

MODAL: ontwikkeling

Verslag

./DATABLE

Document properties

title	MODAL ontwikkeling: verslag		
project	2025MODAL		
document history	date	name	notes
	4 feb 2025	Henk Vanstappen	draft
	25 sep 2025	Henk Vanstappen	v1
	15 okt 2025	An De Ridder	redactie
	19 nov 2025	rony.vissers@meemoo.be	redactie
	5 dec 2025	Henk Vanstappen	eindredactie
	9 feb 2026	An De Ridder	correctie

**Inhoud**

1. Management samenvatting	3
2. Infrastructuur	5
3. Onderzoek en selectie datasets	7
3.1. Overdracht en selectie	7
3.1.1. Overdracht sets en voorverwerking	7
3.1.2. Selectie	8
3.2. Conclusie bij selectie	8
4. Preprocessing	10
4.1. TIKa-tekstextractie	10
4.1.1. Tekstendatabase	10
4.1.2. Opmerking met betrekking tot tekstextractie	11
4.1.3. Toelichting extractie datering	11
4.1.4. Toelichting extractie e-mail correspondenten	12
4.1.5. Toelichting bepalen bestandstype	12
4.1.6. Vectoriseren (embeddings)	13
4.2. Speech to text	14
4.2.1. Toepassing	14
4.2.2. Opschonen named entities	14
4.2.3. Voorbeelden	15
4.2.4. Evaluatie S2T	16

4.3. Conclusies bij preprocessing	17
5. Verrijkingen	18
5.1. NER	18
5.2. Classificatie	18
5.3. Topic detection	20
5.4. Summarization	22
5.5. Conclusies bij verrijking	22
6. Bevragen	24
6.1. RAG	24
6.1.1. Omgevingsanalyse	24
6.1.2. GPT4All	24
6.1.3. Analyse en resultatenevaluatie	25
6.2. Search	25
6.3. Semantic search	25
6.4. Conclusie bij zoekscenario's	26
7. Verkennen	27
7.1. Text similarity	27
7.2. Deduplication	28
7.3. Relatie tussen analoog en digitaal archief	29
7.4. File browser	29
7.5. Time series	30
7.6. Conclusie bij Verkennen-scenario's	33
8. Algemene conclusie	34
9. Addenda	35
9.1. Schema tekstendatabase	35
9.2. Workflow toepassing scripts	36
9.3. Gated Models	37

1. Management samenvatting

In oktober 2024 startte ADVN in samenwerking met Letterenhuis, MoMu en meemoo het MODAL-project. MODAL staat voor 'Metadateren en Ontsluiten van Digitale Archieven met behulp van Large Language Models'. Large Language Models (LLM's) zijn grootschalige taalmodellen die getraind zijn om zelf tekst te genereren door immense hoeveelheden tekstgegevens te verwerken. Het doel van het MODAL-project is te onderzoeken wat deze vorm van generatieve artificiële intelligentie (GenAI) kan betekenen voor de cultureelerfgoedsector en de kennis hierover actief binnen die sector te delen.

Dit rapport beschrijft de ontwikkeling van de infrastructuur, software en dataverwerking ter realisatie van de doelstellingen van het MODAL-project:

1. het verzamelen en verkennen van datasets
2. de voorverwerking van datasets en onderbrengen in een tekstendatabase
3. het ontwikkelen en uitvoeren van scripts voor het genereren van (meta)data volgens verschillende scenario's
4. validatie en verwerking

Deze infrastructuur is gebaseerd op open-source componenten en large language models (LLM's), met offline verwerking op consumentenhardware.

Voor de selectie zijn vier archieven gekozen op basis van omvang, diversiteit en relevantie: het archief van VUJO¹ (ADVN), organisatiearchief² (Amsab-ISG), auteursarchief³ (Letterenhuis) en het audioarchief van MoMu.

De voorverwerkingsfase omvat tekstextractie met Apache Tika, waarbij ook OCR (Optical Character Recognition)⁴ wordt toegepast op beeldbestanden, en spraak-naar-tekst-conversie met Whisper⁵. Geëxtraheerde teksten en metadata worden opgeslagen in een MongoDB-tekstendatabase. Er is speciale aandacht besteed aan het opschonen van named entities, waarbij de 'name_corrector_rapidfuzz.py' methode de meest efficiënte resultaten opleverde.

Verrijkingen omvatten Named Entity Recognition (NER)⁶, classificatie, onderwerpsherkenning (topic detection), en het creëren van samenvattingen (summerization). De resultaten van deze verrijkingen waren bruikbaar voor inzicht in de inhoud en structuur van ongeordende digitale archieven, maar de kwaliteit was soms onvoldoende voor volledig betrouwbare resultaten, voornamelijk door hardwarebeperkingen. NER, topic detection en summarization werden samen met een (semantische) zoekfunctie als het meest zinvol ervaren.

¹ VUJO (Volksuniejongeren) was de politieke jongerenorganisatie van de Vlaamse politieke partij Volksunie (VU).

² Betreft een born-digital archief van een milieubeweging uit Vlaanderen.

³ Betreft een born-digital schrijversarchief van auteur uit Vlaanderen.

⁴ Voor meer informatie, zie <https://kennisbank.meemoo.be/inzichten-en-praktijk/wat-is-ocr>.

⁵ Voor meer informatie, zie <https://kennisbank.meemoo.be/inzichten-en-praktijk/speech-to-text-whisper-en-noscribe>.

⁶ Voor meer informatie, zie <https://kennisbank.meemoo.be/inzichten-en-praktijk/named-entity-recognition-en-named-entity-linking>.

Het bevragen van de data omvat Retrieval-Augmented Generation (RAG)⁷ met GPT4All, waarbij verschillende modellen zijn getest en deepseek r1 distill llama 8b de meest aanvaardbare resultaten opleverde. Ook is er gezocht met MongoDB Compass voor exacte strings en met een app voor semantisch zoeken op basis van embeddings.

De exploratiefase richt zich op het analyseren van de inhoud en structuur van het archief, inclusief tekstgelijkenis (meten van similarity tussen documenten), deduplicatie, het opsporen van relaties tussen analoge en digitale archieven (met beperkt succes), en een file browser met samengevoegde verrijkingen, en tijdreeksanalyses om het verloop van kenmerken over de jaren heen te visualiseren.

Validatie van de methoden is uitgevoerd en de resultaten zijn gedocumenteerd in gerelateerde documenten.

⁷ Retrieval-Augmented Generation (RAG) verbetert de output van een LLM doordat eerst relevante informatie wordt opgehaald uit externe bronnen, zoals databases, documenten of het web. De LLM genereert vervolgens een antwoord dat is gebaseerd op zowel de opgehaalde informatie als de eigen kennis.

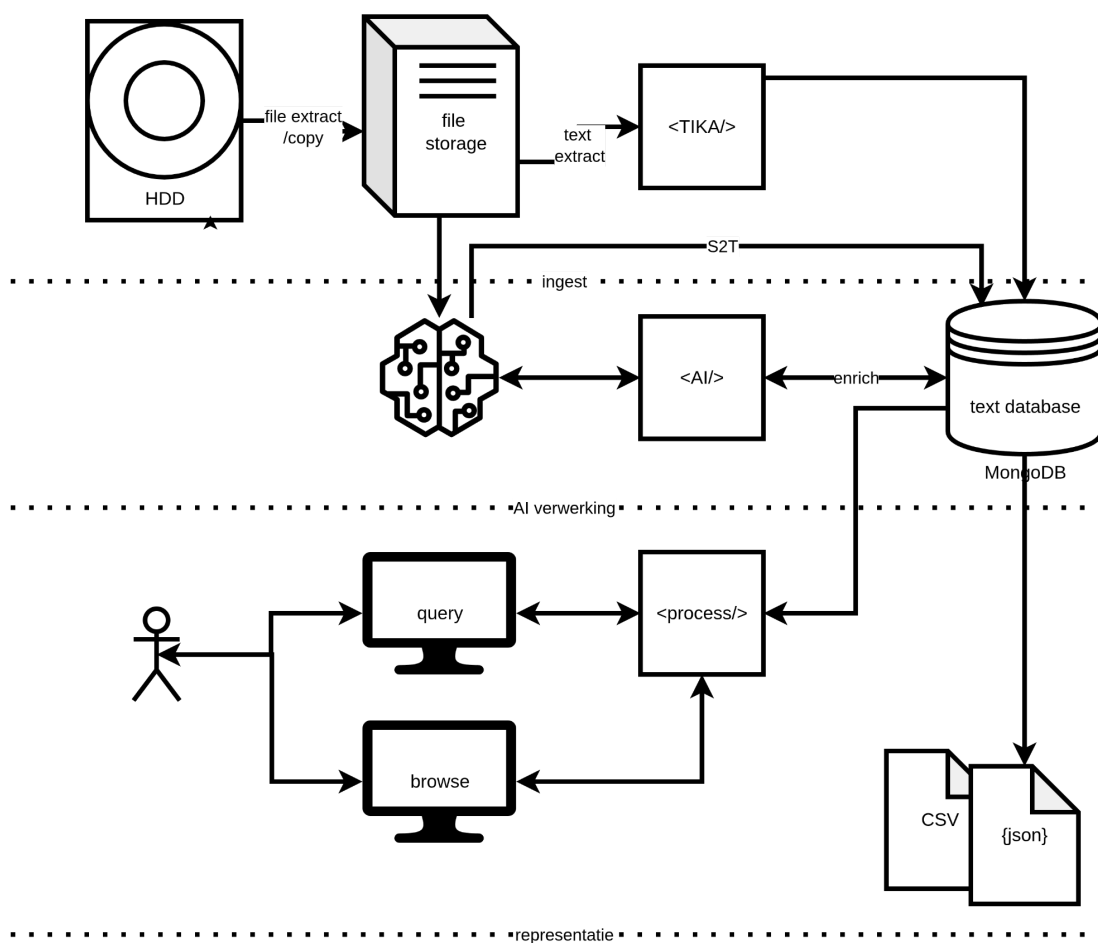
2. Infrastructuur

Vereisten voor de architectuur werden binnen het MODAL-project als volgt gedefinieerd:

- open source componenten
- open source LLM's
- alle verwerking gebeurt offline
- gebruik op consumer hardware, d.w.z. beperkt in RAM, geen of kleine GPU.

De infrastructuur bestond uit volgende basiscomponenten:

1. ingest-laag: zorgt voor
 - a. het opnemen van bestanden in een genormaliseerd formaat
 - b. het genereren van technische metadata
 - c. het omzetten van audio naar tekst
2. AI-laag: zorgt voor het verrijken van documenten met behulp van AI
3. representatielaag: zorgt voor het doorzoekbaar maken en representeren van data voor gebruik en validatie.



Afb. 1: overzicht basisprocessen en infrastructuur

Specificaties Ontwikkelomgeving:

- OS: Linux Tuxedo OS (Ubuntu based)
- Hardware:
 - 20 × 12th Gen Intel® Core™ i7-12700H
 - 62.5 GiB of RAM
 - NVIDIA GeForce RTX 3050 Ti Laptop GPU (4GB)
- Python (3.1x), belangrijkste libraries:
 - [pymongo](#) voor connectie met de MongoDB database ('tekstendatabase')
 - [tika](#) voor tekstextractie
 - [transformers](#) voor gebruik van modellen
- De meeste modellen worden verkregen uit de [Huggingface repository](#).

