Afstudeerscriptie

Hoe kan het algoritme voor documentanalyse verbeterd worden zodat deze de algehele structuur van een document beter herkent?

Naam:	Mathieu Goedhart
Studentnummer:	469901
Afstudeerbegeleider:	Manasse Siekmans
Periode:	13-02-2023 t/m 07-07-2023
Bedrijfsbegeleiders:	Jesper Enzerink & Sven Ordelman
Versie:	4.0
Laatste aanpassing:	16-06-2023
Onderwijsinstelling:	Saxion University of Applied Sciences

Versiebeheer

Versie	Verandering	Datum
1.0	Initiële opzet	07-03-2023
2.0	De hoofdstukken "Ontwerp" en "Realisatie" zijn toegevoegd en de ontvangen feedback is verwerkt.	20-05-2023
3.0	Reflectie en conclusie toegevoegd. Feedback verwerkt.	04-06-2023
4.0	Uitdagingen van de implementatie omschreven. Reflectie uitgebreid. Hoofdstuk testen uitgebreid.	16-06-2023

Begrippenlijst

Begrip	Definitie
Overfitting	Een machine learning model dat goede resultaten behaalt op de trainingsdata maar slechte resultaten behaalt met nieuwe data (Biswal, 2021).
Machine learning bias	Een machine learning model dat bevooroordeelde resultaten behaalt door aannames in het machine learning proces (Pratt, 2020).
Ground truth	De werkelijk accurate gegevens die door machine learning modellen worden gebruikt om patronen te herkennen (C3, 2021).
Fine tuning	Het optimaliseren van een bestaand model, zodat het geschikt is voor het oplossen van een nieuw probleem (Google Developers, n.d.).
Algoritme	Een set aan instructies om het uiteindelijke doel te bereiken (van Essen & van Weerdt, n.d.). Een externe API wordt ook beschouwd als een vorm van een algoritme
GEM	Een pakket/library dat kan worden gebruikt door een Ruby programma (rubygems, n.d.)

Abstract

Moneybird biedt bonnetjes- en factuurherkenning aan om het verwerken van bonnetjes en inkoopfacturen te vereenvoudigen. Veel velden, zoals de factuurregels, worden niet correct herkend. Bovendien worden de velden waarvoor wel een voorspelling wordt gedaan, in 71% van de gevallen verkeerd voorspeld, waardoor het proces van de ondernemer vertraagt wordt.

Het doel van dit onderzoek is om in een tijdsbestek van 5 maanden het huidige algoritme voor documentanalyse te verbeteren, zodat de ondernemer minder interacties nodig heeft om een inkoopfactuur of bonnetjes in te boeken. Hiervoor is de volgende onderzoeksvraag opgesteld: Hoe kan het algoritme voor documentanalyse verbeterd worden, zodat deze de algehele structuur van een document beter herkent? Met document wordt zowel een bonnetje als ook een inkoopfactuur bedoeld.

Om het antwoord op de hoofdvraag te vinden, is gebruikgemaakt van de onderzoeksfases van het HBO-I framework (HBO-I, n.d.). Het onderzoek bestaat uit drie fasen: voorbereiding en prototyping, benchmarking en realisatie. Elk van deze fasen maakt gebruik van specifieke onderzoekspatronen om uiteindelijk tot het eindproduct te komen.

In het onderzoek zijn de frameworks eerst theoretisch vergeleken en vervolgens is een benchmark opgesteld om de nauwkeurigheid van de algoritmes in een realistische omgeving te meten en onderling te vergelijken. Uit de resultaten is gebleken dat Microsoft Azure Form Recognizer zowel het meest nauwkeurig de onderdelen op de documenten herkent, als ook een kosteneffectieve oplossing is.

Na het onderzoek is het vernieuwde algoritme voor documentherkenning geïmplementeerd. Het maakt gebruik van een classificatiemodel om het documenttype vast te stellen en de Microsoft Azure Form Recognizer API is gebruikt als basis voor het ophalen van de voorspellingen. Het algoritme is beschikbaar gesteld als bèta-functie en zal geleidelijk worden uitgerold naar de klanten van Moneybird.

Het vernieuwde algoritme voorspelt de onderdelen op de documenten nauwkeuriger dan het oorspronkelijke algoritme. Bovendien is het in staat om de documentregels te herkennen, wat een significante vermindering van het aantal benodigde interacties bij het boeken van een document oplevert in vergelijking met het oorspronkelijke algoritme.

Inhoudsopgave

1. Inleiding....8
1.1. Organisatieomschrijving....8
1.2. Probleemstelling....8
1.3. De opdracht....8
1.4. Hoofdvraag en deelvragen....9
2. Requirements en risico's....10
2.1. Requirements....10
2.2. Risico's project....11
3. Onderzoeksaanpak....12
3.1. APA-stijl richtlijnen....12
3.2. Onderzoeksmethode....12
4. Project management....14
4.1. Proces....14
4.2. Kanban....14
4.2.1. Het werk visualiseren....14
4.2.2. Beperk de work in progress....14
4.2.3. Behoud de flow....14
4.3. Workflow....15
5. Onderzoeksresultaten....16
5.1. Hoe is het Moneybird-systeem voor documentherkenning momenteel technisch opgebouwd en waar zitten de knelpunten?....16
5.1.1. Het proces van documentanalyse....16
5.1.2. De knelpunten....17
5.2. Welk documentonderdeel levert de meeste winst op qua interacties?....18
5.2.1. Inleiding....18
5.2.2. Methode....18
5.2.3. Resultaten....19
5.2.4. Conclusie....20
5.3. Welke algoritmen voor documentherkenning zijn er, en welke documentonderdelen zijn hierin te herkennen?....21
5.3.1. Inleiding....21
5.3.2. Herkende document onderdelen....21
5.3.3. Beschikbare frameworks voor documentherkenning....21
5.3.4. Conclusie....23
5.4. Hoe kan een fine-tuned machine learning model gebruikt worden voor documentherkenning?....23
5.4.1. Inleiding....23
5.4.2. Dataset....24
5.4.3. Voorbereiding dataset....24
5.4.4. Beperkingen in taal....24
5.4.5. Resultaten....25
5.4.6. Conclusie....26
5.5. Welk algoritme heeft de hoogste nauwkeurigheid bij het herkennen van specifieke

kenmerken op facturen en bonnetjes? 26
5.5.1. Inleiding 26
5.5.2. Benchmark meetwaarden 26
5.5.3. Resultaten 27
5.5.4. Conclusie 28
6. Ontwerp 29
6.1. Inleiding 29
6.2. Overzicht Moneybird 2 (MB2) structuur 29
6.3. Voortbouwen op het bestaande systeem 30
6.3.1. Béta functie 30
6.4. Classificatie 30
6.5. Flow 31
6.6. Opslag 32
6.6.1. Database 32
6.7. Foutafhandeling 33
7. Realisatie 34
7.1. Inleiding 34
7.2. Risico's 34
7.3. Document classificatie 34
7.3.1. Training 34
7.3.2. Implementatie 35
7.4. Voorspelling (Form Recognizer) 36
7.5. Data transformer 37
7.6. Mappen en tonen van de data in MB2 38
7.6.1. Data mappen met predict mutaties 38
7.6.2. De data tonen 40
7.6.3. Suggesties ongedaan maken 41
7.7. Uitdagingen 41
7.8. Eindresultaat 42
8. Testen 43
8.1. Inleiding 43
8.2. Classificatie 43
8.3. Voorspelling (Form Recognizer API) 43
8.4. Mappen van de voorspellingen 44
8.5. Mappen van voorspellingen met betrekking tot belasting 45
9. Conclusie 46
10. Aanbevelingen 47
10.1. Voorspelling op de achtergrond uitvoeren 47
10.2. Ontwikkelingen in AI gebied 47
10.3. Ingebruikname 47
11. Beperkingen 48
12. Reflectie 49
13. Bronnenlijst 51
14. Bijlagen 54

Figuren- en tabellenlijst

Figuren

Figuur 1.	APA-richtlijnen voor kopjes	11
Figuur 2.	Activity diagram huidige proces document analyse	15
Figuur 3.	Velden die herkent worden	16
Figuur 4.	Clickmap factuur inboeken	18
Figuur 5.	Clickmap contact toevoegen	19
Figuur 6.	Benchmark machine learning frameworks	23
Figuur 7.	Frameworks en hun accuraatheid	27
Figuur 8.	Overzicht van de Moneybird structuur	28
Figuur 9.	Application flow	30
Figuur 10.	Database diagram implementatie	31
Figuur 11.	Waarschuwing niet of met onvoldoende zekerheid herkent veld	32
Figuur 12.	Sequence diagram bewerken afbeelding voor classificatie	34
Figuur 13.	Overzicht componenten Moneybird 2 - document analyse	37
Figuur 14.	Activity diagram voorspelling belastingtarief	38

Tabellen

Tabel 1.	Non-functional requirements	09
Tabel 2.	Functional requirements	09
Tabel 3.	Overzicht frameworks voor documentherkenning	21
Tabel 4.	Modellen en hun accuraatheid	24
Tabel 5.	Meetwaarde nauwkeurigheid	25
Tabel 6.	Document onderdelen en hun meetwaarde	26
Tabel 7.	Frameworks en hun accuraatheid per document onderdeel	27
Tabel 8.	Classificatie klassen en hun accuraatheid	33

1. Inleiding

1.1. Organisatieomschrijving

Moneybird is een software ontwikkelingsbedrijf dat online boekhoudsoftware aanbiedt aan voornamelijk zzp'ers, freelancers en mkb'ers. Het bedrijf is opgericht in 2008 en is gevestigd in Enschede. Het platform van Moneybird stelt gebruikers in staat om facturen op te stellen, offertes te maken, bonnetjes en facturen te uploaden, en hun boekhouding bij te houden. Het bedrijf streeft naar gebruiksvriendelijke software, zodat ook ondernemers zonder financiële achtergrond het kunnen begrijpen en gebruiken (Moneybird, n.d.).

