waardoor er een samenhangende graaf database ontstaat.

4.1.3. Coreference

Per paragraaf worden de entiteiten die hierin voorkomen in de database gezet en wordt er voor de desbetreffende woorden een connectie gelegd naar een Entity[8] node. Stel, in 2 documenten komt regelmatig de entiteit CEVA voor, dan kan er door middel van de databasestructuur meteen worden vastgesteld dat deze 2 documenten iets gemeen hebben. Op grote schaal zal dit clusters rondom entiteiten veroorzaken.
Naast dat deze clusters ontstaan, kunnen paragrafen ook individueel worden aangeroepen, om zo de paragraaf weer te geven waarbij de verwijzingen zijn vervangen met de entiteiten zodat de tekst zonder context te begrijpen is[9].
Voor coreference-resolution wordt fast-coref gebruikt, een python library die zich richt op efficiënte en snelle coreferentieanalyse, deze library maakt onder de motorkap weer gebruik van SpaCy. Er is gebruikgemaakt van een getraind model voor coreference-resolution[19] dat beschikbaar is via Hugging Face[10], een hub voor het delen van modellen en datasets voor natuurlijke taalverwerking.

4.1.4. OCR

Binnen de scope van dit project is OCR niet volledig geïmplementeerd in de applicatie. Echter is dit onderwerp wel uitgewerkt tot een werkend prototype zodat toekomstige implementatie sneller geregeld kan worden. Overigens gebruikt de gekozen pdf-reader PyMuPDF al indirect OCR.
De Tesseract OCR implementatie is direct aan te roepen binnen het project door de module te importeren. De module werkt door een folder aan afbeeldingen te doorlopen en daaruit de tekst te herkennen.

4.2. Tools en libraries

De tools waar voornamelijk gebruik van wordt gemaakt luiden als volgt:
- PyMuPDF voor het extraheren van data uit PDF's [6].
- Neo4j database voor het opslaan van de data in de vorm van nodes [13][14].
- fast-coref voor het refereren naar entiteiten in een tekst .
- nltk voor de garbage filter [17]

4.3. Doorlooptijd en nauwkeurigheid

Van de drie fasen van de applicatie (extract, upload en structurize) is het uploaden van de database veruit het meest tijdrovend. Dit kan de bottleneck van het product zijn aangezien de database veel moeite krijgt naarmate het aantal nodes toeneemt. Er is te zien in de testresultaten van de doorlooptijd[10], dat de runtime van de neo4j upload per pagina varieert van bestand tot bestand. Dit is ook afhankelijk van hoeveel inhoud de desbetreffende pagina heeft.

[5] Zie N - Garbage filter resultaten in de bijlagen
[6] Zie E - Neo4j graaf database structuur (absoluut) in de bijlagen
[7] Zie F - Neo4j database structuur (geïnterpreteerd) in de bijlagen
[8] Zie G - Neo4j database entiteiten in de bijlagen
[9] Zie M - coreferenced paragraaf in de bijlagen
[10] Zie K - Tijd Resultaten Neo4j upload in de bijlagen
[11] Zie A - Achtergrondruis in de bijlagen
[12] Zie B - Correcte afbeelding aflezing percentages in de bijlagen

5

Discussie

5.1. Image en Table extraction

Als vervolgstappen voor dit project worden voorgesteld: het implementeren van een image restoration-functie en/of een garbage filter-functie. Verder is er nog de wens om OCR toe te passen op de afbeelding om te testen voor tekst in afbeeldingen. Hiervoor bestaat een functie die nog niet hierop geïmplementeerd is. Tables en andere objecten extracten is een openstaand probleem, dat in het theoretisch kader al toegelicht was. Hier is momenteel geen grote verbetering voor aanwezig die open source verkrijgbaar is. Zodra dit wel zo is zou een dergelijk systeem kunnen bijdragen aan het eindproduct.

5.2. Data opslag prestaties

Over het algemeen werkt Neo4j goed voor het opslaan van complexe datastructuren, echter neemt de tijd voor het uploaden en veranderen van data aanzienlijk toe naarmate de hoeveelheid data groeit[13]. Dit is te verwachten, maar zou wel hinderend kunnen zijn. Verdere optimalisatie van de dataopslag is aan te raden in een mogelijke volgende iteratie.