Hieronder vermelde scripts zijn gepubliceerd op Github in drie repositories:

- [MODAL textdatabase](#): ingest
- [MODAL enrichments](#): verrijkingen met behulp van AI
- [MODAL reporting](#): representatie van gegevens

3. Onderzoek en selectie datasets

Dit hoofdstuk beschrijft het onderzoek en de selectie van datasets voor verdere verwerking binnen het MODAL-project. Het omvat de overdracht en selectie van gegevens. De projectpartners maakten een eerste selectie van geschikte digitale archieven die als dataset konden worden verwerkt. Vervolgens werden deze geanalyseerd om een beperkte testset samen te stellen.

3.1. Overdracht en selectie

3.1.1. Overdracht sets en voorverwerking

- Om veiligheidsredenen en omwille van de omvang van de archieven gebeurde de overdracht van bestanden offline op een harddisk of USB-stick.
- Dragers met brondata worden read-only gekoppeld
- De geselecteerde sets worden gekopieerd naar externe schijf ('LaCie')
- *Disk images* (bv. ISO files) werden uitgepakt. In bepaalde gevallen werden archiefformaten genegeerd, omdat het bestand ook reeds in uitgepakte vorm in het archief aanwezig was (bijvoorbeeld .dd-files bij ADVN). Individuele bestanden werden uitgepakt door TIKa (zie verder)
- WARC-files (gebruikt voor onder andere gearchiveerde websites) werden uitgepakt.

Organisatie	drager_naam	set/folder	kopie_opmerkingen
ADVN	ADVN0	VEA229_ABN_verenigingGevaert	
ADVN	ADVN1	VEA258_EFA	
ADVN	ADVN2	VEA225_UNPO	
ADVN	ADVN3	VEA260_VUJO	
ADVN	ADVN_USB	VEA260_aanvulling	bevat uitgepakte dd files; toegevoegd aan VEA260_VUJO
ADVN	ADVN_USB	X_dataset	WARC en WARZ (duplicaten)
AMSAB	AMSAB_DISK11	AMSAB_archieven_d_1995_014	
AMSAB	AMSAB_DISK11	smc1800	
AMSAB	AMSAB_DISK11	smc1801	
AMSAB	AMSAB_DISK11	smc1802	
AMSAB	AMSAB_DISK11	smc1803	
AMSAB	AMSAB_DISK11	smc1804	
AMSAB	AMSAB_DISK11	warcs	
LH	LH	J-geanonimiseerd-Ce	niet weerhouden
LH	LH	JP-geanonimiseerd-.iso	iso uitgepakt
LH	LH	NT-geanonimiseerd	
LH	LH	UV-geanonimiseerd	

LH	T7	HH_71_geanonimiseerd	
LH	T7	\MM395_lopende_projecten	
LH	T7	P305_g_geanonimiseerd	
LH	T7	P305_m_geanonimiseerd	
LH	T7	P305_s_geanonimiseerd	
MOMU		1IY8fWGwVgblYxcJ2YJxj5bsG2labh__C	audio en transcript

Overzicht per contentpartner:

Name	File Size	Files	% of Parent	% of Total
ADV N	962 GB	206,215	21.5%	21.5%
AMSAB	866.2 GB	537,975	56.1%	56.1%
LH	1.6 TB	215,511	22.5%	22.5%
MOMU	10.1 GB	110	0.0%	0.0%
TOTAAL	3.4 TB	959,811	100.0%	100.0%

3.1.2. Selectie

Na een eerste analyse en overleg met de data-eigenaars werd een aantal archieven geselecteerd voor verdere verwerking. De criteria die hierbij gehanteerd werden waren:

- omvang: er werd gekozen voor archieven met een gemiddelde grootte
- diversiteit aan bestanden
- relevantie voor de data-eigenaar

Verder werd de voorkeur gegeven aan digitale archieven die in een 'klassieke' directory structuur waren ondergebracht. Bijzondere archiefbestanden met een afwijkende interne structuur werden niet verwerkt, zoals WARC-files, waarvan de interne structuur niet gedocumenteerd was.

Uiteindelijk zijn er vier archieven geselecteerd voor verdere verwerking:

- het archief van de VUJO (ADV N)
- het archief van een Vlaamse milieubeweging (Amsab-ISG)
- het archief van auteur uit Vlaanderen (Letterenhuis)
- het audioarchief van MoMu

3.2. Conclusie bij selectie

De overdracht van bestanden gebeurde offline vanwege veiligheidsredenen en de omvang van de archieven. De onmogelijkheid tot online overdracht moet in overweging worden genomen bij de ontwikkeling van applicaties voor archiefanalyse, waarbij rekening moet worden gehouden met beperkte netwerksnelheid.

De analyse van WARC-bestanden bleek te complex door gebrek aan documentatie over de structuur, waardoor deze niet verder zijn geanalyseerd. Een vergelijkbare conclusie werd getrokken voor andere socialemediabestanden, zoals tweets.

Uiteindelijk zijn de vier hogergenoemde archieven geselecteerd voor verdere verwerking. De overige archieven werden niet of nog onvolledig verwerkt.

Zoals uit latere analyses zal blijken, bevatten de archieven vaak minder relevante bestanden (bijvoorbeeld systeembestanden) en duplicaten. Dit blijkt een sterke impact te hebben op de verwerkingssnelheid en de kwaliteit van de verrijkingen.

4. Preprocessing

De voorverwerking (pre-processing) fase heeft als doel de data bruikbaar te maken voor analyse met behulp van generative AI-tools voor verrijking. Dit betreft:

- (1) extractie van teksten en relevante bestandsmetadata uit de bestanden
- (2) omzetten van audio naar tekst

4.1. TIKA-tekstextractie

Voor extractie van tekst en metadata wordt gebruik gemaakt van [Apache Tika](#). Alternatieven zijn extractie- en embeddingtools die geoptimaliseerd zijn voor embeddings/LLMs, zoals [Unstructured](#). Voor het MODAL-project werd gekozen voor TIKA, voornamelijk vanwege de bredere ondersteuning van obsoleete bestandsformaten (bv. WordPerfect).

4.1.1. Tekstendatabase

De tekstendatabase bevat gegevens over bestanden in het digitaal archief, inclusief alle geëxtraheerde teksten. Voor deze database gebruiken we [MongoDB](#). MongoDB is een opensource document-georiënteerde database. De documenten worden in de vorm van BSON (binair JSON) opgeslagen en de structuur van deze documenten is flexibel (zogenaamde NoSQL-database).

Voor elk archief wordt in de database bij verwerking een Collection gecreëerd, die de naam van de folder draagt. De database is in de pythonscripts aanspreekbaar via het default adres (localhost:27017/)

De eigenschappen van het databaseschema worden vermeld in bijlage.

De [scripts](#) automatiseren het proces van tekst- en metadata-extractie uit bestanden, tekstverwerking en opslag in een MongoDB-database.

- selectie brondata:
 - op basis van pad naar folder met de originele bestanden
 - filter om specifieke bestandstypes uit te sluiten, op basis van bestandsnaam en extensie:
 - filter om specifieke folders uit te sluiten
- extractie tekst
 - archiefbestanden (tar, AR, ARJ, CPIO, Dump, Zip, 7Zip, Gzip, BZip2, XZ, LZMA, Z and Pack200) worden uitgepakt
 - OCR wordt toegepast op beeldbestanden (incl. PDF, bv. scans van documenten)
 - verwijderen van meerdere opeenvolgende witregels
- extractie metadata
 - MIME-type
 - aantal woorden
 - taal
 - technische metadata
 - gebruikte TIKA-parser
 - pad naar bestand
 - bestandsnaam
 - datum: optioneel wordt de *creation date* geëxtraheerd uit de file-properties (bijvoorbeeld in MSWord- of PDF-bestanden). Zie opmerking verder.

- creator: waar mogelijk wordt de creator geëxtraheerd uit de file-properties (bijvoorbeeld in MSWord- of PDF-bestanden).
- e-mailcorrespondenten: optioneel worden e-mailcorrespondenten geëxtraheerd. Zie opmerking verder.
- Het script **textdatabase_stats.py** creëert statistieken voor elke collectie in CSV-formaat.

4.1.2. Opmerking met betrekking tot tekstextractie

- Geïdentificeerde bestandsformaten die worden niet ondersteund:
 - Microsoft Word for Macintosh 5.0
 - WordPerfect ouder dan 6.0
 - MS Publisher (.pub)
- Extractie van e-mailbijlagen wordt ondersteund voor open formaten. Uit .msg-bestanden kunnen geen bijlagen worden gehaald (mogelijk wordt dit beter ondersteund in recentere versie van TIKA)⁸. Dit kan mogelijk worden opgelost met een afzonderlijke tool of bibliotheek.⁹
- Dit proces kan in de toekomst gecombineerd worden met de output van DROID¹⁰, waarbij de bestandsidentificatie van DROID wordt gebruikt om te verwerken bestanden te filteren.

4.1.3. Toelichting extractie datering

Er zijn verschillende dateringen verbonden aan een bestand: creatie, laatste wijziging en laatste raadpleging.

Datums van wijziging en laatste raadpleging zijn vaak het minst informatief en het minst betrouwbaar. We beperken ons daarom tot extractie van de datum van creatie.

Datums kunnen op verschillende manieren worden geëxtraheerd:

1. Bestandseigenschappen opgeslagen in het bestand ('file properties date'): het gaat om Datums die door de toepassing zelf (bijvoorbeeld MS Word) worden ingebed in het bestand. Niet elke software en elk bestandstype ondersteunt dit echter. Ook hier zijn afwijkingen mogelijk: een gescand document draagt de datum van digitalisering, niet van creatie van het origineel.
2. Bestandseigenschappen, opgeslagen in het *file system* (file attributes): dit is steeds beschikbaar, maar niet altijd betrouwbaar (bijvoorbeeld de kopie van een bestand kan als creatiedatum worden aanzien). De manier waarop deze eigenschappen worden opgeslagen, verschillen ook volgens het gebruikte file system - vooral FAT is erg beperkt in het bewaren van bestandseigenschappen.
3. Datum vermeld in het document, bijvoorbeeld de datum in de aanhef van een brief. Deze kan enkel geëxtraheerd worden door interpretatie, eventueel door middel van een LLM.

add_estimated_creation_date.py:

- Datums worden opgenomen in ISO-formaat, bijvoorbeeld "2013-03-19T10:52:00Z".

⁸ <https://dist.apache.org/repos/dist/release/tika/3.2.0/CHANGES-3.2.0.txt>

⁹ Bijvoorbeeld <https://www.matijs.net/software/msgconv/>

¹⁰ DROID is een softwaretool ontwikkeld door The National Archives (UK) om de formaten van digitale bestanden te identificeren. Voor meer informatie, zie.