Moneybird heeft momenteel ongeveer 50 medewerkers in dienst en heeft ondertussen al meer dan 250.000 aangesloten administraties. Daarnaast biedt Moneybird sinds 2022 betaalrekeningen en betaalpassen aan. Dit doet het bedrijf in samenwerking met het financiële platform Adyen. Door deze samenwerking kan Moneybird de betaalrekening naadloos integreren in zijn boekhoudsoftware zonder afhankelijk te zijn van traditionele banken (Moneybird, n.d.).

1.2. Probleemstelling

Een van de taken waar vrijwel elke ondernemer mee geconfronteerd wordt, is het verwerken van bonnetjes en inkoopfacturen in de boekhouding. Om dit proces te vereenvoudigen, biedt Moneybird een handige functie genaamd bonnetjes- en factuurherkenning. Met behulp van deze functie wordt geprobeerd zoveel mogelijk attributen op documenten automatisch te herkennen met behulp van reguliere expressies (regex) en optical character recognition (OCR). De factuurherkenning is zowel beschikbaar als IOS- en Android-app en via de website. De bonnetjes scanner is alleen beschikbaar als mobiele app.

De huidige oplossing is echter verre van perfect. In gevallen waarop een factuur meerdere bedragen of datums bevat, wordt vaak het verkeerde attribuut gepakt. Als gevolg hiervan wordt in 71% van de gevallen achteraf de voorspelling aangepast (zie bijlage A). Dit leidt niet tot versnelling, maar juist tot vertraging van het proces.

Ook worden in de huidige oplossing de factuurregels niet herkend, terwijl het aardig wat tijd kost om de velden handmatig in te voeren. De factuurregel omvat een beschrijving van het product of de dienst, de bijbehorende hoeveelheid, de kosten en het btw-tarief.

1.3. De opdracht

Gelukkig zijn er verschillende manieren om het probleem aan te pakken. Naast OCR kunnen meer geavanceerde oplossingen worden gebruikt, zoals modellen voor machine learning. Deze geven vaak betere resultaten dan een OCR gebaseerde oplossing (Kim et al, 2021). Deze modellen hoeven niet altijd zelf getraind te worden aangezien er veel bestaande oplossingen beschikbaar zijn (Lee, 2022).

Naast een verbeterd model moet rekening gehouden worden met de verschillen tussen bonnetjes en facturen. Facturen bevatten factuurregels die elk een ander belastingtarief

kunnen hebben. Bovendien is de betaaldatum een belangrijk gegeven, aangezien facturen vaak nog betaald moeten worden. Aan de andere kant zijn bonnetjes meestal al betaald en bevatten hetzelfde btw-tarief voor alle regels (Moneybird, 2023).

Kortom, om het aantal interacties te verminderen en de tijd die een ondernemer nodig heeft om documenten in te voeren, is het belangrijk om onderzoek te doen naar het meest nauwkeurige model voor documentherkenning. Het is essentieel om rekening te houden met de verschillen tussen bonnetjes en facturen om een goed resultaat te behalen.

Om een duidelijk en meetbaar doel aan de opdracht te geven, is de volgende doelstelling opgesteld:

Het doel van de afstudeeropdracht is om in 5 maanden het documentanalyse algoritme zo veel mogelijk te verbeteren zodat de gebruiker minder interacties nodig heeft om een document in te boeken.

1.4. Hoofdvraag en deelvragen

De hoofdvraag van het onderzoek luidt:

Hoe kan het algoritme voor documentanalyse verbeterd worden zodat deze de algehele structuur van een document beter herkent?

De hoofdvraag wordt beantwoord met de volgende deelvragen:

1. Hoe is het Moneybird-systeem voor documentherkenning momenteel technisch opgebouwd en waar zitten de knelpunten?
2. Welke onderdelen van een documentstructuur zijn belangrijk om herkend te worden in Moneybird?
3. Welk documentonderdeel levert de meeste winst op qua interacties?
4. Welke algoritmen voor documentherkenning zijn er en welke documentonderdelen zijn hierin te herkennen?
5. Hoe kan een fine-tuned machine learning model gebruikt worden voor documentherkenning?
6. Welk algoritme heeft de hoogste nauwkeurigheid bij het herkennen van specifieke kenmerken op facturen en bonnetjes?

2. Requirements en risico's

2.1. Requirements

Dit hoofdstuk beschrijft de requirements voor het project, die zijn onderverdeeld in functionele (F) en niet-functionele (NF) requirements. Om elke requirement te prioriteren wordt de MOSCOW-classificatie gebruikt. De classificatie omvat must have (M), should have (S), could have (C) en won't (W).

De requirements zijn opgesteld op basis van de omschrijving van de opdracht en berichten over het onderwerp in Slack (intern communicatieprogramma). De requirements zijn vervolgens besproken en verfijnd na een gesprek met de bedrijfsbegeleiders (zie bijlage B).

Tabel 1

Non-functional requirements

Nummer	MoSCow	Requirement
NF1	C	Het geselecteerde model moet trainbaar zijn
NF2	M	Het algoritme verwerkt alleen gegevens op servers in Europese landen
NF3	M	De implementatie integreert met de bestaande infrastructuur van Moneybird
NF4	M	De geschreven code voldoet aan de kwaliteitseisen van Hogeschool Saxion en Moneybird
NF5	M	Het algoritme moet goed gedocumenteerd zijn

Tabel 2

Functional requirements

Nummer	MoSCow	Requirement
F1	M	Het algoritme ondersteunt factuurregels
F2	M	Het algoritme ondersteunt documenten in het formaat van bonnetjes en facturen
F3	M	Het algoritme ondersteunt documenten in de talen Nederlands en Engels
F4	M	Het algoritme ondersteunt verschillende valuta-eenheden

F5	M	Het algoritme herkent het factuurnummer
F6	M	Het algoritme herkent de vervaldatum op facturen
F7	W	Het algoritme geeft de coördinaten van de herkende tekst in een document terug
F8	C	Het algoritme herkent het KVK-nummer
F9	S	Het algoritme herkent handgeschreven documenten
F10	M	Het algoritme moet consistent zijn in de manier waarop het data teruggeeft
F11	M	Het algoritme kan het btw-tarief (hoog, laag, vrijgesteld) per factuurregel herkennen
F12	S	Het algoritme kan onderscheid maken tussen Nederlandse btw-tarieven en tarieven binnen of buiten de EU
F13	W	Het algoritme is in staat om specifieke details van het contact te herkennen, zoals de bedrijfsnaam en het IBAN-nummer

2.2. Risico's project

Naast het opstellen van de requirements is er ook een risicoanalyse uitgevoerd. Tijdens de analyse zijn de volgende risico's geïdentificeerd:

1. Beperkte ondersteuning voor niet-Nederlandse documenten.
2. Mogelijkheid van het bouwen van iets onjuist.
3. Complexiteit van het project.
4. Afwezigheid van een bedrijfsbegeleider.

Zie bijlage C voor meer informatie over de risico's, inclusief hun oorzaken en mogelijke oplossingen.

3. Onderzoeksaanpak

In dit hoofdstuk wordt de opzet van het afstudeerproject beschreven inclusief de onderzoeksmethode en de middelen die worden ingezet tijdens het project. Verder wordt beschreven hoe de kwaliteit bewaard wordt.

3.1. APA-stijl richtlijnen

Om consistentie in het verslag te waarborgen, worden de APA versie 7 richtlijnen voor documentopmaak (American Psychological Association, 2019) als leidraad gevolgd. Deze richtlijnen bevatten specifieke aanwijzingen met betrekking tot onder andere koppen, lettertypen en tabellen.

Figuur 1

APA-richtlijnen voor kopjes (Fant Memorial Library, 2020)

Level	Format
1	**Centered, Bold, Title Case Heading** Text begins as a new paragraph.
2	**Flush Left, Bold, Title Case Heading** Text begins as a new paragraph.
3	***Flush Left, Bold Italic, Title Case Heading*** Text begins as a new paragraph.
4	**Indented, Bold, Title Case Heading, Ending With a Period.** Text begins on the same line and continues as a regular paragraph.
5	***Indented, Bold Italic, Title Case Heading, Ending With a Period.*** Text begins on the same line and continues as a regular paragraph.

Note. In title case, most words are capitalized (see Section 6.17).

3.2. Onderzoeksmethode

Voordat er onderzoek gedaan kan worden is het belangrijk om vast te stellen wat er onderzocht moet worden en hoe dit gebeurt. Om deze reden is besloten om het DOT-framework (Development Oriented Triangulation Framework) (HBO-I, n.d.) te gebruiken. Dit framework is specifiek gericht op het uitvoeren van onderzoek in het ICT-domein en biedt diverse onderzoekspatronen die als handige richtlijnen dienen tijdens het onderzoek.

Het onderzoek is verdeeld in drie fasen, waarbij elk van de fasen gebruik maakt van een ander onderzoekspatroon. Hieronder wordt elke fase verder toegelicht:

Voorbereiding & prototyping
In de eerste fase van het onderzoek wordt het 'choose fitting technology'-patroon (HBO-I, 2021) gebruikt om technologieën te selecteren voor het verbeteren van het documentanalyse-algoritme. De strategieën omvatten het verzamelen van informatie via

literatuur, websites en andere bronnen (library-strategie), het vaststellen van criteria voor technologische selectie (field-strategie) en het testen van geselecteerde technologieën via prototyping (workshop-strategie). De resultaten worden gebruikt in de volgende fase om de nauwkeurigheid van het algoritme in een realistische setting te meten.

Benchmarking
In de vorige fase zijn op basis van onderzoek en prototyping drie technologieën gekozen. Deze technologieën worden nu getest op bruikbaarheid en nauwkeurigheid met behulp van het onderzoekspatroon 'validate' (HBO-I, 2021) Dit patroon omvat de strategieën 'lab' en 'showroom'.
Met de workshop strategie worden de doelstellingen van de tests bepaald en worden methoden, formules en technieken beschreven om de technologieën te testen. De gekozen methoden worden kort toegelicht aan de belanghebbenden. Indien nodig worden wijzigingen aangebracht.