5.3. Hiërarchische structuur

Op het moment van afronding van dit project staat de geïnterpreteerde structuur van PDF's niet altijd in correcte hiërarchische volgorde, vanwege tijdsnood is dit niet gelukt. Dit is wel te implementeren in de toekomst aangezien de JSON data deze structuur wel bevat.

5.4. Inconsistentie in hiërarchische structuur

Bij het inlezen van een PDF zijn er twee opties mogelijk die de zogenaamde reading order beinvloeden. Beide van deze opties (sort=True of sort=False) werken goed voor een ander soort PDF; sort=False stelt reading order op van links naar rechts in kolommen, en dan van boven naar onder per kolom. Dit werkt goed voor simpele paginas, in het bijzonder paginas met meerdere kolommen, zoals in het Allianz bestand uit het testrapport[21]. sort=True werkt beter bij paginas die in een keer van boven naar onder gelezen moeten worden zoals in het Ceva bestand uit het testrapport . Onderscheid maken tussen wanneer welke optie gebruikt moet worden is een openstaande taak[14]. Een AI gebaseerde oplossing kan ook waardevol zijn.

5.5. Garbage filter

Vrijwel alle fouten die worden gemaakt door de garbage filter komen door een tekort aan woorden in het lemmatisatie corpus. Om de prestaties ervan te verbeteren is een andere corpus nodig.

[13] Zie K - Tijd Resultaten Neo4j upload in de bijlagen
[14] Zie L - Inconsistentie in hiërarchische structuur in de bijlagen