<https://www.nationalarchives.gov.uk/information-management/manage-information/policy-process/digital-continuity/file-profiling-tool-droid/>

Volgende volgorde wordt toegepast voor het bepalen van een (benaderende) creatiedatum (estimated_creation_date):

1. Als een datum in de bestandseigenschappen wordt gevonden, wordt deze door TIKa als creation_date opgenomen. Voor een groot deel van de bestanden volstaat dit, maar vooral bij bestanden die dateren van voor 2000 ontbreekt deze eigenschap vaak.
2. De datum van creatie volgens het filesystem wordt in de property creation_date_filesystem opgenomen.
3. Samenvoegen: als er een creation_date is, wordt deze gekopieerd naar de property estimated_creation_date. Is er geen creation_date, dan wordt de waarde van creation_date_filesystem genomen.

4.1.4. Toelichting extractie e-mail correspondenten

In de TIKa extract.py werd een functie toegevoegd om correspondenten te extraheren. Dit script is geschikt voor e-mailberichten van type .eml en .msg.

De functie **tika_extract_correspondents.py** in TIKa_extractor.py haalt informatie uit e-mailbestanden. Het leest de inhoud van een e-mailbestand en extraheert namen en adressen van de afzender, ontvanger en eventueel kopieën (cc) van de e-mail.

Het script add_email_correspondents.py gebruikt deze functie voor alle bestanden met extensie .msg of .eml en schrijft de resultaten weg naar de database.

Voorbeeld:

```
"sender_email": [
    "geanonimiseerd"
],
"sender_name": [
    "R[...] [...]"
],
"recipient_email": [
    "geanonimiseerd.be",
    "info@geanonimiseerd.be",
    "geanonimiseerd@groen.be"
],
"recipient_name": [
    "geanonimiseerd",
    "geanonimiseerd",
    "geanonimiseerd@groen.be"
],
"cc_name": []
```

Opmerking: de resultaten wisselen lichtjes naargelang het oorspronkelijk bestandstype.

4.1.5. Toelichting bepalen bestandstype

Het TIKAScript bepaalt het [mime-type](#) van elk bestand, maar dit is niet altijd bruikbaar (te specifiek).

Script **mime_to_genericType.py** bevat een functie om bestanden te categoriseren op basis van

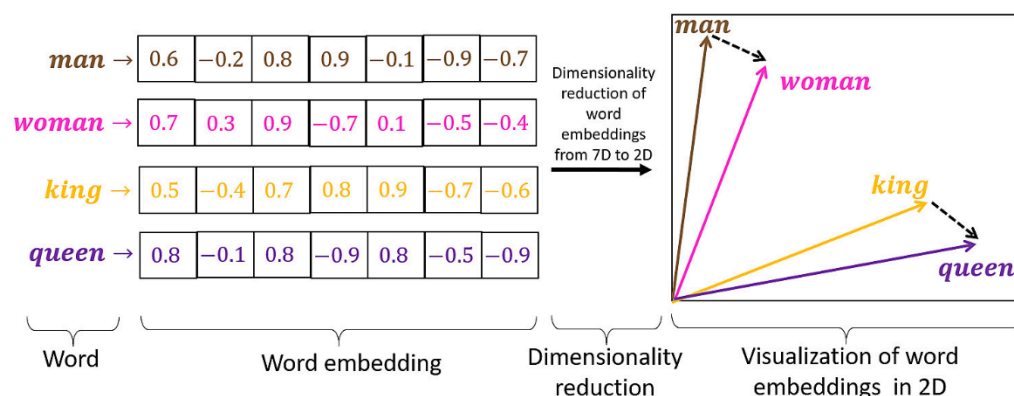
- (1) het eerste lid van het mime-type het type te bepalen (image, document, audio...)
- (2) indien eerste lid 'application' is: regels om type te bepalen (bijvoorbeeld als het MIME-type "dbase" bevat, is het type "gegevensbestand")
- (3) eventueel wordt ook de extensie gebruikt om de categorie te bepalen
- (4) de categorie wordt toegevoegd aan het tekstendatabaserecord
- (5) het script rapporteert eventuele niet-gecategoriseerde types

Volgende categorieën wordt toegepast:

1. tekstbestand (doc, pdf, txt)
2. rekenbestand (.xlsx)
3. presentatiebestand (ppt)
4. beeldbestand (jpg, tif)
5. geluidsbestand (mp3, mpeg)
6. videobestand (mp4)
7. berichtbestand (msg)
8. compressiebestand (zip, tar)
9. gegevensbestand (xml, csv)
10. lettertypebestand (font file)
11. andere (bijvoorbeeld application/x-msdownload, application/vnd.iccprofile)

4.1.6. Vectoriseren (embeddings)

Embeddings zijn vectorrepresentaties ('getallenreeksen') van woorden, zinnen of andere gegevens, waarbij de semantische betekenis en relaties tussen deze elementen worden vastgelegd (zie illustratie). In plaats van woorden als afzonderlijke entiteiten te behandelen, brengen embeddings woorden dicht bij elkaar in de vectorruimte als ze vergelijkbare betekenissen of contexten hebben.



Voor het MODAL-project worden Transformer-gebaseerde embeddings gebruikt op basis van BERT-taalmodellen¹¹ (script **create_embeddings.py**). Deze embeddings kunnen bijvoorbeeld worden gebruikt om teksten te clusteren en semantisch te doorzoeken (zie verder). Deze embeddings worden toegevoegd aan de tekstendatabase.

4.2. Speech to text

Een onderdeel van het project bestond in het onderzoek naar het gebruik van open source tools voor het omzetten van audiobestanden naar tekst. Deze teksten werden vervolgens in de tekstendatabase opgenomen om als andere documenten te worden verwerkt.

4.2.1. Toepassing

Er werd een script ontwikkeld (**speech_to_text.py**) op basis van Whisper, dat audiobestanden selecteert uit de tekstendatabase, de audiobestanden verwerkt met Whisper en de resultaten (geëxtraheerde tekst) toegevoegd aan het record in de tekstendatabase, met toevoeging van het aantal woorden en de taal. Hiervoor werd gebruikgemaakt van het model openai/whisper-medium. Dit bleek een erg traag model. Betere performantie bij gelijke kwaliteit werd bereikt met het FastWhisper model.

Deze directe verwerking laat echter geen mogelijkheid om de tekst te valideren en eventueel te bewerken voor het naar de database wordt weggeschreven.

Daarom werd een alternatieve benadering uitgewerkt, waarbij:

- audiobestanden via gewone TIKAS-procedure aan tekstendatabase werden toegevoegd
- met een bestaande tool (bv. SpeechNote of [noScribe](#)) een transcriptie werd gemaakt (met de optie een srt-formaat te gebruiken, waarin tijds codering is toegevoegd). SpeechNote is voorzien van een eenvoudige GUI, wat ook erg handig is voor het testen van modellen
- script **srt_to_txt.py** srt srt-bestanden (output van speech-to-text-software) omzet naar platte tekst (verwijderen van tijdcodes, samenvoegen van lijnen die niet op een punt, vraagteken e.d. eindigen.)
- (optie:) **name_corrector.py** toegepast werd voor correcties van verkeerd herkende namen op basis van namenlijst wordt uitgevoerd → zie verder
- **s2toutput_to_textdatabase** taal detecteert, woorden telt en transcripties toevoegt aan de tekstendatabase

4.2.2. Opschonen named entities

Bij een eerste evaluatie werd vastgesteld dat *named entities* niet steeds goed herkend worden, daarom is er nood aan een manier om die op te schonen. Mogelijke benaderingen zijn:

- (1) *dictionary based*: er wordt een namenlijst gebruikt. De namen worden vergeleken met *strings* in de tekst. Wanneer er een gedeeltelijke overeenkomst is, wordt de string in de tekst vervangen door de

¹¹ BERT staat voor Bidirectional Encoder Representations from Transformers en verbetert het begrip van de context in taalverwerking.

naam. Matching kan op basis van een fuzzy matching algoritme.

De *dictionary* kan worden gecreëerd door:

- manueel een lijst op te stellen
- met NER entities opsporen, en deze vervolgens manueel verbeteren
- gebruik maken van een publiek beschikbare dictionary, zoals Wikidata.

(2) gebruik van een LLM, met de instructie namen te corrigeren.

Hiervoor maakten we gebruik van [Wikipedia Notable people](#): deze lijst bevat de naam, thema en beroep en andere metadata over personen. Niet alle gewenste personen bleken echter in deze set te worden vermeld.

Opmerking: de kennis van LLM's kan gebruikt worden om correcties uit te voeren - maar dit vereist veel rekenkracht.

Dictionary:

- Een eigen dictionary in de vorm van een CSV-bestand.

Methodes:

- Methode 1: Fuzzy match in combinatie met NER: eerst wordt NER toegepast, vervolgens worden de ontdekte namen vergeleken met de dictionary en gecorrigeerd.
 - met Spacy worden named entities geselecteerd, o.a. 'Alessandro', 'Dirk Bickenbergs'
 - Wikipedia-dataset wordt gefilterd op domeinen 'fashion', 'design'
 - fuzzy match op basis van **rapidfuzz**-library, gecombineerd met beperking in variatie in lengte → zou moeten werken op OCR-fouten, niet goed genoeg voor s2t
 - opmerking: fuzzy match op basis van **doublemetaphone**-library: geen goede resultaten, bv.:
 - 'Dirk Bickenbergs' → geen match (omdat M niet gelijk is aan N) ('Dirk Bickemberghs' wordt wel goed gematched)
 - 'Alessandro' → Jill Sander (!)
- Methode 2: **name_corrector_rapidfuzz.py** directe correctie, zonder NER
 - Het script gebruikt een CSV-bestand met correcte namen als referentie. Deze namenlijst kan eventueel aan de hand van van een NER-algoritme worden gecreëerd, maar hierbij is het belangrijk de *threshold* laag in te stellen (<50) om een zo ruim mogelijk vangst te verzekeren. Vervolgens moet de namenlijst handmatig worden opgeschoond
 - Er is een matchingsdrempel van 80% ingesteld
 - Maximum van 5 woorden wordt gebruikt voor naamherkenning
 - Het script leest teksten uit een tekstendatabasecollectie ("MOMU_audio"). Voor elke tekst zoekt het naar mogelijke namen. Deze worden vergeleken met een lijst van correcte namen uit het CSV-bestand. Namen die voldoende overeenkomen (>80%) worden vervangen door hun correcte versie
 - `normalize_for_match()`: maakt tekst vergelijkbaar door alles naar kleine letters om te zetten en leestekens te verwijderen
 - Voor elk record wordt de gecorrigeerde tekst opgeslagen in een nieuw veld "extracted_text_corrected"
 - De originele tekst blijft behouden

De tweede methode bleek na tussentijdse evaluatie de meest efficiënte resultaten op te leveren.