Na de workshop strategie wordt de lab strategie toegepast om de nauwkeurigheid van de technologieën te meten. Hiermee kan aangetoond worden hoe goed het gekozen algoritme de structuur van documenten herkent. De resultaten van deze metingen worden beschreven en gepresenteerd aan de belanghebbenden.

Realisatie
Nu de beste technologie is geselecteerd, wordt het uiteindelijke eindproduct ontwikkeld. Dit eindproduct zal worden geïntegreerd met de bestaande infrastructuur van Moneybird en zal worden getest.
Voor deze fase wordt het onderzoekspatroon 'realise as an expert' (HBO-I, 2021) gebruikt. Dit patroon omvat de strategieën library, 'workshop' en 'lab'.

Met de library strategie wordt onderzocht hoe de onderzochte technologie kan worden geïntegreerd in de huidige systemen van Moneybird. Dit omvat een analyse van de huidige implementatie en het raadplegen van literatuur om de nodige informatie te verzamelen.

Vervolgens wordt de workshop strategie gebruikt voor het ontwikkelen van de uiteindelijke implementatie en het schrijven van tests. Nadat de implementatie gereed is, wordt de showroom strategie gebruikt om de implementatie te presenteren en krijgen belanghebbenden de kans om feedback te geven op de implementatie. Indien nodig worden er aanpassingen gemaakt. De resultaten worden beschreven in een verslag en opgeleverd als eindproduct.

4. Project management

4.1. Proces

Gedurende het project is gewerkt met iteraties van één week, waarbij er elke woensdag een afstudeerbespreking plaats heeft gevonden. Hierin is de voortgang van de betreffende week besproken en daarnaast de planning van de komende week besproken. Hierdoor was het zowel voor mij als voor de bedrijfsbegeleiders duidelijk wat de voortgang is en wat er verwacht kon worden. Op diezelfde dag is ook het afstudeermoment. Hierin vertellen alle afstudeerders in de vorm van een stand-up wat ze de afgelopen week gedaan hebben.

4.2. Kanban

Er is gekozen voor het Kanban Framework. Kanban heeft geen vereiste rollen en meetings en past daarom goed bij het afstudeertraject dat zelfstandig wordt uitgevoerd.

Kanban kent 4 kernprincipes:

- Het werk visualiseren
- Beperk de work in progress (W.I.P)
- Behoud de flow
- Monitor, wijzig en verbeter

4.2.1. Het werk visualiseren.

Kanban wordt gebruikt om het werk visueel weer te geven via een projectbord, dat wordt bijgehouden met behulp van Phabricator. Dit is een tool die door Moneybird wordt gebruikt voor projectmanagement. Op het projectbord worden de verschillende statussen van het werk weergegeven met behulp van de labels 'to-do', 'doing', 'in review' en 'done'.

4.2.2. Beperk de work in progress.

Binnen Kanban wordt een Work in Progress (W.I.P) limiet gehanteerd. Voor dit project is ervoor gekozen om dit limiet op 2 te zetten, wat betekent dat er maximaal aan twee tickets tegelijkertijd gewerkt kan worden. Hiervoor is gekozen om te voorkomen dat er aan te veel tickets tegelijk wordt gewerkt, wat tot een ongestructureerde manier van werken kan leiden.

4.2.3. Behoud de flow.

Om een goede flow te behouden wordt het Kanban bord ingericht met 3 fases. De fases zijn als volgt:

1. **To-do:** Een taak die is gepland maar nog niet gestart.
2. **Doing**: In deze fase worden de taken uitgevoerd. Er mogen twee taken tegelijk op 'doing' staan.
3. **In-review:** De taak staat klaar om te worden nagekeken door een collega van Moneybird.
4. **Done:** De taak is in de vorige fase goedgekeurd. Er kan nu een andere taak worden opgepakt.

4.3. Workflow

Voor het afstudeerproject is de workflow van Moneybird gebruikt. Deze workflow verschilt van de workflows zoals bekend van Github omdat er gebruik is gemaakt van arc. Het proces omvat de volgende stappen:

1. **Ticket oppakken:** Er wordt een ticket van het bord gepakt.
2. **Revision maken:** Bij het maken van revision wordt er eerst een lokale branch aangemaakt. Vervolgens wordt er een "revision" gemaakt waarin een beschrijving en een testplan worden opgenomen. Dit proces lijkt op het maken van een pull-request in Github.
3. **Feedback verwerken:** De feedback die de beoordelaar heeft gegeven wordt in deze stap verwerkt.
4. **Revision aanpassen:** Als er na het indienen van de revision nog wijzigingen zijn, moet er een nieuwe commit worden gemaakt in de lokale branch. Op basis hiervan wordt dan een nieuwe revision aangemaakt.

5. Onderzoeksresultaten

5.1. Hoe is het Moneybird-systeem voor documentherkenning momenteel technisch opgebouwd en waar zitten de knelpunten?

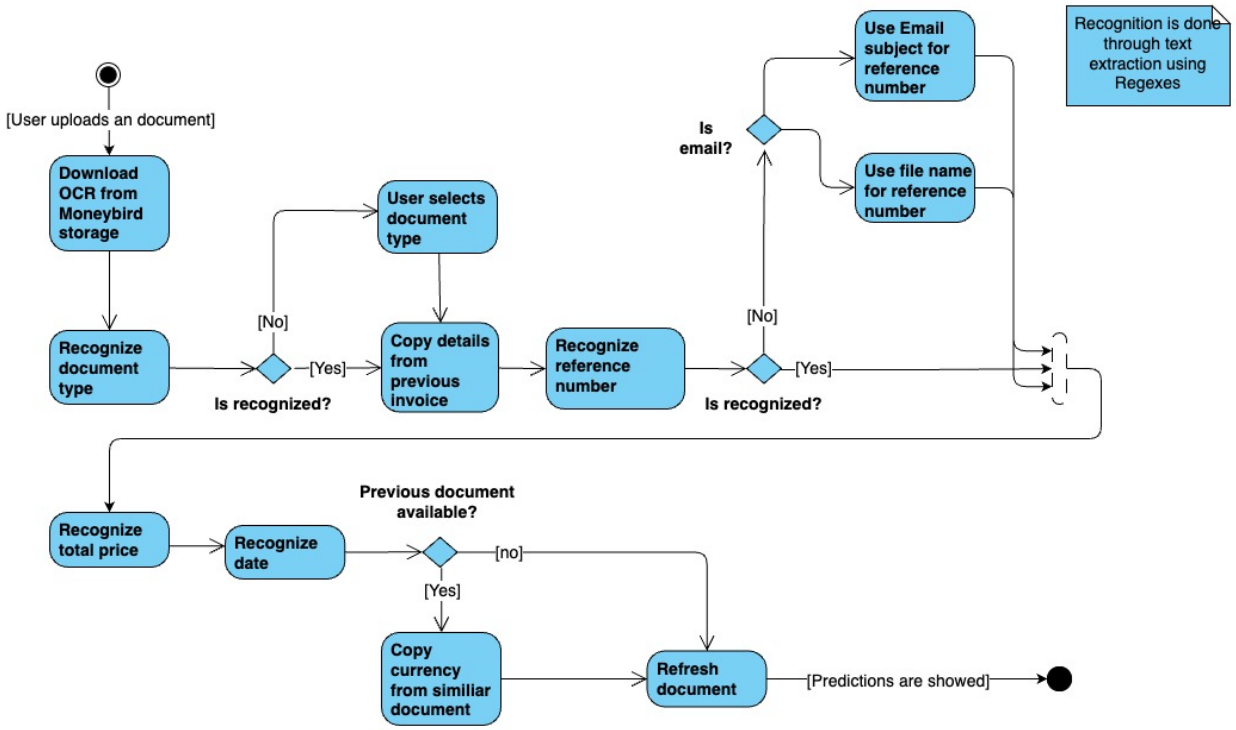
5.1.1. Het proces van documentanalyse

Het huidige proces van documentanalyse bestaat uit veel stappen. Eerst wordt de OCR-tekst gedownload van het *Moneybird storage (MBS)* systeem waar de afbeelding wordt geconverteerd naar tekst met *Google Vison*. Vervolgens wordt gezocht naar een bestaand document met vergelijkbare eigenschappen. Als een dergelijk document wordt gevonden, dan worden alle eigenschappen overgezet naar het nieuwe document.

Hierna wordt geprobeerd het referentienummer te identificeren met behulp van patroonherkenning. Indien de oorspronkelijke methode niet slaagt, zal het onderwerp van de e-mail worden gebruikt als het een e-mail betreft, en anders wordt de bestandsnaam gebruikt. Vervolgens wordt er een poging gedaan om de overige velden te herkennen.
Tot slot wordt opnieuw gecontroleerd of er een vergelijkbaar document beschikbaar is. Indien dat het geval is, worden de valuta en betalingstermijn uit het document gehaald.