6

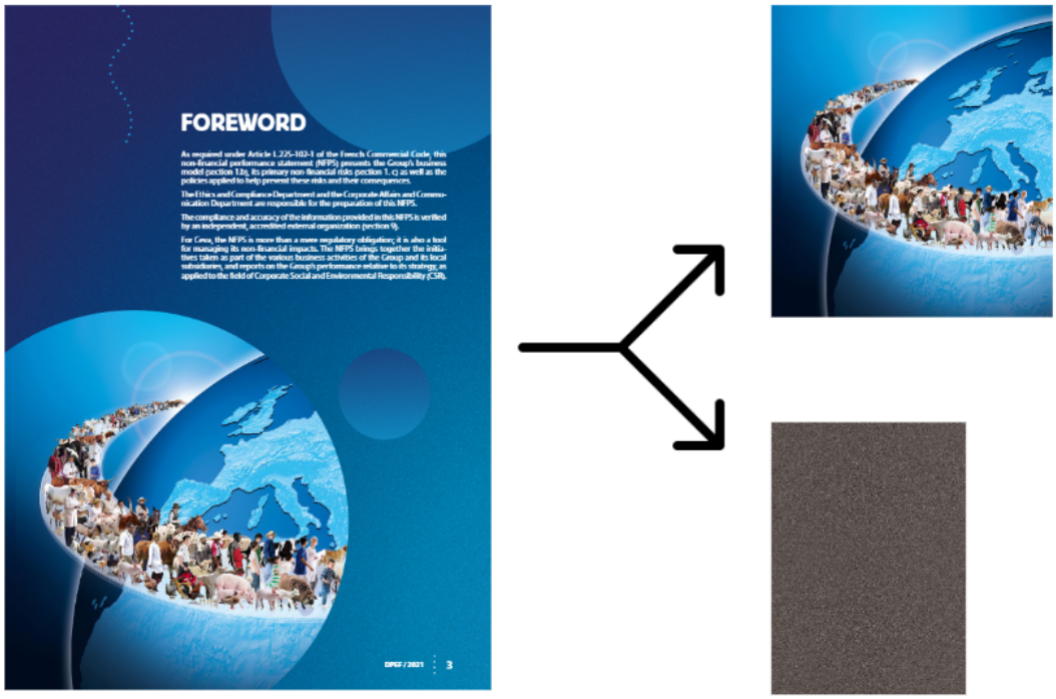
Conclusie

Dit product vormt een ankerpunt voor toekomstige collega's die zich bezighouden met dit onderwerp. Data wordt gelezen uit PDF's, waarna de data wordt geüpload naar een Neo4j database en daar gestructureerd wordt. In de toekomst kan deze structuur gebruikt worden om uitgebreide analyses uit te voeren op het gebied van circulariteit. Kortom, dit project heeft zoals op gedoeld werd een basis gelegd voor toekomstige iteraties van dit project. Het heeft de eerste stap gezet in de richting van het efficiënt en structureel analyseren van PDF bestanden. Dit project is succesvol afgesloten en zowel de opdrachtgever als de teamgenoten zijn content met de resultaten. Voor toekomstige groepen is te adviseren zich bezig te houden met: Het bepalen welke extractie optie voor welke PDF gebruikt moet worden, de Neo4j database structuur verbeteren en de prestaties optimaliseren van voornamelijk de Neo4j upload fase.

Appendices

A

Achtergrondruis



Pagina uit testbestand [7] [21]

B

Correcte afbeelding aflezing percentages

Company	#Total Files	#Extracted Images	#Garbage Images
Allianz (2018)	0	0	-
CMA	170	57	33.5%
EFRAG	49	49[14]	0%
DPEF	75	33	44%
VDL	38	4	10.5%
NASDAQ	167	146	87.4%

[21]

[15] 44 van de 49 corrupted images zijn onleesbaar, omdat het .jpx-files zijn.

C

Libraries voor pdf's in python

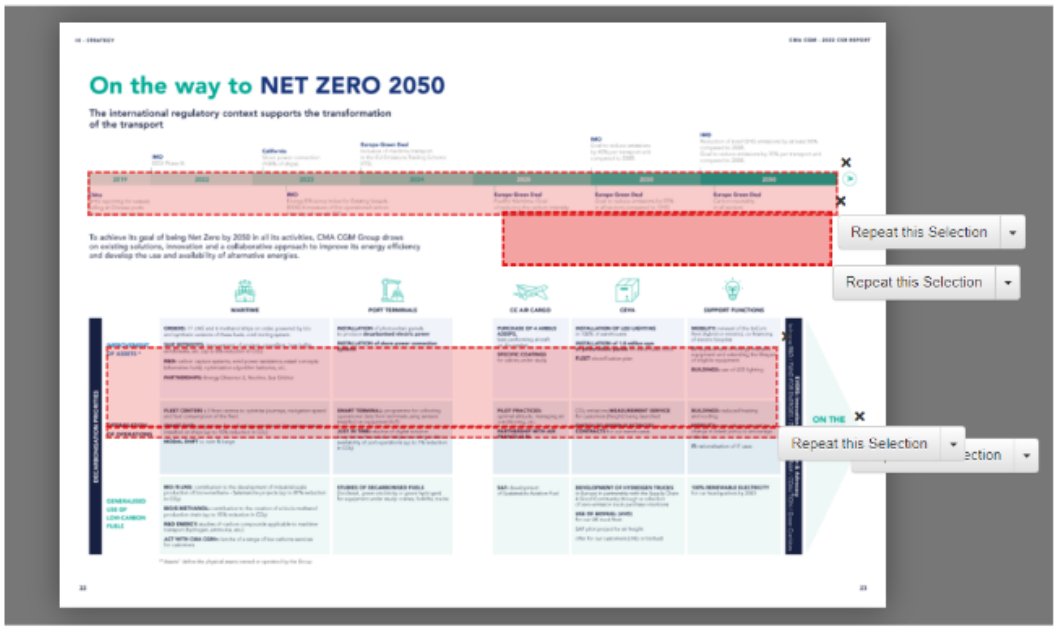
Tool	Tekst	Afbeeldingen	Tabellen	Metadata	Positie	OCR	Fonts	API usage	Dependencies
PyPDF2	Ja	Ja	Nee[16]	Ja	Nee[17]	Nee[18]	Ja	Nee	Nee
Textract	Ja	Nee	Nee	Nee	Nee	Ja	Nee	Nee	Ja[19]
PyMuPDF	Ja	Ja	Ja[20]	Ja	Ja	Ja	Ja	Nee	Ja
PDFtotext	Ja	Nee	Nee[21]	Nee	Ja	Nee	Nee	Nee	Nee
PDFMiner	Ja	Ja	Nee	-	Ja	Nee	Ja	Ja[22]	Ja
Tabula	Nee	Nee	Ja	Nee	Nee	Nee[23]	Nee	Nee	Ja
Tika	Ja	Ja	Nee[24]	Ja	Ja	Ja[25]	Ja	Ja[26]	Ja