4.2.3. Voorbeelden

Tekst werd geëxtraheerd door middel van Speech-to-Text (S2T)-algoritme:

```
S2T = "yeah it's a bit yeah i have many examples of uh of it later because i
also found just sketches you know uh and and of course i know the shoes so i
i recognize them immediately it's like yeah you cannot uh the first
presentation in london wow that's the one is it that's the he said yes
that's one where they call them the six yeah [...]"
```

Manuele transcriptie:

```
audio_Iz = '''- Yeah, I have many examples of uh of it later, eh, because I
also found just sketches you know uh -

-- Amazing

- uh and of course I know the shoes so I

-- yeah

- I recognise them uh immediately

[....]
```

4.2.4. Evaluatie S2T

Om S2T-transcripties te vergelijken met bestaande (door een mens gemaakte) transcripties en de automatisch opgeschoonde transcripties, werd een script geschreven (**validate_S2T_jiwer.py**) dat gebruikmaakt van de [Jiwer library](#).¹²

Dit script negeert een aantal verschillen:

- tekens tussen [] en <>
- hoofd/kleine letters
- meerdere spaties
- spaties aan het begin en einde van een lijn
- interpunctie
- regeleinden
- koppeltekens
- woorden ["he","yeah","uhh","ah","uh","yep","mhmm","um","umm","eh","uhhh","uhh","mmm"]

Toch bleken de verschillen (gemeten in Word Error Rate, WER) nog groot (cf. tabel)

file	WER GT vs. original	GT vs. corrected
20240812_RobertMcCaffrey2.WAV	0.22014221448265853	0.22065012334929618

¹² Opmerking: de pip-library bleek niet up-to-date, maar kan manueel worden geïnstalleerd.

20240819_MarinaYee1.WAV	0.5249582764375664	0.5263237748444849
20240819_MarinaYee2.WAV	0.4321284038685781	0.43260854653954317
20240829_FlorienPénard1.WAV	0.29179580674567	0.2933454876937101
20240829_FlorienPénard2.WAV	0.3950518754988029	0.39584996009577017

- WER ligt hoog (ca 22-50%%) bij vergelijk S2T <> transcripties
- De impact van naamcorrectie op de WER is beperkt

4.3. Conclusies bij preprocessing

Voor de verwerking is Apache TIKa gebruikt, dat de meeste bestandsformaten goed kan verwerken. De ingebouwde OCR-functie was vooral nuttig voor gescande PDF-documenten. OCR op andere afbeeldingen leverde echter weinig bruikbare tekst op.

Het omzetten van gesproken audio naar tekst had veel last van een beperkte herkenning van persoonsnamen, waardoor semi-automatische correctie nodig is.

Door teksten te extraheren en gestandaardiseerd in een database op te slaan, wordt AI-verwerking sterk vereenvoudigd. Deze voorbereiding biedt ook selectiecriteria voor verdere verwerking, zoals het filteren op documenten met een minimaal aantal woorden of een specifiek bestandstype.

Tekstextractie dient dus als voorbereiding op het gebruik van AI voor archiefverkenning, maar levert zelf al belangrijke inzichten op, zoals een indeling in bestandstypen, datering en (voor e-mails) correspondenten. Deze gegevens kunnen later worden gecombineerd met inzichten uit AI-analyse, bijvoorbeeld in de vorm van visualisaties.

5. Verrijkingen

Met verrijkingen worden metadata gecreëerd op basis van de geëxtraheerde teksten, om inzicht te verkrijgen in de inhoud van de documenten. Deze verrijkingsmetadata maken het mogelijk documenten te clusteren en te bevragen.

We maken onderscheid in:

- Named Entity Recognition (NER)
- Classificatie van documenten
- Topic detection (onderwerpsherkenning)
- Samenvattingen (summerization)

5.1. NER

Verschillende benaderingen en modellen voor NER werden getest, waaronder generieke modellen en specifieke NER-modellen [Wikineural](#) en Spacy.

Het script op basis van Wikineural werd geselecteerd als meest performant (zie script **ner_wikineural.py**)

- NER_persons selecteert personen
- NER_organisations selecteert organisaties
- NER_locations selecteert locaties
- NER_miscellaneous selecteert andere topics
- voegt filter toe op alle entities met confidence > 0.6

NER met generiek LLM heeft als voordeel dat er meer specifieke vragen kunnen worden gesteld, bv. "geef enkel namen van landen" (of talen, volkeren) - zoals gevraagd door ADVN.¹³

NER_gemma3.py is een script dat gebruik maakt van een compact LLM voor NER: de kwaliteit is iets beter dan Wikineural met het google/gemma-3-4b-it model, maar dit vereist veel meer rekenkracht, en is daardoor weinig geschikt in een productieomgeving.¹⁴

5.2. Classificatie

Het doel van classificatie is het indelen van documenten volgens een vooraf bepaalde lijst met types of genres. Verschillende benaderingen werden onderzocht:

1. *zero shot*: hierbij wordt een aantal klassen voorgesteld, waaruit een keuze moet worden gemaakt. Dit vereist een krachtig (intelligenter) model. Deze aanpak is bruikbaar bij grote taalmodellen (GPT4 etc.), maar het resultaat kan nog steeds last hebben van hallucinaties.
2. *few shot*: hierbij wordt in de prompt een aantal voorbeelden gegeven van teksten en de klasse waarin ze thuishoren. Dit betekent dat de voorbeelden in elke prompt moeten worden meegegeven,

¹³ Voor methodologie zie bv. [Brian Llinas. Do Large Language Models Agree on Entity Extraction?](#)

¹⁴ In augustus 2025 werd de LangExtract library van Google gepubliceerd. Om deze te gebruiken in combinatie met een lokaal LLM, moet het model via een tool als vLLM of Ollama door een API beschikbaar worden gesteld, waarna LangExtract kan worden gebruikt.

waardoor deze aanpak veel resources vraagt. Dit is haalbaar met een kleiner (OS) model, maar vanwege de omvangrijke prompt verloopt dit erg traag.

3. *fine tuning*: hierbij wordt een bestaand model aangeleerd om onderscheid te maken in een aantal klassen. *Model fine tuning* vraagt veel resources, maar verwerking is daarna sneller dan de bovenstaande methoden. Het vereist wel dat er voor elke classificatie een nieuwe fine-tuning wordt uitgevoerd. Bovendien is er nood aan voldoende voorbeelden. Het vereiste aantal voorbeelden is afhankelijk van het aantal klassen, de mate waarin de klassen gemakkelijk kunnen worden onderscheiden en de gewenste kwaliteit. In de praktijk kan dit variëren van enkele tientallen tot honderden voorbeelden per klasse.

Het proces kan worden versneld door van een document een fragment (bijvoorbeeld enkel de eerste en laatste 100 tekens) te nemen: deze fragmenten bevatten doorgaans de meest significante elementen. Door enkel het begin van een document voor classificatie te gebruiken, wordt de prompt veel compacter en de kans op geheugenproblemen vermeden.

Finetuning

Er werd een voorbeeld uitgewerkt met de scripts **classification_train.py** en **classification_apply_trained.py**. Met dit voorbeeld kunnen teksten (bijvoorbeeld e-mails) worden ingedeeld in twee klassen: 'formeel' of 'informeel'.

In een eerste stap wordt met `classification_train` een model gefinetuned.

- Deze *finetuning* gebeurt op basis van een CSV-bestand met manueel geclassificeerde tekstfragmenten. In dit CSV-bestand bevinden zich twee kolommen:
 - (1) `text_short`: de geclassificeerde teksten
 - (2) `label`: de naam van de klasse.

In het script wordt gebruikgemaakt van het CSV-bestand **training_data.csv** (de naam en het pad van de training-file kan in het script worden aangepast.) De dataset wordt verdeeld in drie delen: een stuk om te trainen, een stuk om tussendoor te controleren (validatie), en een stuk om achteraf te testen. Vervolgens worden de teksten omgezet in de vorm die het taalmodel kan begrijpen (*tokenization*). Gebruik bij voorkeur '*balanced*' trainingsets, waarbij het aantal voorbeelden per klasse even groot is.

- Een bestaand Nederlands BERT-model wordt geladen en ingesteld om teksten te classificeren. Het model leert van de trainingsset en wordt na elke ronde gecontroleerd met de validatieset. Hiervoor wordt gebruik gemaakt van het model `GroNLP/bert-base-dutch-cased`, een model dat getraind is op de Nederlandse taal. Alternatieven zijn onder meer:
 - `bert-base-multilingual-cased` of `xlm-roberta-base`: geschikt voor meertalige content
 - `distilbert-base-multilingual-cased`: kleiner en lichter model

Settings kunnen in het script worden aangepast om het verder te optimaliseren:

Parameter	Betekenis
<code>model_name</code>	gebruikt model

use_cpu	zet op 'True' wanneer er geen (of een te kleine) GPU beschikbaar is.
max_length	maximum aantal tokens dat voor de training wordt gebruikt (hoe korter, hoe sneller)
batch_size (train/eval)	aantal voorbeelden dat tegelijk wordt verwerkt (meer voorbeelden vereist meer werkgeheugen)
num_train_epochs	aantal training rondes: hoe meer hoe beter, maar ook hoe langer het trainen duurt. Bij te hoog aantal is er risico op overfitting. 4-5 maximum aanbevolen.
metric_for_best_model	Gebruik "accuracy", maar vervang door 'f1' als de klassen niet in balans zijn.

- Na de training wordt gemeten hoe goed het model presteert. Het beste model wordt bewaard als './bert-base-dutch-cased_finetuned'

Vervolgens wordt het script `classification_apply_trained.py` gebruikt om een set teksten te classificeren:

- Gegevens worden opgehaald uit de tekstendatabase. In het voorbeeld worden alle teksten opgehaald van het type 'berichtbestand' (e-mails), met een lengte van minstens 100 tekens.
- Vervolgens wordt het gefinetunde model geladen en worden de teksten geclassificeerd. De classificaties worden weggeschreven naar de tekstendatabase in de vorm van een verrijking.

Het creëren van een algemeen classificatiemodel voor tekstgenres blijkt voor het MODAL-project niet erg zinvol of haalbaar te zijn. Dit komt door de volgende factoren:

- Het is lastig om een lijst van genres op te stellen die relevant en nuttig is voor alle betrokken organisaties.
- Er is onvoldoende geschikt trainingsmateriaal beschikbaar om een robuust classificatiemodel te ontwikkelen dat alle organisaties kan bedienen.
- Het nut van een dergelijke algemene toepassing is vaak beperkt, omdat de inhoud van documenten doorgaans belangrijker is dan de vorm.