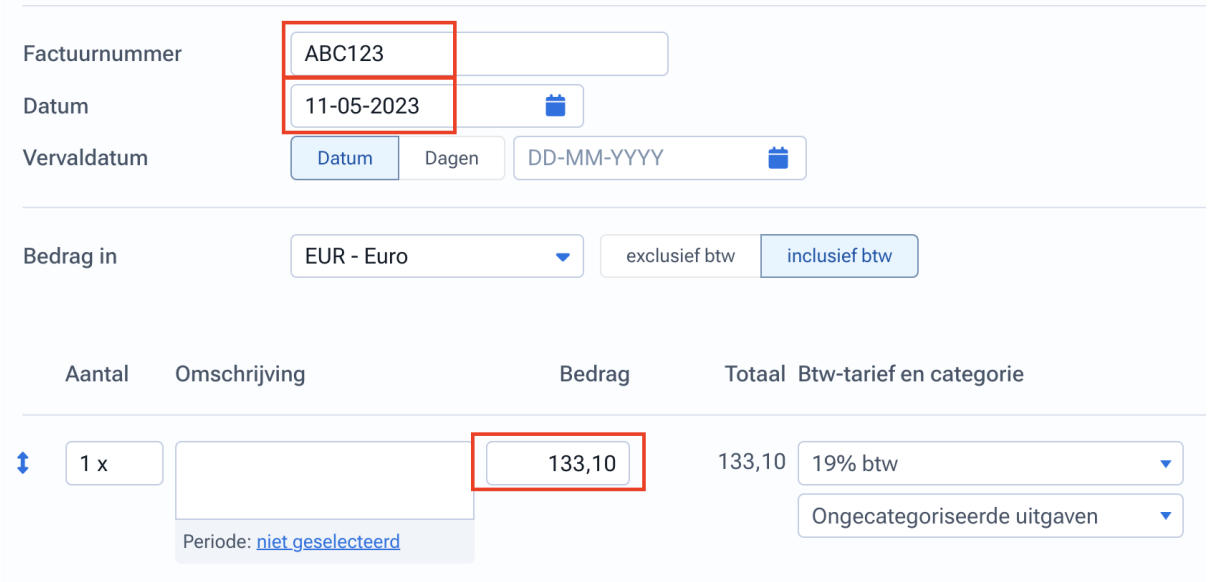
Figuur 2

Activity diagram huidige proces document analyse



Figuur 3

De velden waarvan een voorspelling wordt gemaakt zijn in het rood gemarkeerd



5.1.2. De knelpunten

De huidige oplossing vertrouwt op reguliere expressies, wat betekent dat het afhankelijk is van specifieke teksten in het document. Hierdoor kan het niet goed omgaan met uitzonderingen en afwijkende gevallen en verschillende talen (Dullin, n.d.).

Ook vinden gebruikers van Moneybird het niet fijn als gegevens van een bestaande, vergelijkbaar document worden gebruikt voor het nieuwe document (S. Ordelman, persoonlijke communicatie).

Indien het referentienummer niet gevonden kan worden, dan wordt teruggevallen op het gebruik van de bestandsnaam of het E-mail onderwerp als het een E-mail betreft. Indien een van beide niet met het referentienummer begint, wordt een onjuiste suggestie gegeven.

In alle gevallen moet de gebruiker handmatige aanpassingen doen, wat extra interactie vereist en de gebruiker uiteindelijk tijd kost.

Conclusie

Het huidige algoritme voor documentherkenning geeft te vaak een voorspelling die achteraf aangepast moet worden. Om dit op te vangen is ervoor gekozen om informatie van oude documenten (van hetzelfde contact) te gebruiken. Dit is iets wat gebruikers als niet fijn ervaren.

Het nieuwe systeem moet minder afhankelijk zijn van reguliere expressies om beter om te gaan met afwijkende gevallen. Om dit te bereiken, is er behoefte aan een schaalbare oplossing die naadloos kan integreren met de bestaande systemen van Moneybird en voldoet aan hoge standaarden op het gebied van codekwaliteit en testen.

5.2. Welk documentonderdeel levert de meeste winst op qua interacties?

5.2.1. Inleiding

Het doel van Moneybird is om het meest geautomatiseerde boekhoudpakket van Nederland te worden, waardoor gebruikers meer tijd overhouden voor de dingen die voor hen belangrijk zijn. Een verbeterde documentherkenning is een stap in die richting.

Op dit moment worden niet alle kenmerken op een factuur automatisch herkend. Velden die wel automatisch herkend worden, worden in 71 % van de gevallen achteraf gewijzigd (zie bijlage A). Het gevolg is dat de gebruiker meer interacties moet uitvoeren voor het inboeken van documenten en daardoor uiteindelijk meer tijd kwijt is.
Om het proces te kunnen versnellen is het belangrijk om vast te stellen voor welk document onderdeel veel interacties nodig zijn zodat vastgesteld kan worden waarin de meeste winst in interacties gewonnen kan worden.

5.2.2. Methode

Er is gebruikgemaakt van een clickmap, een vorm van heatmap, om te bepalen welke delen van een document veel interactie vereisen bij het inboeken. Uit onderzoek van Gordillo et al. (2014) is gebleken dat clickmaps tot de meest effectieve methoden behoren om inzicht te krijgen in de *user experience.* Ze beschrijven het als zeer effectief in het algehele proces van softwareontwikkeling.

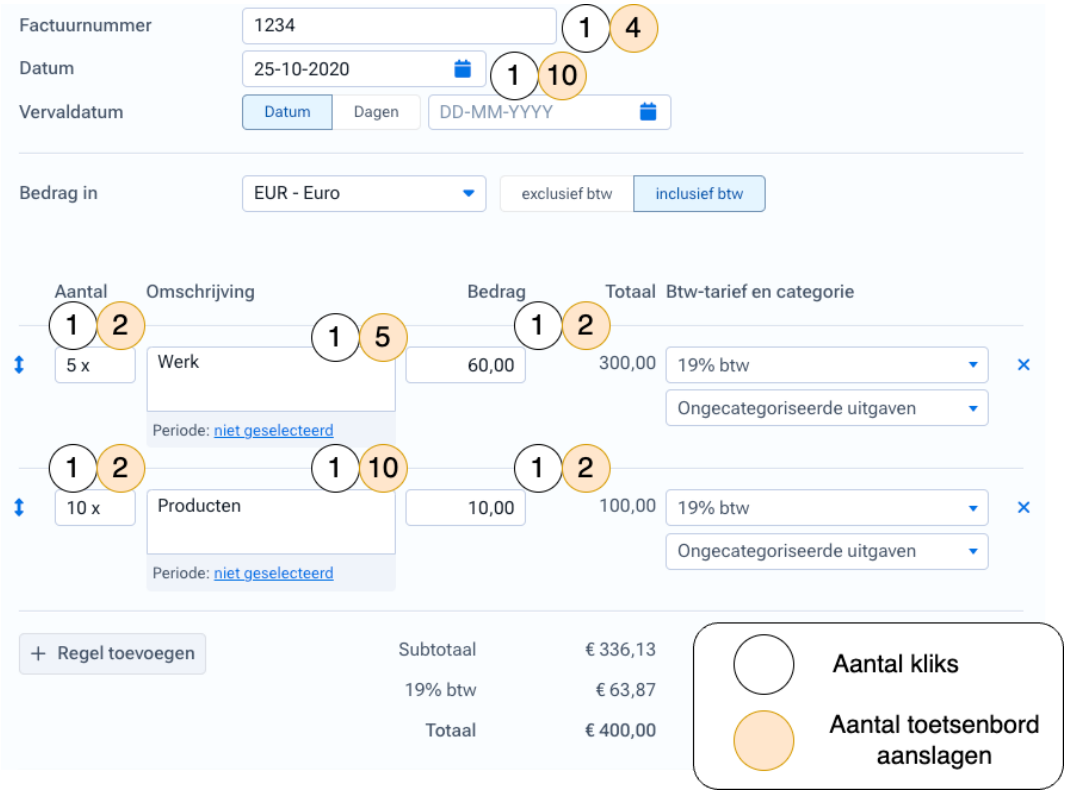
De test is uitgevoerd door een persoon buiten Moneybird, om een realistische setting te creëren. Tijdens de test houdt een script (zie bijlage D) bij waar de gebruiker klikt en hoe vaak dit gebeurt. Daarnaast registreert het, het aantal toetsenbordaanslagen van de gebruiker per veld. Op basis van deze informatie is een "click map" gemaakt om de interacties te visualiseren.

5.2.3. Resultaten

5.2.3.1. Factuur met bestaand contact.

Figuur 4

Aantal toetsenbordaanslagen en -kliks die nodig zijn om een factuur in te boeken



Het valt op dat het toevoegen van factuurregels veel interacties vereist. Elke factuurregel kan tot 4 velden bevatten, die allemaal moeten worden ingevuld. In het ergste geval moeten voor elke factuurregel alle 4 de velden worden ingevuld. Het veld "omschrijving" vereist de meeste toetsenbordaanslagen ten opzichte van andere onderdelen van de factuurregel.
Voor de velden 'factuurnummer' en 'datum' zijn minder interacties nodig in vergelijking met andere velden op de factuur.