[20]

[16] "It does not preserve the table structure." [18]
[17] "In complicated documents the calculated positions might be wrong." [8]
[18] "PyPDF2 will also never be able to extract text from images." [8]
[19] "it extracts information from files in byte format. To convert byte data into a string we need to use other python packages" [18]
[20] "the problem of extracting tables in their original format still exists" [18]
[21] "the main advantage of the package is that it preserves the structure of the PDF text as well as the table structure format." [18] Het kan echter niet herkennen of er een tabel is, alleen de exacte locatie.
[22] "PDFminer provides its service in the form of an API request" [18]
[23] "Caveat: Tabula only works on text-based PDFs, not scanned documents." [5]
[24] "Tika does not currently do this. Please see TabulaPDF as one open source project that extracts tables from PDFs and maintains their structure." [3]
[25] "you can now use the awesome Tesseract OCR parser within Tika!" [2]
[26] "Tika-Python is Python binding to the Apache TikaTM REST services which allows Tika to be called natively in python language." [18]

D

Tabulas tabel herkenning op pagina 13 van het CMA rapport



Afbeelding gemaakt met Tabula op 2024, 5 April [4]

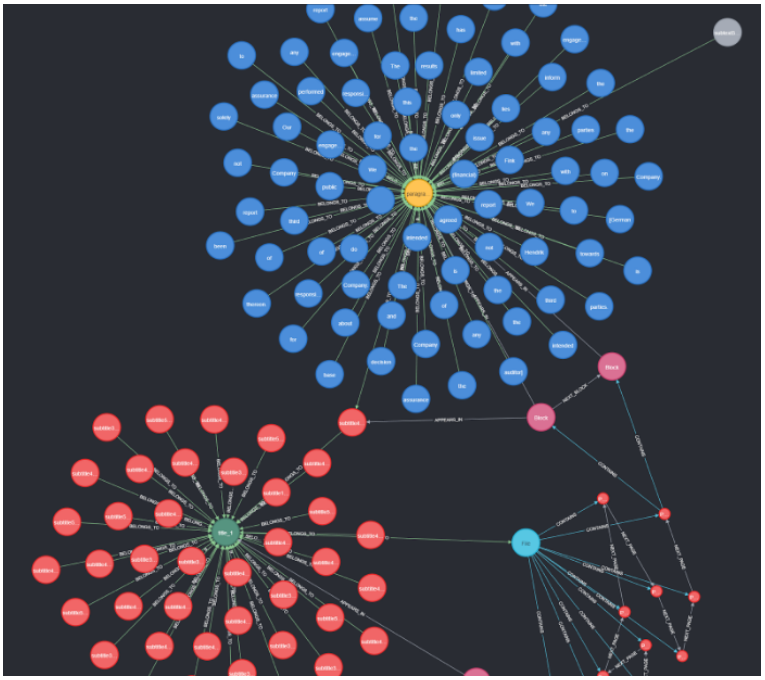
E

Neo4j graaf database structuur (absoluut)

De absolute structuur loopt als volgt:
Absolute structuur: File > nPage > nBlock > Lines > nWord
Ook worden images en tables aan pages gelinkt met de -CONTAINS-> relatie.
Deze structuur bevat zoals de naam suggereert een absolute opvatting van het document. De PDF lezer tool PyMuPDF geeft blocks als output. Deze worden in deze structuur direct ingevoerd zodat de data uit de database factueel te achterhalen is uit de output data van de tool.[21]

F

Neo4j database structuur (geïnterpreteerd)