5.3. Topic detection

Het doel van *topic detection* (onderwerpsherkenning) is het vaststellen van het onderwerp van een tekst door middel van een of enkele trefwoorden (topics). Verschillende benaderingen werden onderzocht:

- Het gebruik van een generiek LLM (o.a. microsoft/Phi-3.5-mini-instruct, google/gemma3): dit genereert per document een of meerdere trefwoorden. Er wordt daarbij geen rekening gehouden met andere documenten, dus deze methode is geen topic detection in de strikte betekenis. Deze methode is relatief traag, waardoor het best is enkel de X eerste tekens te verwerken. Voorbeeld: **topic_from_llm_gemma.py**: dit script analyseert de eerste 1.000 tekens en geeft één of meerdere onderwerpen in json-formaat.
- Topic modelling: een *topic model* is een statistisch model voor het ontdekken van abstracte "topics" die voorkomen in een verzameling documenten. Topicmodellering is een veelgebruikte tool om

door middel van tekstmining verborgen semantische structuren in tekstcorpora te ontdekken. Aan de hand van *topics* kunnen teksten geclusterd worden. Mogelijkheden zijn:

- Topic modeling met LDA (Latent Dirichlet Allocation (LDA)): zeer snel, maar weinig relevante woorden.
- Topic Modeling met [BERTopic](#)¹⁵ is zeer snel en genereert topics in de vorm van een set trefwoorden ('topic_name'). Vervolgens wordt elk document aan een topic toegekend.
- Bij [Guided Topic Modeling](#) wordt gebruikgemaakt van een aantal vooraf bepaalde *topic representations* (trefwoorden die bij een bepaald thema horen). Deze sturen het topicmodelleringsproces bij, waardoor de resultaten dichter liggen bij de verwachte topics. BERTopic kan naast bestaande topics nog steeds eigen topics voorstellen - wat een sterk voordeel is t.o.v. classificatie. Deze benadering vereist voorkennis van een archief. Het is een mogelijk alternatief voor classificatie.

Omdat het gebruik van een generiek LLM (te) veel resources vraagt, werd het scenario met topicmodellering verder uitgewerkt. Het script **topicdetection_bertopic.py** voert topicmodellering uit op basis van BERTopic:

- Selectie documenten:
 - Selecteert documenten uit de tekstendatabase met specifieke bestandstypen (PDF, Word, etc.) en met minimaal 100 woorden
 - Verwijdt stopwoorden en getallen uit de teksten
 - Beperkt tekstlengte tot maximaal 10.000 karakters
- Onderwerpsdetectie:
 - Bepaalt automatisch het optimale aantal onderwerpen (tussen 10-100)
 - Gebruikt BERTopic voor het identificeren van topics in de documenten met behulp van een meertalig BERT-model. Deze topics bestaan uit een set trefwoorden.
- Labeling:
 - Genereert labels voor elk geïdentificeerd topic. Hiervoor wordt gebruik gemaakt van een functie **topic_representation2subject** (zie verder), waarbij OpenAI wordt gebruikt om een label te genereren, bijvoorbeeld:
['insecten', 'vogels', 'zoogdieren', 'muizen', 'soorten', 'soort', 'invertebraten', 'lieveheersbeestjes', 'dieren', 'poelkikker'] ⇒ 'Dieren en biodiversiteit.'
 - Slaat de resultaten op in een tijdelijk CSV-bestand
- Database Updates verrijkt de oorspronkelijke databaserecords met:
 - Geïdentificeerde onderwerpen
 - Onderwerplabels
 - Waarschijnlijkheidsscores
 - Datum van verrijking

Topic representations zijn een set van keywords die representatief zijn voor het topic. De naam van het topic wordt meestal gevormd door de eerste X keywords. Het script **topic_representation2subject.py** vormt dit

¹⁵ Zie uitgewerkt voorbeeld op

<https://medium.com/@pauloyc/topic-modelling-and-multiclass-text-classification-using-transformers-6fc3ee69c15d>

met behulp van een LLM om tot een meer compacte representatie van het topic, in één of enkele woorden. bijvoorbeeld:

['insecten', 'vogels', 'zoogdieren', 'muizen', 'soorten', 'soort', 'dieren', 'poelkikker']
⇒ 'Dieren en biodiversiteit'

Hiervoor wordt een extern model gebruikt (OpenAI). Een vergelijkbare oplossing met Mistral geeft minder goede resultaten.

Conclusie:

- Topic modeling, waarbij meerdere documenten tegelijk worden geanalyseerd, lijkt meer zinvolle resultaten te geven, omdat de context van de documenten kan worden gebruikt. Bovendien is deze aanpak in staat om documenten te clusteren.
- Optioneel kan er gebruikgemaakt worden van guided topic modeling.
- Het creëren van onderwerpterm op basis van de trefwoorden die bij een topic horen, wordt voor het Nederlands best uitgevoerd met behulp van een online service. Het veiligheidsrisico is veel kleiner, omdat er enkel onderwerptermen worden gepubliceerd. [uitgevoerd met productiedata van VUJO-archief].

5.4. Summarization

Het script **summarizer.py** maakt korte, Nederlandse samenvattingen van tekstbestanden.

1. Het laadt google/gemma-3-1b-it, dat goed is in het begrijpen en genereren van meertalige tekst.
2. Het script doorzoekt de tekstendatabase naar tekstbestanden (zoals Word, PDF, e-mails, etc.) die nog geen samenvatting hebben en een bepaalde lengte hebben. Uit deze documenten worden de eerste X (bijvoorbeeld 1000) tekens geselecteerd.
3. Het formuleert een duidelijke opdracht voor het AI-model: "Vat deze tekst samen in het Nederlands" en stuurt deze opdracht samen met de tekst naar het AI-model.
4. De samenvatting, datum en de naam van het gebruikte AI-model, worden opgeslagen in de database bij het originele document.

Een alternatief (niet uitgewerkt) is een *Metadata-based summary*, waarbij een LLM op basis van enkele metadata-elementen een beschrijving maakt, bv.

- teksttype
- bestandsnaam
- naam van bovenliggende folder
- NE-resultaten
- Topicmodelingresultaten

5.5. Conclusies bij verrijking

Er werden verschillende benaderingen uitgetest om teksten uit documenten door middel van generatieve AI verder te verrijken. De resultaten waren in vele gevallen bruikbaar binnen het kader van de use case ("inzicht

verkrijgen in de inhoud en structuur van een ongeordend digitaal archief"), maar vaak van onvoldoende kwaliteit om werkelijk betrouwbare resultaten op te leveren.

De oorzaak van de beperkte performantie ligt eerder in de beperkingen van de gebruikte hardware en daaruit voortkomende beperking in het gebruik van bepaalde modellen.

Bij de nabespreking bleken vooral de verrijkingen met NER, Topic detection en Summarization als meest zinvolle ervaren te worden.

6. Bevragen

Bevragen betreft een aantal gebruiksscenario's waarbij gericht werd gezocht naar

- (1) realistische antwoorden die een gebruiker of archivaris kan stellen documenten (RAG).
- (2) documenten waarin deze antwoorden te vinden zijn (search, semantic search).

6.1. RAG

Retrieval-Augmented Generation (RAG) is een AI-techniek waarbij een groot taalmodel (LLM) wordt gekoppeld aan een externe kennisbank, die bestaat uit een aantal tekstdocumenten. Op deze manier kunnen gebruikers genieten van betere en meer actuele antwoorden.

6.1.1. Omgevingsanalyse

Verschillende RAG-frameworks en applicaties werden beoordeeld:

- [Ragflow](#).
- [LightRAG](#): zie ook <https://learnopencv.com/lightrag/> een out-of-the-box toepassing
- [txtai](#): AI framework for semantic search, LLM orchestration and language model workflows.
- [Weaviate](#)
- [LLMWare](#)
- [unstructured](#): framework voor voorbereiding, geen RAG UI etc
- [mmore](#)
- Voor WARC files [WARCgpt](#)
- [GPT4All](#)

Behalve bovenstaande offline oplossingen zijn er aanbieders van **RAG as a service** oplossingen, bijvoorbeeld [CustomGPT.a](#), [groundx](#) (EyeLevel), [Pryon](#) of GraphRAG.

Omdat de opdracht was beperkt tot het gebruik van gegevens in een strikte offline omgeving, zijn deze online diensten niet verder onderzocht. Wanneer blijkt dat deze dienstverleners voldoende garanties bieden qua beveiliging en eigenaarschap van data, kan het gebruik van een dergelijke service oplossing overwogen worden.

6.1.2. GPT4All

GPT4All is een veel gebruikt, out-of-the-box RAG met GUI en API. Documentatie is beschikbaar op <https://docs.gpt4all.io/index.html>. Er is keuze uit verschillende modellen (in principe kunnen alle modellen in gguf-formaat worden gebruikt), maar vele gguf-geformateerde modellen vereisen bijkomende configuratie.

GPT4All kan worden voorzien van meerdere documentensets waaruit de antwoorden worden gegenereerd (txt, pdf, docx). Deze tekstdocumenten werden opgehaald uit de tekstendatabase en als .txt-bestand opgeladen in GPT4All.

(script `file_output_from_textdb.py`)

- script `/MODAL_RAG_gpt4all/create_collection.py` voor creëren van lokale collectie

- verschillende modellen werden getest:
 - deepseek r1 distill llama 8b
 - deepseek r1 distill qwen 7b
 - deepseek r1 distill qwen 14b
 - llama 3 8b instruct
 - paraphrase-multilingual-mpnet-base-v2

6.1.3. Analyse en resultatenevaluatie

- De CP's bezorgden een aantal reële vragen, met daarbij het verwachte antwoord en mogelijk het document waarin het antwoord te vinden is.
- Zie [2025 MODAL RAG evaluatie](#) voor een overzicht.
- deepseek r1 distill llama 8b levert de meest aanvaardbare resultaten.
- De meeste modellen blijken betere resultaten op te leveren wanneer de vraagstelling in het Engels is (zie voorbeelden). Mogelijk omdat dan betere fragmenten worden opgezocht.

6.2. Search

Met de MongoDBCompass-applicatie is het mogelijk te zoeken op strings. Daarbij kunnen waarden en velden met elkaar gecombineerd worden via AND, OR en NOT.

Enkele voorbeelden:

Minstens 1 van beide namen komt voor in NER_persons ('OR')

```
{"enrichments.NER_persons":  
  {$all: ["Wouter Beke", "Sven Gatz"]} }  
}
```

Beide namen komen voor in NER_persons ('AND')

```
{"enrichments.NER_persons":  
  {$all: ["Wouter Beke", "Sven Gatz"]} }  
}
```

Beide namen komen voor in NER_persons en het document is een gegevensbestand ('AND')

```
{$and:  
  [{"enrichments.NER_persons":  
    {$all: ["Wouter Beke", "Sven Gatz"]} },  
  {generic_file_type: "gegevensbestand"}]}
```

Gebruik van Reguliere expressie: W. Beke of Wouter Beke, hoofd- of kleine letters

```
{extracted_text: RegExp("(w\\.|outer) beke", "i") }
```

6.3. Semantic search

Voor semantisch zoeken werd een app ontwikkeld (**search_semantic.py**) met een grafische zoekomgeving. Deze app maakt gebruik van de embeddings om verwante documenten op te zoeken.

- gebruiker geeft een query, bestaande uit een of meer zoektermen
- maak embedding van de query
- zoek beste match met documentembeddings

Deze benadering maakt meertalig zoeken mogelijk en ondersteunt natuurlijke taalgebruik.

Voorbeeld:

Query:

```
"wleke documenten zijn er in verband met minderheden in Azië?" [sic]
```

Best match:

```
[...]
```

Beste mensen,

Op het secretariaat kwam net een oproep toe vanwege een aantal Tibetaanse verenigingen. Zij organiseren op 10 maart, de verjaardag van de nationale opstand in Lhasa van '59, een betoging voor een vrij Tibet.