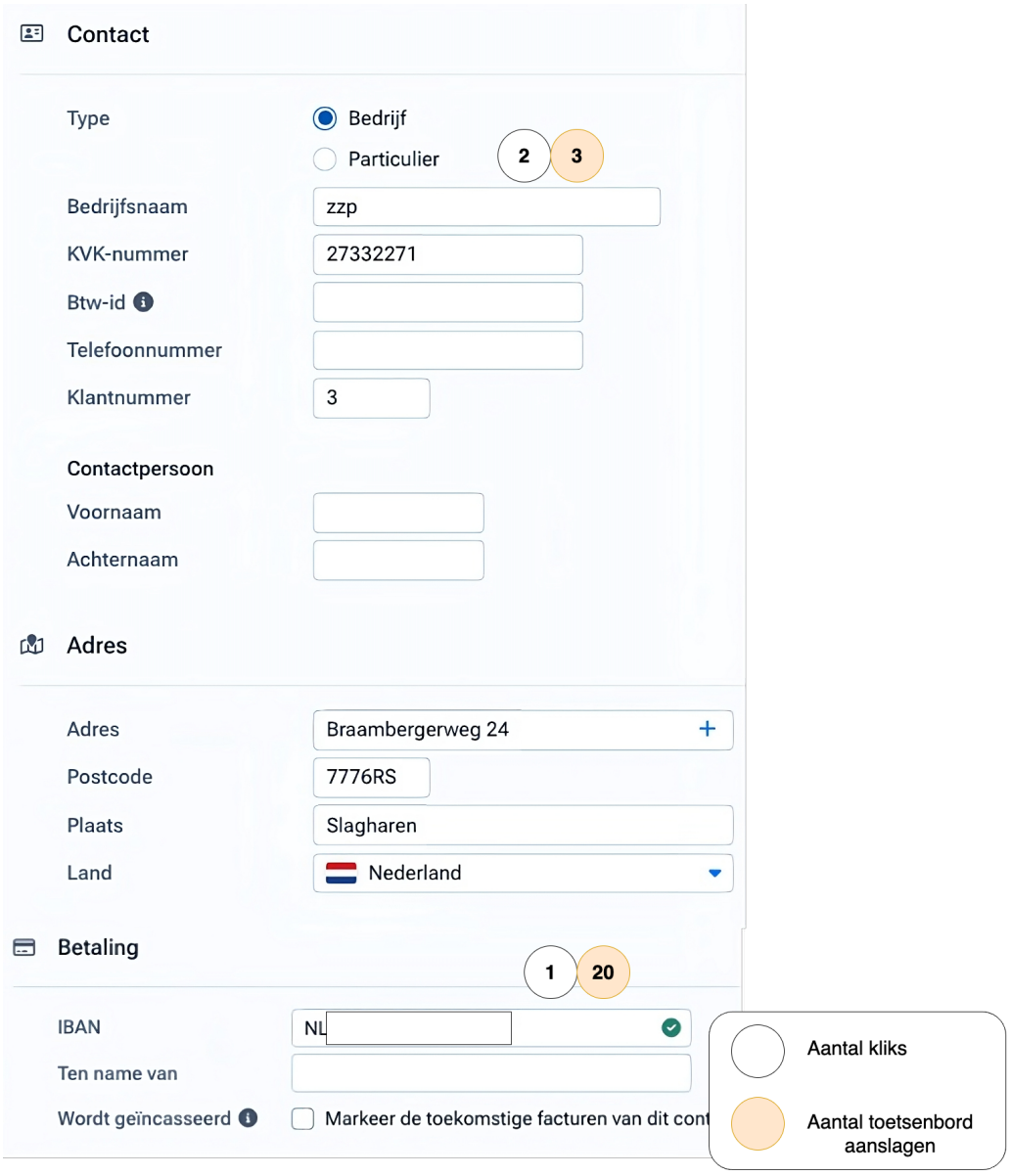
Kortom, de meeste winst qua interacties kan worden geboekt bij de factuurregels en de bijbehorende velden. Deze velden vereisen immers aanzienlijk meer interacties dan andere velden op de factuur.

5.2.3.2. Factuur onbekend contact.

Bij het verwerken van een factuur waarvoor het contact nog niet bekend is, moet er een nieuw contact worden aangemaakt. De meeste gegevens kunnen automatisch worden ingevuld vanuit het KvK-register.

Figuur 5

Aantal kliks en toetsenbord aanslagen die nodig zijn om een nieuw contact aan te maken



Dankzij het register van de Kamer van Koophandel (KVK) kan veel informatie eenvoudig worden verkregen op basis van de bedrijfsnaam, waardoor het proces minder muisklikken en toetsaanslagen vereist. Het invoeren van de bedrijfsnaam is de voornaamste interactie die nodig is. Alsmede het invoeren van het IBAN-nummer.

5.2.4. Conclusie

Verreweg de meeste interacties zijn nodig voor de factuurregels. Daarbij is het aantal interacties afhankelijk van het aantal factuurregels en de lengte van de omschrijving.
Bij het toevoegen van een nieuw contact kan veel informatie automatisch worden aangevuld op basis van gegevens van de Kamer van Koophandel. Meestal zijn alleen de naam van het bedrijf en het IBAN-nummer nodig om de interactie te voltooien.

Kortom, er kan veel winst in interacties geboekt worden als de factuurregels automatisch herkend worden. Het herkennen van de bedrijfsnaam en het IBAN-nummer bij het toevoegen van een contact zal tevens voor veel winst zorgen.

5.3. Welke algoritmen voor documentherkenning zijn er, en welke documentonderdelen zijn hierin te herkennen?

5.3.1. Inleiding

Amazon Textract, Microsoft Azure Form Recognizer en Google Document AI zijn populaire aanbieders van documentherkenning software die zijn getraind op datasets van onder andere facturen en bonnetjes (Heller, 2021).

GPT-3 (Generative Pre-trained Transformer 3) is daarentegen een erg populaire *generative AI* die verbluffende resultaten behaalt met een wijd scala aan use-cases (Mavuduru, 2021). Aangezien de GPT-3 API niet wordt gehost binnen de Europese Unie, voldoet deze niet aan de AVG-wetgeving met betrekking tot de verwerking van persoonsgegevens (Autoriteit Persoonsgegevens, n.d.). Om deze reden wordt deze toevoeging als experimenteel beschouwd.

5.3.2. Herkende document onderdelen

Alle onderzochte frameworks claimen dezelfde onderdelen op de facturen en bonnetjes te kunnen herkennen. Dit omvat alle belangrijke onderdelen zoals documentregels, contactgegevens en verdere basisinformatie.

Deze onderdelen kunnen worden samengevat in de volgende categorieën:

- **Document regels:** De omschrijving, prijs, datum, de hoeveelheid en het btw-percentage van de factuur/ bonnetjesregels.
- **Document informatie:** Het factuurnummer, de factuur / bonnetjes datum, het btw-bedrag en het totaalbedrag.
- **Contact informatie:** volledige adresgegevens (verkoper, ontvanger en leverancier)
- **Belasting informatie:** Belastingnummers
- **Overig:** Kortingsbedrag, valuta en Iban-nummers

Er zijn enkele kanttekeningen. Amazon Textract ondersteunt nog geen Nederlandse documenten (Amazon, n.d. -a). Hierdoor moet bij de implementatie *pattern matching* gebruikt worden. De herkende informatie is hierin grotendeels afhankelijk van de eigen implementatie.

5.3.3. Beschikbare frameworks voor documentherkenning

In het algemeen bieden alle bestudeerde frameworks vergelijkbare functionaliteiten als basis. Ze ondersteunen bijvoorbeeld gangbare bestandsformaten zoals pdf, jpg en png en bieden de mogelijkheid om zowel handgeschreven als getypte tekst te verwerken. Bovendien kunnen ze informatie lezen van zowel bonnetjes als facturen.

Toch zijn er enkele belangrijke verschillen. Deze verschillen zijn gebruikt om de frameworks met elkaar te vergelijken en worden hieronder verder toegelicht:

- **Integratie:** De mate waarin de bestaande oplossing integreert met de huidige infrastructuur van Moneybird.
    - **Score van 0:** geen aansluiting op de huidige infrastructuur.
    - **Score van 1:** al toegepast binnen de bestaande infrastructuur.
    - **Score van 2:** breed toegepast binnen de bestaande infrastructuur.
- **Kosten:** De kosten per 200.000 verwerkte documenten. Het aantal dat Moneybird ongeveer per maand verwerkt.
- **Out-of-the-box functionaliteit:** Standaard Functionaliteit die beschikbaar is zonder aanpassingen of configuraties.
    - **Score van 0:** functionaliteit grotendeels afhankelijk van eigen implementatie.
    - **Score van 1:** enige aanpassingen nodig.
    - **Score van 2:** weinig tot geen aanpassingen nodig.

Tabel 3

Overzicht frameworks voor documentherkenning

Framework	costs per 200.000	out-of-the-box experience	Integration
Azure Form Recognizer	$ 2.000,00	2	0
GPT-3	$ 290,00	2	0
Document AI	$ 13.000,00	2	1
AWS Textract	$ 13.300,00	0	2
Existing solution	$ 298,50	-	-

Noot. De kosten voor GPT-3 zijn afhankelijk van de hoeveelheid tekst in het document (zie bijlage E).

5.3.3.1. Amazon Textraxt.

Amazon Textraxt zou goed integreren met de bestaande Moneybird systemen omdat Moneybird op dit moment veel gebruikmaakt van AWS-diensten. Echter ondersteunt de Textraxt nog geen Nederlandse facturen en bonnetjes (Amazon, n.d. -a). Om toch goede resultaten te behalen is veel extra configuratie nodig omdat de data geëxtraheerd moeten worden met reguliere expressies (Amazon, n.d. -b). Verder is de oplossing vrij duur met een prijs van $ 13.300 euro voor het verwerken van 200.000 documenten (Amazon, n.d. -c).

5.3.3.2. Google Document AI.

In tegenstelling tot AWS is Google's Document AI-algoritme specifiek getraind op Nederlandse facturen en bonnetjes (Google, n.d.-a). waardoor er minder configuratie nodig is. Hoewel binnen Moneybird al Google-diensten worden gebruikt, is de inzet ervan niet zo uitgebreid als dat van AWS. Bij gebruik van gespecialiseerde modellen bedragen de kosten voor het verwerken van 200.000 facturen 20.000 dollar. Bij gebruik van het generieke model bedragen de kosten $ 13.000 dollar (Google, n.d.-c).

5.3.3.3. Microsoft Azure Form Recognizer.

Net als Google Document AI ondersteunt Microsoft Azure Form Recognizer Nederlandse facturen en bonnetjes zonder dat er extra configuratie nodig is (Microsoft, 2023. -b). Een nadeel is dat Microsoft Azure producten nog niet gebruikt worden binnen de infrastructuur van Moneybird, waardoor er binnen het bedrijf geen ondersteuning en expertise is voor dit platform. Daarentegen is het de meest kosteneffectieve oplossing, aangezien het verwerken van 200.000 documenten $ 2.000 dollar kost (Microsoft, 2023. -b).

5.3.3.4. Generative Pre-trained Transformer 3 (GPT-3).

GPT-3 (Generative Pre-trained Transformer 3), bekend van chat-GPT, is toegevoegd als experimentele optie. GPT-3 wordt op dit moment gebruikt als Slack-chatbot binnen de organisatie maar verder niet op een zakelijke manier toegepast. Uniek aan GPT-3 in vergelijking met de andere frameworks is dat het een *generative AI* is, wat betekent dat het niet specifiek ontwikkeld is voor de gegeven use-case. Het kostenmodel van GPT-3 is anders dan bij de andere frameworks.