Geïnterpreteerde structuur: File > nTitle > nSubtitle > nParagraph (> nSubtext) > nPargraphWord
In deze structuur is ook al meteen de link te zien tussen de absolute en de geïnterpreteerde structuur doordat de absolute “blocks” met de geïnterpreteerde elementen worden verbonden.
Deze structuur zal zoals de naam al suggereert een geïnterpreteerde indeling van de PDF volgen. Aan de hand van BELONGS TO relaties is te zien hoe de structuur in elkaar zit. De blocks die voorkomen in een geïnterpreteerde node worden ermee verbonden door middel van een APPEARS IN relatie zodat de originele structuur achterhaald kan worden.[21]

G

Neo4j database entiteiten

In deze afbeelding is te zien dat de woorden(de blauwe nodes) verbonden zijn met een REFERS TO relatie. Sommige nodes komen uit hetzelfde bestand, maar er is ook te zien dat er enkele verschillen van de rest (zie omcirkelde nodes). Dit betekent dat verwijzingen worden gekoppeld over de hele database. Hierdoor zijn verbanden te leggen tussen bestanden op basis van de entiteiten die naar voren komen uit de tekst.[21]

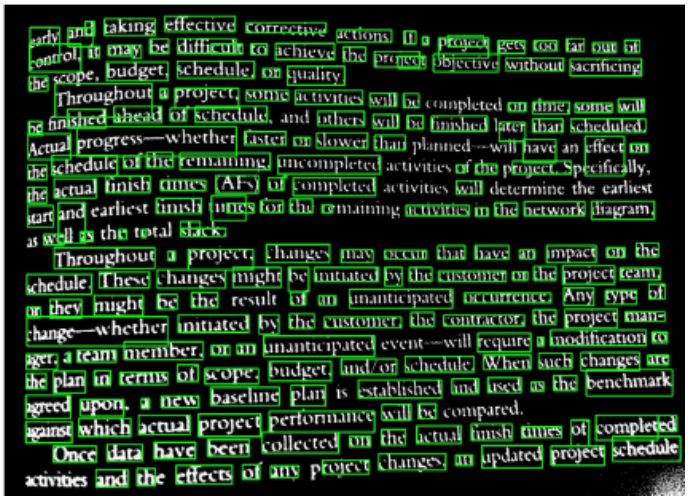
H

Tabel herkenning statistieken van het product

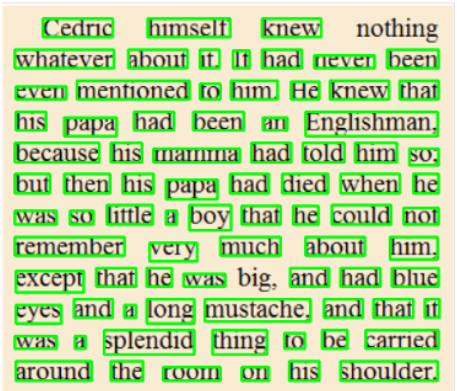
Files	gemiste tabellen	slechte tabellen	correcte tabellen	deels correcte tabellen
CMA	3	6	7	8

[21]

OCR voorbeelden



Hier is te zien hoe de OCR software moeite heeft met verschillende perspectieven, doordat schuine elementen niet altijd onbelangrijk zijn kan dit een probleem zijn.



In dit voorbeeld is te zien hoe een OCR gemakkelijk duidelijke tekst detecteert.

[21]

J

Requirements

ID	Titel	Prioriteit	Eigenaar
NF1	Functies bevatten maximaal 50 regels code.	Must	Pascal Slaats
NF2	De bestanden geschreven in de volgende taal kunnen minimaal herkend worden: ENG (extra: NL, FR, DU).	Must	Pascal Slaats
NF3	Het labelen vanuit het systeem heeft een zelflerende functie.	Should	Pascal Slaats
NF4	Minimaal de bestanden opgeslagen in het volgende documenttype kunnen verwerkt worden: PDF (extra: PDFA).	Must	Pascal Slaats
NF5	Apache Hop bewaakt de workflow.	Could	Pascal Slaats
NF6	Apache Hop stuurt de Scripts aan.	Could	Pascal Slaats