Concreet worden de zware schendingen van de mensenrechten, van de vrijheid van eredienst en van de burger- en politieke rechten van het Tibetaanse volk aangeklaagd. Daarom wordt de stopzetting gevraagd van de illegale bezetting van het land door de Chinese Volksrepubliek.

```
[...]
```

De app blijkt minder geschikt voor zoeken op exacte woorden en eigennamen; betere resultaten door een begrip te omschrijven, bv. "standpunten in verband met Vlaamse onafhankelijkheid en autonomie"

6.4. Conclusie bij zoekscenario's

Zoeken met MongoDB Compass werkt efficiënt en snel, maar is enkel geschikt voor zoekvragen met een exacte match. Het semantisch zoeken op basis van embeddings bleek mogelijk maar nog weinig effectief. Deze strategieën om een archief doorzoekbaar te maken bleken de verwachtingen niet in te lossen, voornamelijk omdat er met zeer omvangrijke, ongestructureerde archieven werd gewerkt, en omwille van de intrinsieke beperkingen van RAG en semantisch zoeken. Verdere optimalisering (dedupliceren, verwijderen van betekenisloze documenten of fragmenten zoals code) en krachtigere modellen kunnen hierin mogelijk verbetering brengen.

7. Verkennen

Verkennen betreft gebruiksscenario's waarbij de archivaris wordt ondersteund in het analyseren van de inhoud en de structuur van een digitaal archief. Om dit te ondersteunen werden verschillende strategieën onderzocht:

- (1) overzichten van identieke documenten of varianten, bijvoorbeeld voor het uitzuiveren van duplicaten
- (2) overzichten van identieke documenten of varianten, zodat onderzoek kan worden gedaan naar tekstgenese
- (3) relaties tussen analoog en digitaal archief opsporen
- (4) browser: de inhoud van een *directory* als geheel verkennen
- (5) visualisaties van gegevens die in reeksen met elkaar worden gecombineerd

7.1. Text similarity

Voor het meten van tekstsimilariteit wordt gebruikgemaakt van embeddings. Deze methode is in principe geschikt om varianten op te sporen op basis van betekenis in plaats van woordvorm - theoretisch kunnen hiermee ook verschillende taalvarianten worden opgespoord. Alternatieve methoden (zoals [MinHash](#)) kunnen enkel variaties op woordvorm herkennen.

Het script **report_similarities_json.py** meet de similariteit tussen documenten op basis van de embeddings. Vectoren worden ingelezen uit de tekstendatabase en *cosine similarity* wordt berekend voor elk document ten opzichte van alle andere documenten. Hierbij kan een drempelwaarde (threshold) worden ingesteld, (bijvoorbeeld 0,98).

Resultaten worden weggeschreven in een json-document (/data/similarities), bijvoorbeeld:

```
"67c5b126e5bb4b014f2450fb": {
  "id": "67c5b126e5bb4b014f2450fb",
  "file_path": "/ADVN/VEA260_VUJO/VEA260_01/ZIP-100/E-MAIL/VOEREN98.DOC",
  "similar_documents": [{
    "id": "67c5b1d1e5bb4b014f248633",
    "file_path":
"/ADVN/VEA260_VUJO/VEA260_04/ZIP-100/Volksuniejongeren/Vredesfietseling/Voerenroute
2001-Limburg.doc",
    "similarity_score": 0.9807061656830269
  ]
}
```

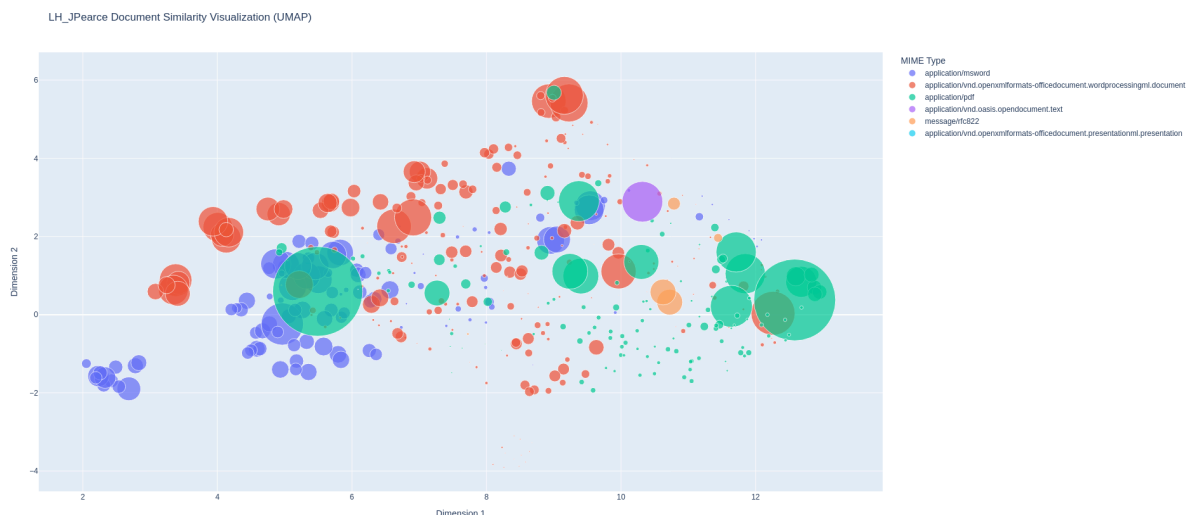
Dit voorbeeld geeft aan dat het document VOEREN98.DOC voor meer dan 98% verwant is met het document Voerenroute 2001-Limburg.doc.

Clustering en Visualisering

Documenten kunnen op basis van similariteit worden geclusterd en clusters kunnen grafisch worden gevisualiseerd. Zie scripts:

- **visualize_versions_tSNE.py** (sneller voor kleinere sets)
- **visualize_versions_UMAP.py** (vermoedelijk beter)

Deze scripts clusteren tekstdocumenten (word, openoffice, wordperfect, pdf e.d.)



Deze methode blijkt vooral goed te werken voor kleinere archieven: bij grotere documentencollecties worden de clusters al snel onoverzichtelijk. Het clusteren is vooral zinvol bij literaire archieven, waar het opsporen van meerdere versies van een document bijvoorbeeld inzage kan geven in het schrijfproces. Voor onderzoek naar tekstgenese kan een vergelijkbare aanpak worden genomen, waarbij verwante documenten in een chronologische volgorde worden geplaatst.

Het visualiseren geeft een indicatie van het bestaan van clusters, maar mist nog een duidelijke verwijzing naar de documenten zelf. Een alternatieve benadering is het clusteren op basis van verrijgingsgegevens, bijvoorbeeld op basis van gedetecteerde topics en NER.

7.2. Deduplication

Bij deduplicatie is het doel het opsporen van inhoudelijk identieke bestanden. Hierbij kan dezelfde aanpak genomen worden als bij Similarity: wanneer de similariteit 100% is, is er sprake van een (inhoudelijk) duplicaat.

Opmerkingen:

- deze methode is in staat (quasi) duplicaten te vinden op basis van de inhoud, onafhankelijk van het formaat (txt, retrieved from, docx...)
- exacte duplicaten kunnen worden opgespoord met een hash-functie, bijvoorbeeld MD5

7.3. Relatie tussen analoog en digitaal archief

Het doel van dit gebruiksscenario is te onderzoeken hoe een digitaal document kan worden ondergebracht in een onderdeel van het geïnventariseerde analoog archief. Om dit te realiseren werd de volgende strategie gevolgd:

- beschouw de rubrieken van het geïnventariseerde analoog archief als klassen, waarin digitale documenten worden ondergebracht
- creëer embeddings van de hoofdrubrieken in een archiefinventaris door deze te vectoriseren met hetzelfde model als waarmee de embeddings van de digitale documenten werden gecreëerd ([zie hoger](#))
- zoek voor elk digitaal document de rubriek waarvan de embeddings het best overeenkomen

Deze strategie leverde echter onvoldoende kwaliteitsvolle resultaten op. Dit kan worden verklaard door:

- (1) Inconsistentie in rubrieken: een archief bevat bijvoorbeeld een rubriek met financiële documenten, en een aantal rubrieken per regionale afdeling, met daarin telkens een subrubriek financiën/administratie-gerelateerde documenten. De gebruikte methode zal het document steeds bij de hoofdrubriek 'Financiën' onderbrengen.
- (2) Namen van rubrieken zijn doorgaans te kort en of te weinig betekenisvol om een voldoende rijke embedding te creëren. Een meer betrouwbare methode zou erin bestaan een sample van de analoge documenten te digitaliseren en daarvan embeddings te creëren.

7.4. File browser

Doel: bladeren in hiërarchie van het archief, waarbij inhoud van bestanden en folders wordt getoond aan de hand van (samengevatte) tags etc.

Bestaat uit twee stappen:

1. Creëer hiërarchie met samenvatting en schrijf weg naar de tekstendatabase.
2. Creëer op basis van *record* en '*summary record*' een doorbladerbaar html-bestand.

Verschillende verrijkingen kunnen worden samengevoegd:

- topics
- NER
- summaries

Het script **summarize_records_to_db.py**:

- haalt records op uit de tekstendatabase die verwijzen naar een bestand
- creëert records die de folder in de hiërarchie representeren
- leest van *all records* die bestanden representeren de verrijkingen (NER...) en voegt deze samen op het niveau van de bovenliggende folder (top 20 meest voorkomende NER_personen, etc). Deze gegevens worden toegevoegd aan het record dat de bovenliggende folder representeert
- schrijft de resultaten terug naar de tekstendatabase
- dit proces wordt herhaald tot de top van de folderhiërarchie bereikt is

Het script: **summarize_summaries_to_db.py**:

- maakt samenvattingen van samenvattingen van documenten (gecreëerd met het [summarization script](#))
- schrijft de resultaten terug naar de tekstendatabase
- herhaalt dit met nieuw gecreëerde samenvattingen tot de top van de hiërarchie is bereikt

Het script **report_summaries_to_html.py** tenslotte haalt de records op uit de tekstendatabase en genereert een overzicht in HTML-formaat van de bestanden en mappen, inclusief metadata (zoals NER-data). Het bootst hiermee de hiërarchische structuur van het archief na, waarbij extra informatie over elk bestand of map in een samenvatting wordt weergegeven. Het resultaat is een interactieve HTML-pagina waarmee je de structuur en bijhorende details kunt bekijken door deze open en toe te klappen.