De kosten bedragen $0.002 per 1.000 tokens (Microsoft, n.d. -a) en zijn daardoor afhankelijk van de lengte van zowel de invoer- als uitvoer tekst van het model. Ervan uitgaande dat de gemiddelde factuur drie factuurregels bevat, dan zouden de kosten voor het verwerken van 200.000 documenten neerkomen op $ 290 dollar (zie bijlage E).

5.3.4. Conclusie

Uit de vergelijking blijkt dat Microsoft Azure Form Recognizer de beste prijs-kwaliteitverhouding biedt voor de use-case. Het ondersteunt zowel Nederlandse facturen als bonnetjes en behoort daarnaast tot de goedkopere opties. Om een beter inzicht te krijgen in de daadwerkelijke prestaties van het framework, is een benchmark uitgevoerd om de nauwkeurigheid ervan te meten. De resultaten van deze benchmark zijn beschreven in hoofdstuk 7.5.

5.4. Hoe kan een fine-tuned machine learning model gebruikt worden voor documentherkenning?

5.4.1. Inleiding

De bestaande oplossing voor documentherkenning maakt gebruik van Optical Character Recognition (OCR) voor het uitlezen van de tekst. OCR heeft echter enkele beperkingen. Het gebruik van OCR vraagt veel rekenkracht van de computer en de opmaak van het oorspronkelijke document gaat verloren.

Om dit probleem aan te pakken, is een nieuwe methode ontwikkeld genaamd Donut (Document Understanding Transformer). In tegenstelling tot OCR gebruikt Donut een pseudo-OCR-techniek om invoer rechtstreeks om te zetten in het gewenste formaat. Donut heeft bewezen uitstekende resultaten te behalen in nauwkeurigheid en snelheid, en overtreft daarmee de prestaties van bestaande machine learning oplossingen (Kim et al., 2021).

Figuur 6

Benchmark accuraatheid machine learning frameworks (Kim et al., 2021, p. 10)

			CORD [45]			Ticket [12]			Business Card			Receipt		
	OCR	#Params	Time (s)	F1	Acc.	Time (s)	F1	Acc.	Time (s)	F1	Acc.	Time (s)	F1	Acc.
BERT* [22]	✓	$86^{\dagger}_{M} + \alpha^{\ddagger}$	1.6	73.0	65.5	1.7	74.3	82.4	1.5	40.8	72.1	2.5	70.3	54.1
BROS [18]	✓	$86^{\dagger}_{M} + \alpha^{\ddagger}$	1.7	74.7	70.0									
LayoutLM [65]	✓	$89^{\dagger}_{M} + \alpha^{\ddagger}$	1.7	78.4	81.3									
LayoutLMv2* [64,66]	✓	$179^{\dagger}_{M} + \alpha^{\ddagger}$	1.7	78.9	82.4	1.8	87.2	90.1	1.6	52.2	83.0	2.6	72.9	78.0
Donut		$143^{\dagger}_{M}$	**1.2**	**84.1**	**90.9**	**0.6**	**94.1**	**98.7**	**1.4**	**57.8**	**84.4**	**1.9**	**78.6**	**88.6**
SPADE* [25]	✓	$93^{\dagger}_{M} + \alpha^{\ddagger}$	4.0	74.0	75.8	4.5	14.9	29.4	4.3	32.3	51.3	7.3	64.1	53.2
WYVERN* [21]	✓	$106^{\dagger}_{M} + \alpha^{\ddagger}$	1.2	43.3	46.9	1.5	41.8	54.8	1.7	29.9	51.5	3.4	71.5	82.9

5.4.2. Dataset

Moneybird B.V. heeft een dataset van 834 facturen beschikbaar gesteld voor het trainen van het Donut-model. Om dit model te kunnen trainen, is het noodzakelijk om voor elk document een JSON-bestand te maken waarin de inhoud van het document wordt beschreven. Dit JSON-bestand wordt ook wel de "ground truth" genoemd.

Een deel van de dataset is geautomatiseerd verwerkt met behulp van Azure Form Recognizer. Voor iedere factuur wordt door middel van de Form Recognizer API de *ground truth* gegenereerd.
Een andere dataset bestaat uit 100 facturen die handmatig zijn verwerkt. Hoewel deze dataset kleiner is dan de volledige dataset, is de kans groter dat de inhoud van de *ground truth* correct is, omdat ze handmatig zijn samengesteld en gecontroleerd.

5.4.3. Voorbereiding dataset

Donut past een three-way data split toe, waarbij de dataset in drie delen wordt opgesplitst. Deze aanpak is gericht op het minimaliseren van *overfitting* en het verminderen van *bias* in de testresultaten, om zo de algehele nauwkeurigheid van het model te verbeteren (Joseph, 2022).

Er bestaat geen overeenstemming over wat de optimale verhouding is voor het splitsen van datasets, aangezien er geen duidelijke richtlijnen zijn vastgesteld (Joseph, 2022). De verhouding van 10% voor testgegevens, 10% voor validatiegegevens en 80% voor trainingsgegevens, zoals gebruikt in het onderzoek van Kim et al. (2021), zal worden gebruikt in dit onderzoek.

5.4.4. Beperkingen in taal

Hoewel Donut een meertalig model is, is het niet specifiek getraind op de Nederlandse taal. Tijdens steekproeven bleek het model Nederlandse teksten met Engelse invloeden te tonen. Hieronder staat een voorbeeld van een dergelijke output:

```
{'description': 'Verzend- en Ainstralekosten'}
```

Om de meertaligheid van Donut uit te breiden, heeft Donut een generator uitgebracht onder de naam SynthDoG. Als input verwacht de generator een database dump van Wikipedia in de betreffende taal, geconverteerd naar .TXT-formaat. SynthDoG genereert vervolgens afbeeldingen van verschillende kwaliteiten met de tekst van Wikipedia. Door deze output te gebruiken, kan Donut worden getraind om de Nederlandse taal te leren en zijn meertaligheid verder uit te breiden (cloveai, n.d.; Kim et al., 2021).

5.4.5. Resultaten

Er zijn verschillende datasets gebruikt om de werking van Donut te testen. Een van deze datasets was een pre-built model getraind op 1280 Engelstalige bonnetjes. Daarnaast is een dataset met 1.000 Engelstalige bonnetjes en bijbehorende metadata gebruikt die afkomstig is uit een openbare bron (zie bijlage F). Ten slotte zijn 834 facturen van Moneybird B.V. gebruikt om het model te testen.

Donut heeft een scriptje dat een inschatting kan maken over de nauwkeurigheid van het getrainde model. Dit is gebruikt voor het vaststellen van de nauwkeurigheid van de modellen.

Tabel 4

Modellen en hun accuraatheid

Dataset	Content	Size	Accuracy score
Naver-clova-ix/donut-base	Engelstalige bonnetjes. Pre-trained model.	1280	91.05%
IDCAR-2019-SCROIE (zie bijlage F)	Engelstalige bonnetjes	1000	94.04%
Model getraind op Moneybird B.V. facturen (zie bijlage G)	Nederlandstalige facturen. Data geautomatiseerd verwerkt.	834	34.04%
Model getraind op Moneybird B.V. facturen (zie bijlage G)	Nederlandstalige facturen. Data handmatig verwerkt.	100	12,00%

5.4.5.1. Opmerking bij het resultaat.

De dataset IDCAR-2019-SCROIE is getraind op het herkennen van slechts een select aantal kenmerken, met daarin geen nesting. Het pre-trained model en het model getraind op facturen van Moneybird B.V. zijn getraind om meer kenmerken met nesting.

Als de test- en validatie data fouten bevatten, kan dit invloed hebben op de schatting van de nauwkeurigheid van het model. Het is daarom niet gegarandeerd dat de resultaten volledig correct zijn (Bresseler, 2021).

5.4.6. Conclusie

De resultaten van de Naver-clova-ix/donut-base en IDCAR-2019-SCROIE datasets zijn veel hoger dan de dataset bestaande uit facturen van Moneybird B.V. Het is moeilijk vast te stellen waarom de nauwkeurigheid zo laag is, aangezien dit kan afhangen van veel factoren. De kans is groot dat het te maken heeft met de kleine omvang van de dataset en het beperkte aantal variaties daarin. Een grotere dataset geeft immers vaak betere resultaten (Google Developers, n.d.).

Verder is Donut tegen het einde van 2022 uitgebracht en daarom is er momenteel weinig documentatie beschikbaar. Het kan daarom zijn dat er fouten zijn gemaakt bij het inrichten van het model. Echter, zijn de veelbelovende resultaten een indicatie dat Donut in de toekomst verder zal groeien en dat er meer documentatie beschikbaar zal komen die een verbeterde implementatie ondersteunen.

5.5. Welk algoritme heeft de hoogste nauwkeurigheid bij het herkennen van specifieke kenmerken op facturen en bonnetjes?

5.5.1. Inleiding

Om een indicatie te krijgen van de nauwkeurigheid van de frameworks, is een benchmark uitgevoerd. In totaal zijn 50 bonnetjes en 130 facturen gebruikt. De informatie van deze documenten is handmatig overgebracht naar een JSON-datastructuur, ook bekend als de *ground truth*.

Deze JSON-bestanden zijn gebruikt om de output van het algoritme te vergelijken met de verwachte waarde van het document. De benchmark is uitgevoerd om te meten hoe nauwkeurig de frameworks informatie op de documenten kunnen herkennen.

5.5.2. Benchmark meetwaarden

De benchmark is uitgevoerd op de document onderdelen die momenteel kunnen worden ingevoerd voor het inboeken van facturen en bonnetjes binnen Moneybird. De nauwkeurigheid wordt weergegeven als een percentage, dat het gemiddelde is van de nauwkeurigheid van alle gemeten elementen op het document.