ID	Titel	Prioriteit	Eigenaar
F1	Er is een functie die de tekst uit een bestand omzet naar een lijst van alle zinnen.	Must	Pascal Slaats
F2	Er is een functie die tabellen uit bestanden herkent en omzet naar een lijst van cel-inhouden.	Must	Pascal Slaats
F3	Er is een functie die tekst uit afbeeldingen herkent.	Must	Pascal Slaats
F4	Er is een functie die afbeeldingen uit het document herkent en opslaat.	Must	Pascal Slaats
F5	Er is een functie die verwijzingen tussen woorden/zinnen/alinea's herkent.	Must	Pascal Slaats
F6	Er is een optie om tussen 2 bestanden alle gelijkenissen te vinden.	Should	Pascal Slaats
F7	Er is een functie die voor elke zin in de lijst met zinnen deze zin ontleed.	Must	Pascal Slaats

[20]

K

Tijd Resultaten Neo4j upload

Files	Elapsed Time (s)	# Pages	Average Time per Page (s)
Allianz (2018)	51.28	11	4.66
CMA	1290.33	83	15.55
EFRAG	1811.18	78	23.22
DPEF	1306.75	41	31.85

[21]

L

Inconsistentie in hiërarchische structuur

File	Correct	Relatively Correct	Fout
Allianz (sort=True)	63	18	27
Allianz (sort=False)	103	0	1

[21]

Legenda:

Goede plek/volgorde

Foute plek/volgorde

Foute plek/volgorde maar relatief aan het vorige element wel goed

Allianz (sort=True):

Allianz (sort=False):

M

Coreferenced paragraaf

Door bij het draaien de coreferencing functie te kiezen. Kan de gebruiker paragraph"paragraaf nummer" invullen om zo deze paragraaf contextueel kloppend terug te krijgen. Met contextueel kloppen wordt bedoeld: Verwijzingen van entiteiten zijn vervangen met de entiteiten zelf zodat de lezer het stuk per zin kan gebruiken zonder de gehele context te kennen.

M.1. Input tekst

As of 31 December 2017, with approximately € 1,960 bn of assets under management, we are one of the largest asset managers in the world that manage assets by means of active investment strategies. We run our Asset Management business through two distinct invest-ment management units, Allianz Global Investors (AllianzGI) and PIMCO, both of which operate under the umbrella of Allianz Asset Management GmbH (AAM).

M.2. Nieuwe text

As of 31 December 2017, with approximately €1,960 in of asset under management, Allianz are one of the largest asset managers in the world that manage asset by means of active investment strategies. Our run Allianz Asset Management business through two distinct investment management units, we Global Investors (Allianz GI) and PICO, both of which operate under the umbrella of we Asset Management GmbH (AAM).
Deze functionaliteit werkt zeker niet perfect, maar werkt voor nu acceptabel.

M.3. Performance van het model

Het huidige model, aangedreven door Fast-Coref, vertoont opmerkelijke prestaties in verschillende benchmarktests. Voor OntoNotes behaalde het model een F-score van 80.9. Dit toont aan dat het model consistent en betrouwbaar presteert in het identificeren van coreferenties in een diverse reeks tekstuele contexten.
Voor LitBank bereikte het model een F-score van 80.2. Dit wijst op de effectiviteit van Fast-Coref in het aanpakken van coreferentie-resolutie in literaire teksten, waar nuances en complexiteit vaak een uitdaging vormen.
Daarnaast behaalde het model een F-score van 88.3 voor PreCo.. Dit bevestigt de effectiviteit van Fast-Coref bij het omgaan met coreferentie-resolutie in gespreksdata, waarbij sprake kan zijn van informeel taalgebruik en variabiliteit in de zinsstructuur.
Deze resultaten onderstrepen de indrukwekkende prestaties van het model in het oplossen van coreferentie in diverse contexten en datasets, en bevestigen de effectiviteit van Fast-Coref als kerncomponent van de tool[21].