File and Folder Structure ADVN_VEA260_VUJO

```
VEA260_VUJO
Summary: De tekst beschrijft een reeks initiatieven gericht op jongeren in Vlaanderen, met focus op onderwijs, migratie en de rol van jongeren in de samenleving.
Topic Label:
Path: media/henk/LaCie/2025_MODAL/ADVN/VEA260_VUJO
VEA260_01
Summary: De Volksuniejongeren en VUJO organiseren activiteiten gericht op jongeren, met de focus op inclusie en het bevorderen van digitale vaardigheden.
Topic Label:
Path: media/henk/LaCie/2025_MODAL/ADVN/VEA260_VUJO/VEA260_01
VEA260_02
Summary: De VUJO-JD-WEEKEND onderzoekt Vlaamse gemeentelijke politiek, armoedebestrijding en jongerenuitdagingen.
Topic Label:
Path: media/henk/LaCie/2025_MODAL/ADVN/VEA260_VUJO/VEA260_02
VEA260_03
Summary: De campagne voor de VUJO Jongeren is gebaseerd op een selectieprocedure waarbij kandidaat en arrondissement de beslissing bepalen.
Topic Label:
Path: media/henk/LaCie/2025_MODAL/ADVN/VEA260_VUJO/VEA260_03
VEA260_04
Summary: VUJO is een politieke jongerenorganisatie die zich richt op participatie, maatschappelijke vorming en het bevrijdingsnationalisme, met een focus op lokale gemeen
Topic Label:
Path: media/henk/LaCie/2025_MODAL/ADVN/VEA260_VUJO/VEA260_04
V
NER Persons: Geert De Winter, Bart Volk, Klaas Van Audenhove, Robin Ramakers, Bart Devits, Nelly Maes, Kristin De Winter, Natalie Schoonbaert, Geert Conaerts, Bart
Vandenbroucke, Karl Vermaercke, Wouter Buyt, Jan Roegiers, Herman Lauwers
NER Organisations: FUJI Magnetics GmbH, VUJO, V, Volksuniejongeren, Europese Unie, Volksunie, Vlaamse Regering, Vlaams Parlement, VU, VO, VLD, SP, Toekomst
Jongeren van de Toekomst
V
NER Locations: Brussel, Europa, Vlaanderen, Gent, Antwerpen, Leuven, Diksmuide, Barricadenplein, Mechelen, Voeren, België, Frankrijk, Toekomst, Barricadenplein, No
NER Miscellaneous: Vlaamse, Vlaams, V, Wij - Jongeren, Belgische, StatusbalkTekst, Vredesfetseling, Frans, StatusbalkTeks, Agenda, IJzerbedevaart, StatusbalkTekst
Topic Representation:
Summary: De tekst onderzoekt de initiatieven rondom jongerenactiviteiten en politiek in Vlaanderen, met name via de Jeugdherberg Den Drossaard en Volksunie
Topic Label:
Path: media/henk/LaCie/2025_MODAL/ADVN/VEA260_VUJO/VEA260_06/ZIP-100
DschijfKaat 2de deel
Summary: De tekst beschrijft een reeks initiatieven rondom jongerenactiviteiten en politiek in Vlaanderen, met name via de Jeugdherberg Den Drossaard
mobiliteit, en de uitdagingen van de jeugd, met een focus op de rol van VUJO en de verschillende politieke bewegingen. De Volksunie jongeren zijn krit
Topic Label:
Path: media/henk/LaCie/2025_MODAL/ADVN/VEA260_VUJO/VEA260_06/ZIP-100/DschijfKaat 2de deel
JOINT
Summary: De tekst beschrijft een reeks internationale initiatieven en activiteiten van IFLRY, met een focus op de internationale zendingen en de
seminarie en de zendingen hebben een totale investering van 43.414,- euro. * **Verbieden:** Er zijn verschillende zendingen, waaronder
Topic Label:
Path: media/henk/LaCie/2025_MODAL/ADVN/VEA260_VUJO/VEA260_06/ZIP-100/DschijfKaat 2de deel/JOINT
DEELNAME IFLRY SEMINARIE BERLIJN 20.doc
```

7.5. Time series

Doel: toon grafisch het verband tussen kenmerken (bijvoorbeeld NER_locaties) en tijd, om het verloop van het kenmerk over de jaren heen te kunnen zien.

Het script **report_graph_by_year_enrichments.py**:

- haalt documenten op die verrijkt zijn met bepaalde kenmerken (bijvoorbeeld persoonsnamen, locaties, etc.)
- kent een jaartal toe
- telt op hoe vaak deze kenmerken per jaar voorkomen.

Vervolgens wordt een selectie toegepast met volgende opdracht:

- filter items die minder dan een minimum aantal keer voorkomen (standaard 10 keer)
- bereken voor elk jaar het relatieve aandeel (percentage) van elk item

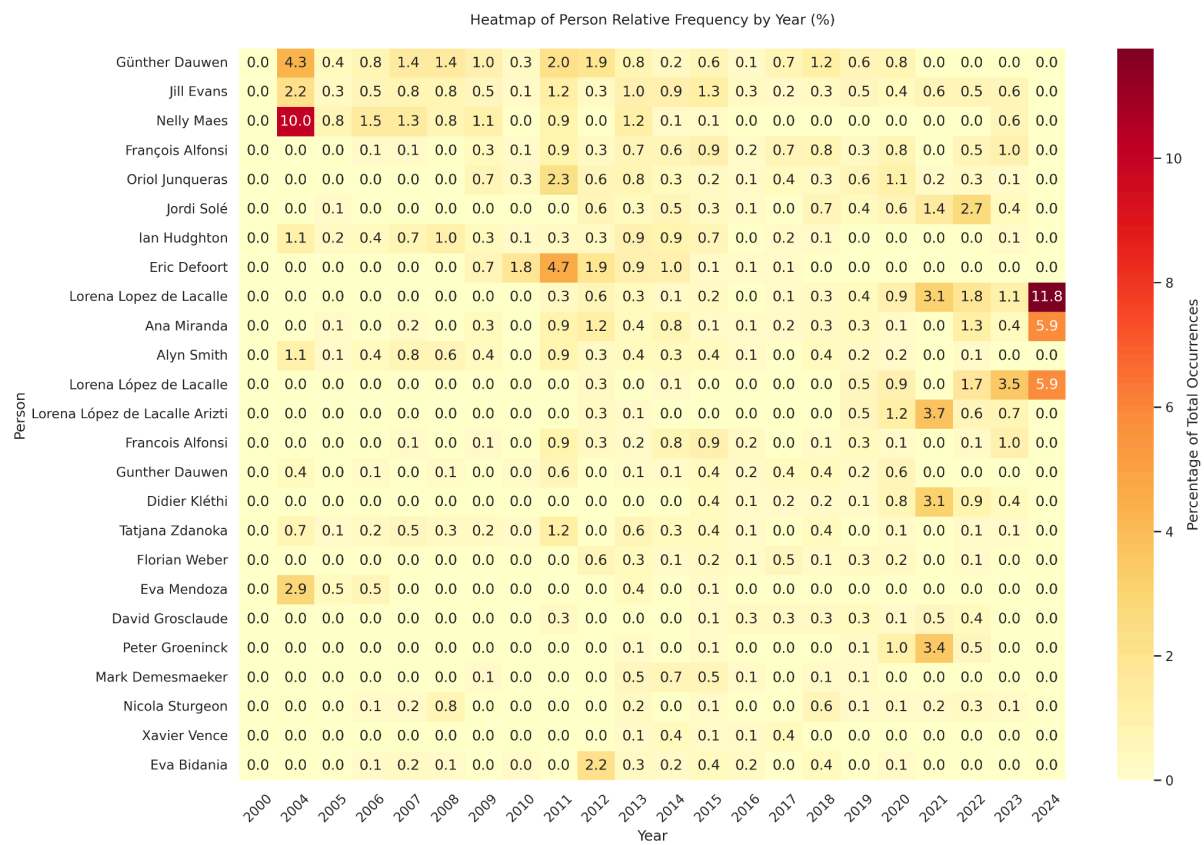
- selecteer de meest voorkomende items (default: top 50)

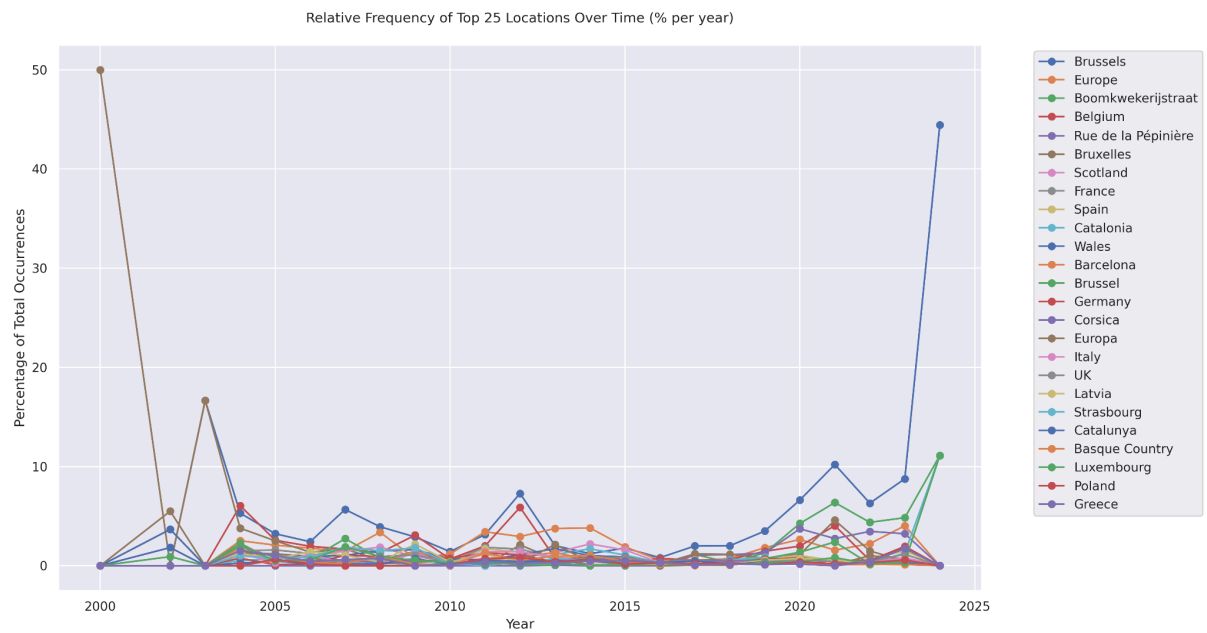
Hiervan wordt een visualisatie gecreëerd:

- maak een lijngrafiek die laat zien hoe de relatieve frequentie van items over de jaren verandert
- genereer een heatmap die de percentages per jaar in kleur weergeeft
- sla de grafieken op als PNG-bestanden
- exporteer de data naar CSV-bestanden (zowel absolute aantallen als percentages)
- toon statistieken over totalen en verdelingen