Voor financiële gegevens is het essentieel dat deze geen fouten bevatten. Daarom is voor dergelijke gegevens een nauwkeurigheidspercentage van 100% vereist. Dit betekent dat de *Character Error Rate* (CER), die het foutenpercentage meet, gelijk moet zijn aan 0%. Indien de CER hoger is dan 0%, wordt de nauwkeurigheid als 0% beschouwd. Tabel 5 geeft een voorbeeld:

Tabel 5
Voorbeeld van meetwaarde die wordt gebruikt om de nauwkeurigheid van financiële gegevens te benchmarken

Ground truth	Voorspelde waarde	CER-percentage	Nauwkeurigheidspercentage
Hallo	Hall0	20%	0%

Hallo	Hallo	0%	100%

In de eerste onderzoeksvraag is vastgesteld dat de omschrijving van een product of dienst beknopt mag zijn, zolang het maar aangeeft welke producten of diensten worden geleverd. Het is daarom niet nodig dat de beschrijving exact overeenkomt met de *ground truth*. Om te bepalen in hoeverre de voorspelde waarde overeenkomt met de verwachte waarde is het Levenshtein-distance-algoritme gebruikt.

Tabel 6

Document onderdelen en hun meetwaarde

Documentonderdeel	Meetwaarde
Factuurnummer	0% (bevat fouten) – 100% (Geen fouten)
Factuurdatum	0% (bevat fouten) – 100% (Geen fouten)
Totale prijs	0% (bevat fouten) – 100% (Geen fouten)
Factuurregel/Stukprijs	0% (bevat fouten) – 100% (Geen fouten)
Factuurregel/Belastingpercentage	0% (bevat fouten) – 100% (Geen fouten)
Factuurregel/Aantal	0% (bevat fouten) – 100% (Geen fouten)
Factuurregel/Omschrijving	Levenshtein-distance percentage

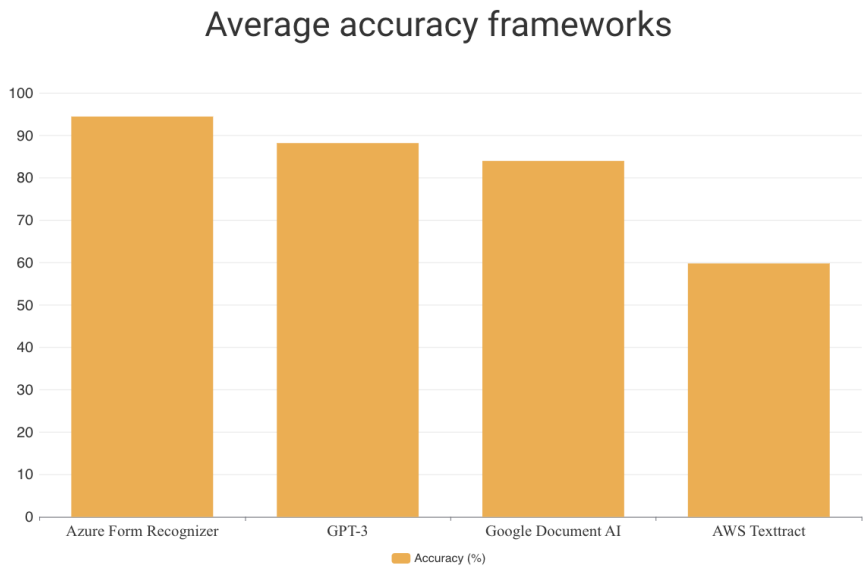
5.5.3. Resultaten

De benchmark (Tabel 7, volgende pagina) toont dat Azure Form Recognizer de hoogste scores behaalt. Zowel in het algemeen als op alle individuele velden, met uitzondering van 'tax rate', waar GPT-3 iets beter presteert. De gemiddelde score van Azure Form Recognizer is het hoogst, gevolgd door GPT-3 met een gemiddelde score van 86,78%. Google Document AI presteert goed op individuele velden, zoals invoice number, date en total amount, maar scoort lager op de factuurregels.

AWS Textract presteert het minst goed van alle frameworks, met lage scores op alle onderdelen vergeleken met andere frameworks. Een uitzondering is de nauwkeurigheid in het bepalen van de stukprijs (unit price), waar het beter presteert dan Google Document AI.

Figuur 7

Frameworks en hun accuraatheid



Tabel 7

Frameworks en hun accuraatheid per document onderdeel

Service	Invoice number	Date	Total price	Unit price	Tax rate	Quantity	Average
Azure							94.21%
	98.51%	98.88%	97.75%	89.18%	98.97%	81.96%	
Document AI							86.65%
	96.69%	98.88%	97.75%	57.29%	97,91%	71.35%	
GPT-3							92.89%
	90.92%	95.51%	94.38%	89.94%	99.44%	87.15%	
Textract							55.28%
	62.45%	69.66%	92.13%	67.43%	29.14%	10.86%	

Noot. Het onderdeel 'description' is weggelaten omdat het een vertekend beeld van de gemiddelde score geeft

5.5.4. Conclusie

Op basis van de uitgevoerde benchmark kan geconcludeerd worden dat Azure Form Recognizer als de beste keuze naar voren komt. Het levert gemiddeld gezien de meest nauwkeurige resultaten bij het herkennen van velden op de documenten in de testset. Bovendien is het gemiddeld genomen een kosteneffectieve oplossing in vergelijking met de andere frameworks (Tabel 3, pagina 21).

6. Ontwerp

6.1. Inleiding

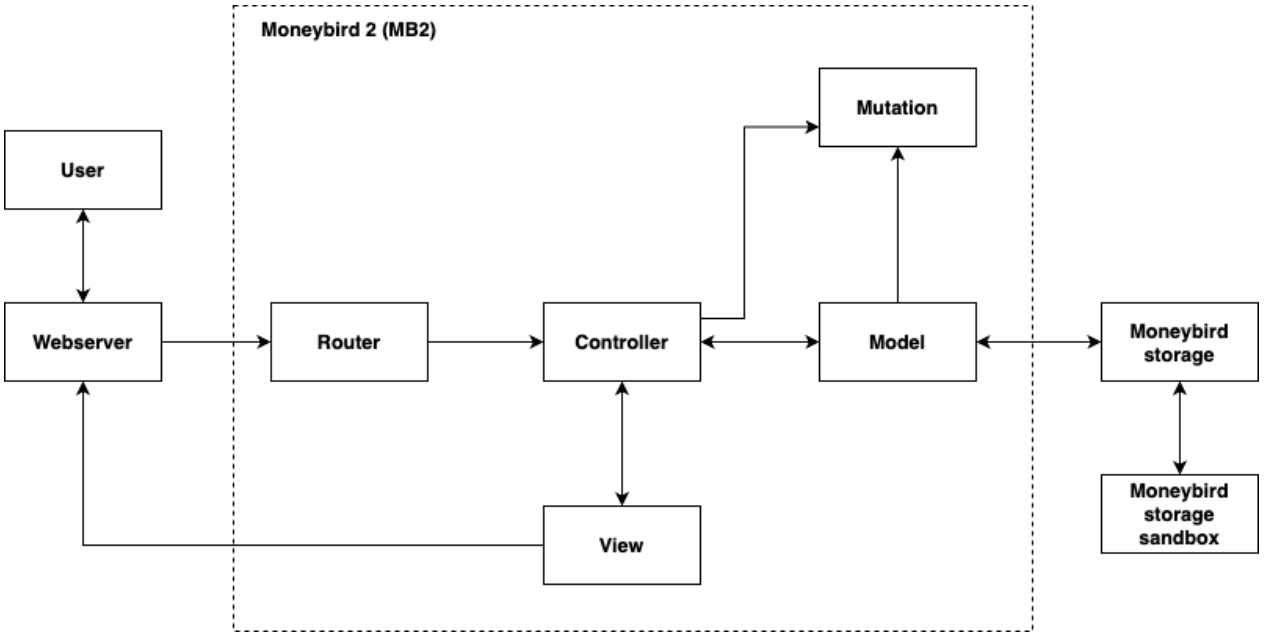
Op basis van het gedane onderzoek is besloten om de Microsoft Azure Form Recognizer API te gebruiken. In dit hoofdstuk wordt beschreven welke programmeertaal en frameworks zullen worden gebruikt, hoe het systeem integreert met het bestaande systeem en wordt er een overzicht van de architectuur gegeven.

6.2. Overzicht Moneybird 2 (MB2) structuur

Het Moneybird-platform (mb2) is ontwikkeld in het Ruby on Rails-framework, dat de Model-View-Controller (MVC) architectuur volgt. In het MVC-patroon wordt de complexiteit van de applicatie opgedeeld in drie delen, namelijk het model, de view en de controller. Dit helpt om de complexiteit te beheren en de verantwoordelijkheden van de applicatie duidelijk te verdelen over deze componenten (Singh, 2020).

Figuur 8

Overzicht van de Moneybird structuur



In de bovenstaande afbeelding wordt de architectuur van Moneybird weergegeven. Elk vierkant vertegenwoordigt een specifiek component van het systeem, met elk zijn eigen taak.

Wanneer een gebruiker de applicatie bezoekt, is het de taak van de router om ervoor te zorgen dat de juiste methode in de bijbehorende **controller** wordt aangeroepen. De router fungeert als een laag tussen de gebruikersinterface en de backend van de applicatie.

In Ruby on Rails wordt server-side rendering toegepast, waarbij de **view**-component verantwoordelijk is voor het weergeven van de gegevens afkomstig van de controller.

Mutations worden gebruikt om de business logica uit te voeren. Ze veranderen de staat van de applicatie en zorgen ervoor dat de logica wordt verdeeld over verschillende bestanden, waardoor het gemakkelijker te testen en te hergebruiken is.

Het **model** definieert een entiteit en de bijbehorende attributen. Het omvat niet alleen de gegevensstructuur, maar ook de specifieke business logica en validaties.Voor de documentanalyse wordt gebruikgemaakt van het attachment model en ook het document model.