N

Garbage filter resultaten

Bestanden	Spaties	Nummers	Geen letters	Undecided
Allianz	222	20	121	61
CMA	214	353	263	201
EFRAG	30	398	483	304
DPEF	204	79	209	136
VDL	72	99	197	236
NASDAQ	179	205	321	213

Bestanden	Sub 3 woorden of garbage	Andere talen of gemiste woorden	Weblinks
Allianz	23	33	8
CMA	90	112	-
EFRAG	160	138	6
DPEF	104	31	-
NASDAQ	106	107	-

Bibliography

[1] albertochigua. In the extract images from a pdf, some are shown with inverted colors · issue #725 · pymupdf/pymupdf. GitHub, November 19 2020. URL https://github.com/pymupdf/PyMuPDF/issues/725.

[2] T. & Apache Allison. Tikaocr. cwiki.apache.org, October 14 2021. URL https://cwiki.apache.org/confluence/display/TIKA/TikaOCR.

[3] T. & Apache Allison. Pdfparser (apache pdfbox) - tika - apache software foundation. cwiki.apache.org, July 17 2023. URL https://cwiki.apache.org/confluence/pages/viewpage.action?pageId=109454066.

[4] M. Aristarán, M. Tigas, J. B. Merrill, J. Das, D. Frackman, and T. Swicegood. Tabula (1.2.1). Software, 2018. URL https://tabula.technology/.

[5] M. [jazzido] & TabulaPDF Aristarán. tabula/README.md at master · tabulapdf/tabula, n.d. URL https://github.com/tabulapdf/tabula/blob/master/README.md. GitHub.

[6] Atrifex. Pymupdf. PyPI, April 2 2024. URL https://pypi.org/project/PyMuPDF/.

[7] Ceva. Non-financial performance statement 2021. ceva.com, 2022. URL https://www.ceva.com/wp-content/uploads/2022/09/EN_2022_CEVA_DPEF_FINAL-WEB.pdf.

[8] M. Fenniak. Extract text from a pdf — pypdf2 documentation. pypdf2.readthedocs.io, September 27 2022. URL https://pypdf2.readthedocs.io/en/3.x/user/extract-text.html.

[9] gradient. Untangling the complexities of pdf extraction. Gradient Blog, n.d. URL https://gradient.ai/blog/complexities-of-pdf-extraction.

[10] Hugging Face. The AI community building the future. Hugging Face, n.d. URL https://huggingface.co/.

[11] Maślankowska & Mielniczuk. Intro to coreference resolution in nlp. neurosys, October 4 2023. URL https://neurosys.com/blog/intro-to-coreference-resolution-in-nlp.

[12] Neo4j. Neo4j. Neo4j, May 21 2022. URL https://neo4j.com/why-graph-databases/.

[13] Neo4j. Pypi. PyPI, April 2 2024. URL https://pypi.org/project/neo4j/.

[14] Neo4j. Neo4j deployment center - graph database & analytics. Graph Database & Analytics, April 8 2024. URL https://neo4j.com/deployment-center/.

[15] Neo4j. What is a graph database? Neo4j, n.d. URL https://neo4j.com/docs/getting-started/get-started-with-neo4j/graph-database/.

[16] Neo4j. Introduction - cypher manual. Neo4j Graph Data Platform, n.d. URL https://neo4j.com/docs/cypher-manual/current/introduction/.

[17] NLTK. NLTK. PyPI, January 2 2023. URL https://pypi.org/project/nltk/.

[18] R. Sawarkar. Python packages for pdf data extraction - analytics vidhya - medium. Medium, February 11 2024. URL https://medium.com/analytics-vidhya/python-packages-for-pdf-data-extraction-d14ec30f0ad0.

[19] shtoshni. longformer coreference joint. Hugging Face, n.d. URL https://huggingface.co/shtoshni/longformer_coreference_joint.

[20] Frederiks.A & Rompelberg.M & Theelen.O & Siemers.J. Onderzoeksrapportcirculairnlp.pdf, April 2024. PDF.

[21] Frederiks.A & Rompelberg.M & Theelen.O & Siemers.J. Testrapportcirculairnlp.pdf, April 2024. PDF.