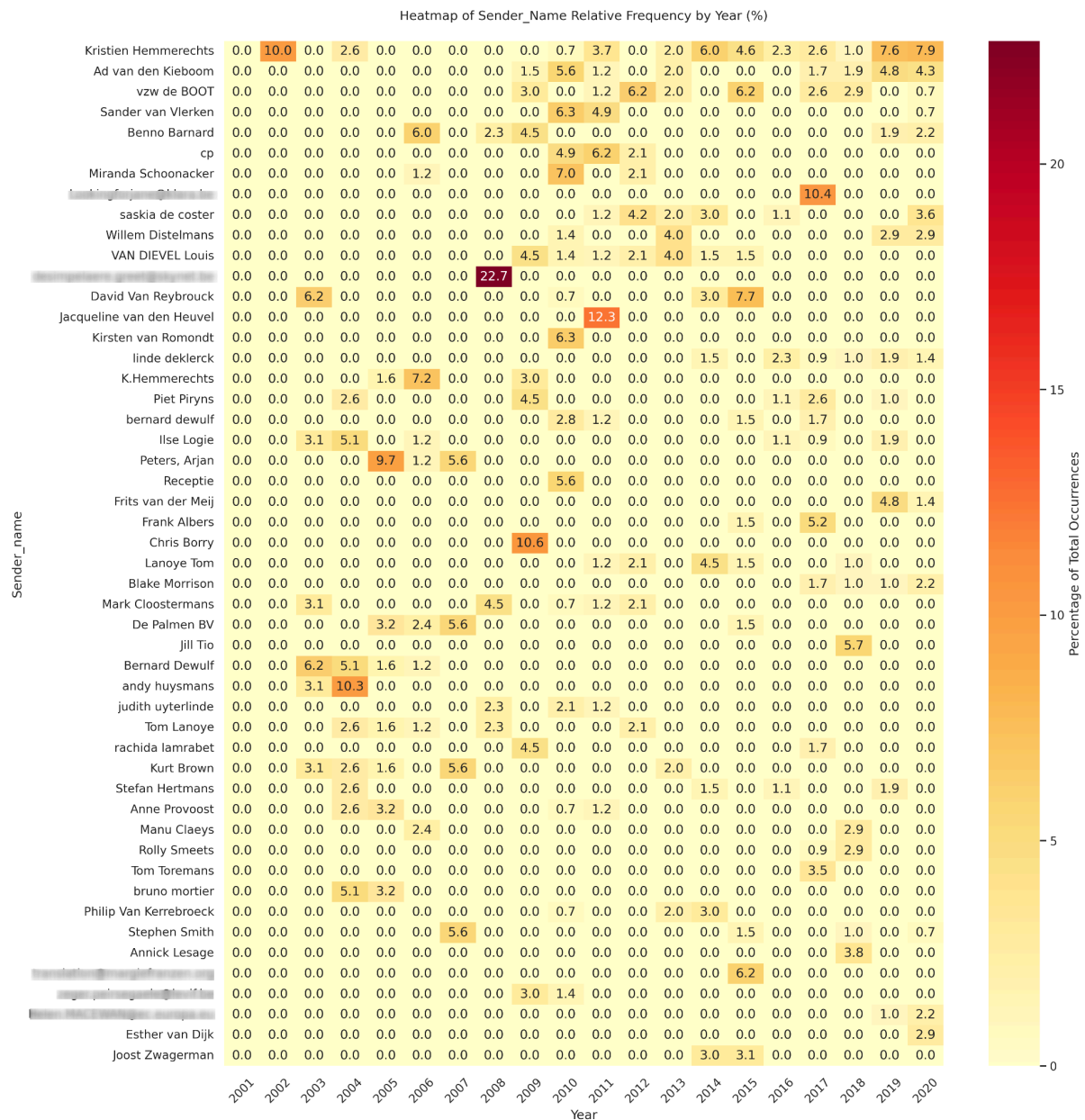
Door de parameter aan te passen kun je verschillende soorten verrijkingen analyseren (personen, locaties, organisaties) uit de tekstendatabase.

Voorbeeld:





Het script **report_graph_by_year_correspondents.py** is vergelijkbaar met het vorige script, maar toont e-mailafzenders of -ontvangers, bijvoorbeeld afzenders van een archiefvormer:



Optimalisaties zijn mogelijk door:

- (dynamische) labels toe te voegen
- interactief te maken, bijvoorbeeld mogelijkheid naam uit te lichten
- varianten te maken met lijstweergave van namen, met de documenten waarin deze voorkomen

7.6. Conclusie bij Verkennen-scenario's

Scenario's voor verkennen sluiten nauwer aan bij de doelstelling van het MODAL-project, namelijk een inzicht verwerven in de inhoud van een digitaal archief. Hiertoe werden verschillende manieren van visualisering onderzocht. Het bleek niet eenvoudig een geschikte visualiseringsmethode te vinden die grote aantallen documenten inzichtelijk kan maken.. De meest bevredigende oplossing was de filebrowser.

8. Algemene conclusie

In de technische uitvoering van het MODAL-project werd de infrastructuur opgezet, software ontwikkeld en data verwerkt om inzicht te krijgen in de inhoud en structuur van ongeordende digitale archieven.

De infrastructuur is opgezet met open source componenten en LLM's, waarbij alle verwerking offline gebeurt en beperkt is tot *consumer hardware*. Na overleg met data-eigenaars en een eerste analyse zijn vier archieven geselecteerd voor verdere verwerking: VUJO (ADV), J[...]X (AMSAB), auteursarchief (LH) en het audioarchief van MoMu. De overdracht van bestanden gebeurde offline vanwege veiligheidsredenen en omvang. WARC-bestanden en andere sociale mediabestanden werden niet verder geanalyseerd vanwege complexiteit en gebrek aan documentatie.

De voorverwerkingsfase omvatte tekstextractie met Apache Tika, inclusief OCR voor gescande PDF's. De geëxtraheerde teksten en metadata werden gestandaardiseerd opgeslagen in een MongoDB-tekstendatabase. Het omzetten van gesproken audio naar tekst met Whisper bleek effectief, hoewel de herkenning van persoonsnamen semi-automatische correctie vereiste. Tekstextractie en opslag in de database vereenvoudigen de AI-verwerking en leveren belangrijke inzichten op, zoals bestandstypen, datering en correspondenten.

Verschillende AI-verrijkingstechnieken werden getest, waaronder Named Entity Recognition (NER), classificatie, onderwerpsdetectie en samenvattingen. Hoewel de resultaten bruikbaar waren voor het verkrijgen van inzicht, was de kwaliteit vaak onvoldoende voor werkelijk betrouwbare resultaten, voornamelijk door hardwarebeperkingen en de daaruit voortvloeiende beperkingen in modelkeuze. NER, onderwerpsdetectie en samenvattingen werden als het meest zinvol ervaren.

Het project onderzocht het bevragen van documenten met Retrieval-Augmented Generation (RAG) en zoekfunctionaliteiten. GPT4All werd gebruikt als out-of-the-box RAG-oplossing. Zoeken met MongoDB Compass en semantisch zoeken met embeddings bleken mogelijk maar nog weinig effectief. Deze strategieën om een archief doorzoekbaar te maken bleken de verwachtingen niet in te lossen, voornamelijk omdat er met zeer omvangrijke, ongestructureerde archieven werd gewerkt, en omwille van de intrinsieke beperkingen van RAG en semantisch zoeken.

Een meer bruikbare strategie werd gevonden in de vorm van het visualiseren van de inhoud van het archief op basis van de geëxtraheerde gegevens en de verrijkingen die door middel van AI werden toegevoegd. Met name de browser, waarin de inhoud van een archief op elk hiërarchisch niveau (directory en subdirectories) werd samengevat, bleek een zinvolle benadering.

Over het algemeen heeft het MODAL-project methoden en inzichten opgeleverd voor het verwerken en analyseren van digitale archieven met behulp van AI. De bevindingen benadrukken het potentieel van AI voor archiefverkenning, maar wijzen ook op de noodzaak van verdere optimalisatie, vooral met betrekking tot de kwaliteit van verrijkingen en de performantie van modellen op beperkte hardware.

9. Addenda

9.1. Schema tekstendatabase

IDENTIFICATIE

_id	record identifier
file_name	originele bestandsnaam
file_path	pad naar origineel bestand
file_mimetype	mime-type
generic_file_type	generiek bestandstype, via afzonderlijk script gegenereerd

CREATIE

creation_date	creatiedatum (embedded)
file_creation_date	creatiedatum (file system)
estimated_creation_date	geschatte creatiedatum
creator	maker (embedded)
recipient_email	ontvanger van het e-mailbericht (e-mail adres)
recipient_name	ontvanger van het e-mailbericht (naam)
sender_email	verzender van het e-mailbericht (e-mail adres)
sender_name	verzender van het e-mailbericht (naam)
cc_name	naam in CC

TEKSTEIGENSCHAPPEN

tika_parser	Tika component die voor extractie werd gebruikt
extracted_text	geëxtraheerde tekst
word_count	aantal woorden
language	taal

VERRIJKINGEN

enrichments	(elke verrijking wordt geclusterd in een element)
enrichment_date	datum van de verrijking
model_used	model gebruikt voor verrijking
NER_locations	gedetecteerde locaties
NER_miscellaneous	gedetecteerde entiteiten (overige)
NER_organisations	gedetecteerde organisaties
NER_persons	gedetecteerde personen

summary	samenvatting
Topic_label	toegekende naam van de Topic
Topic_name	label van de topic
Topic_probability	waarschijnlijkheid van de topic
Topic_representation	woorden die de topic representeren
EMBEDDINGS	
embeddings	(elke embedding wordt geclusterd in een element)
embedding_model_used	model gebruikt voor embeddings
text_embeddings	embeddings (vector)

9.2. Workflow toepassing scripts

Het volgende voorbeeld van *workflow* geeft de volgorde van de te gebruiken scripts om een filebrowser te creëren aan.

fase	script
process folder	process_folder.py
add S2T	speech_to_text.py
add dates	add_estimated_creation_date.py
add correspondents	tika_extract_correspondents.py
add embeddings	create_embeddings.py
add summaries	summarizer.py
add topics	topicdetection_bertopic.py
add NER	ner_wikineural.py
add classifications	classification_train.py en classification_apply_trained.py
create hierarchy	create_folder_hierarchy.py
summarize metadata (topics, NER):	summarize_records_to_db.py
summarize summaries	summarize_summaries_to_db.py
create HTML	report_summaries_to_html.py

De scripts zijn gepubliceerd op Github in drie repositories:

- [MODAL_textdatabase](#): ingest
- [MODAL_enrichments](#): verrijkingen met behulp van AI
- [MODAL_reporting](#): representatie van gegevens

9.3. Gated Models

Sommige modellen vereisen instemming met gebruikseisen vooraleer ze kunnen worden toegepast/gedownload.

Dit is bijvoorbeeld het geval bij Huggingface, waar een *access token* moet worden gebruikt. Dit moet alleen worden opgegeven bij het eerste gebruik (d.w.z. wanneer het model wordt gedownload). Access tokens worden aangemaakt via de Huggingface [gebruikersaccount](#). In het script wordt de parameter 'token' gebruikt om het model bij eerste gebruik te kunnen downloaden, bijvoorbeeld:

```
# Load the tokenizer and model
token = "hf_XXX"
tokenizer = AutoTokenizer.from_pretrained(model_name, token=token)
model = AutoModelForCausalLM.from_pretrained(model_name, token=token)
```

Na het eerste gebruik van het script kunnen de tokenparameters worden verwijderd.