Moneybird storage is een systeem dat een interface biedt voor het opslaan, beheren en ophalen van gegevens in AWS S3, een cloudopslagdienst. Ook biedt het zogenoemde 'transformer stappen'. Dergelijke stappen voeren een specifieke bewerking uit op een afbeelding in een gegeven volgorde. Net als Moneybird 2 maakt het Moneybird Storage systeem ook gebruik van het Ruby on Rails framework.

Vanwege veiligheidsredenen is het niet toegestaan om libraries, zoals Magick, te gebruiken binnen de Moneybird storage omgeving. Dergelijke bewerkingen moeten plaatsvinden binnen de **Moneybird storage sandbox**, een volledig afgeschermde omgeving.

6.3. Voortbouwen op het bestaande systeem

De bijgewerkte bonnetjes- en factuurherkenning zal uiteindelijk de oude implementatie vervangen. Een deel van de oude implementatie zal nog nodig blijven voor andere doeleinden. De oude implementatie wordt bijvoorbeeld ook gebruikt om te controleren of een IBAN-nummer correct is en voor autocomplete functionaliteit. De functionaliteiten en de bijbehorende tests moeten dus naast het nieuwe systeem blijven werken maar zullen geen gebruik van elkaar maken.

6.3.1. Béta functie

Om de implementatie gecontroleerd te kunnen schalen naar meerdere gebruikers, is besloten om de nieuwe functie als bèta-functionaliteit uit te rollen. Dit betekent dat de vernieuwde documentherkenning naast de bestaande documentherkenning wordt geïmplementeerd. Gebruikers kunnen eenvoudig tussen de implementaties schakelen via hun instellingen.

6.4. Classificatie

Azure Form Recognizer maakt gebruik van twee verschillende modellen: één voor facturen en één voor bonnetjes. Bij het gebruik van de service moet op voorhand het documenttype bekend zijn. Daarom is het belangrijk dat op voorhand bekend is welk type document herkend moet worden.

Het huidige algoritme probeert het documenttype te achterhalen door middel van patroonherkenning, waarbij het vertrouwt op specifieke patronen in de tekst.

In tegenstelling tot op reguliere expressies gebaseerde oplossingen, zijn machine learning

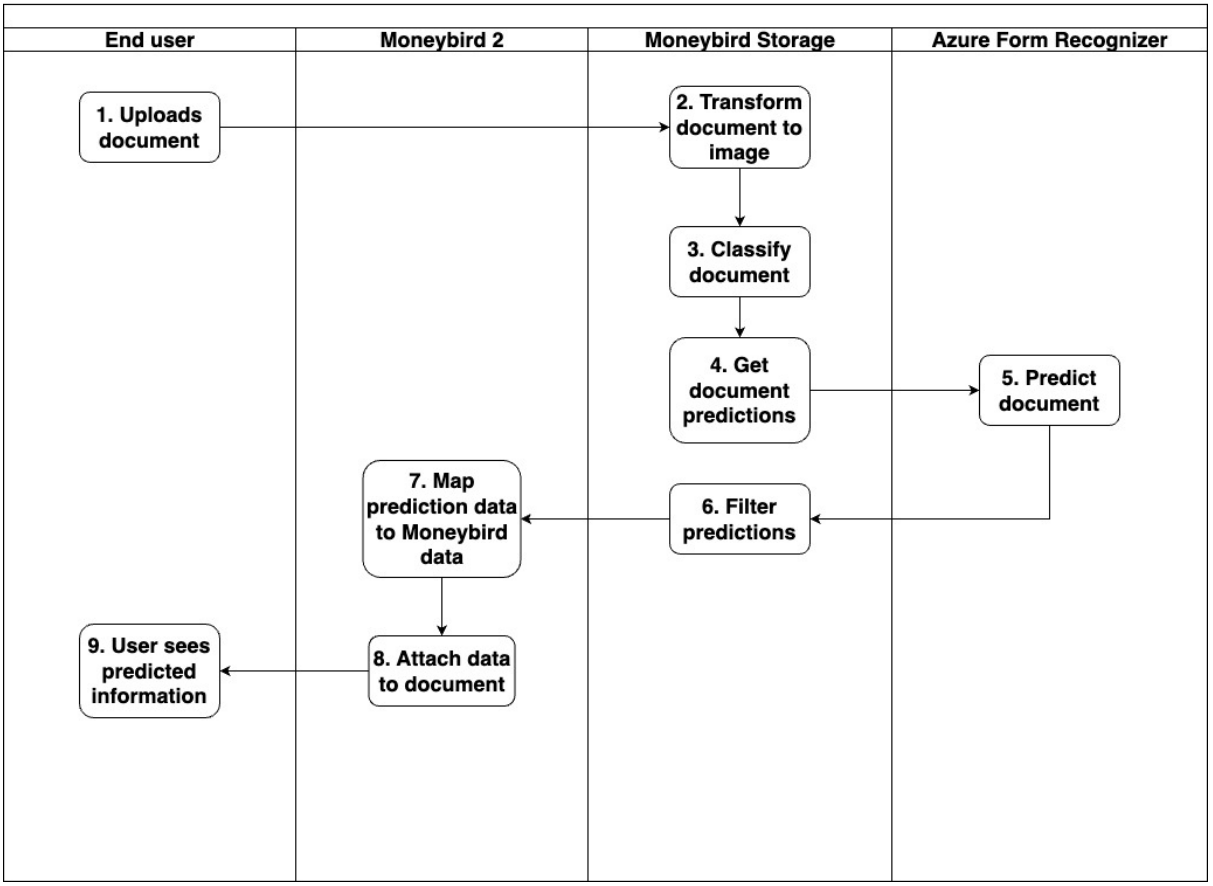
oplossingen niet afhankelijk van specifieke tekstpatronen in een document. Ook hebben machine learning modellen in het verleden uitstekende resultaten behaald bij het classificeren van afbeeldingen (Tensorflow, n.d.). Om deze reden is ervoor gekozen om in de toekomstige implementatie gebruik te gaan maken van een machine learning oplossing.

6.5. Flow

Om een overzicht te geven van de beoogde implementatie, is er een flow-diagram opgesteld. Het proces start wanneer een gebruiker een inkomend document uploadt en eindigt wanneer de voorspellingen van het document worden weergegeven.

Figuur 9

Application flow



Het proces begint wanneer de gebruiker een inkomend document uploadt in Moneybird 2 (Figuur 9, Stap 1). Vervolgens wordt het document omgezet naar een afbeelding, zodat het gemakkelijk bewerkbaar is in de volgende stappen. (stap 2) Daarna wordt het documenttype bepaald (stap 3), wat kan variëren tussen een factuur (invoice), bonnetje (receipt) of een ander type document (other).

In de volgende stappen (stap 4 en 5) worden de voorspellingen voor het document verkregen via de Form Recognizer API. Het specifieke type document (bonnetjes of facturen) bepaalt welke API-URL wordt gebruikt.

Aangezien het bonnetjes- en facturenmodel een verschillende structuur hebben, is het nodig om dit verschil te overbruggen. Om dit te bereiken, zal de data worden gefilterd om een uniform formaat te creëren (stap 6).

Daarna wordt de data teruggestuurd naar het Moneybird 2-systeem en daar opgeslagen. De opgeslagen data is nog ongestructureerd en wordt vervolgens gemapt naar de juiste velden. Extra logica wordt toegepast om ervoor te zorgen dat de data het juiste formaat heeft (stap 7). Als laatste stap wordt de data toegewezen aan het document model, waarna de voorspellingen aan de gebruiker worden weergegeven (stap 8 en 9). De data wordt ook direct opgeslagen in de database, zodat de voorspellingen beschikbaar blijven.

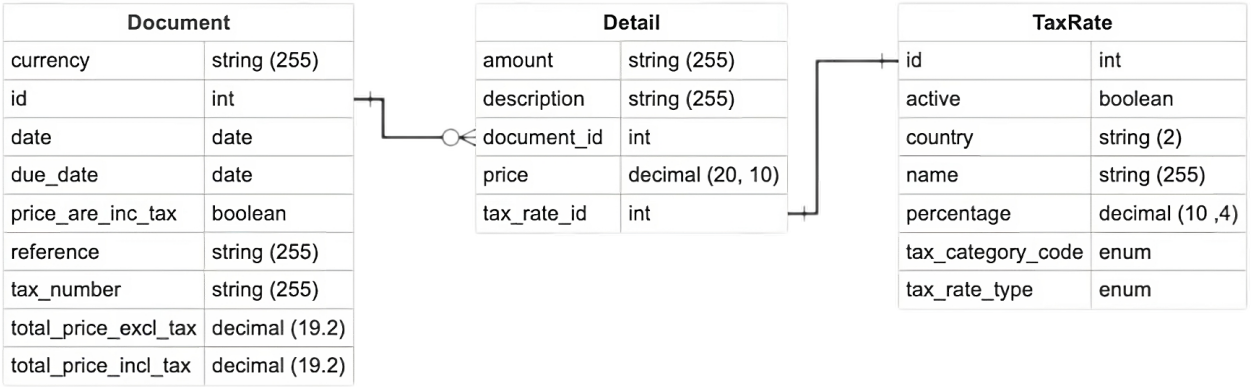
6.6. Opslag

6.6.1. Database

De voorspellingen worden opgeslagen in de database. Hiervoor zijn geen aanpassingen aan het database ontwerp nodig.

Figuur 10

Database diagram implementatie



De "**Document**" tabel geeft de globale informatie over het document. Het vernieuwde algoritme kan alle kolommen van de tabel voorspellen. Het oude algoritme ondersteunt alleen de 'total price', de 'date'' en de 'reference' kolommen.

De tabel "Document" heeft een one-to-many relatie met de tabel "**Detail**", waarin de individuele regels of "document lines" van het document worden opgeslagen. In tegenstelling tot het oude algoritme, heeft het nieuwe algoritme de mogelijkheid om voorspellingen te doen voor de documentregels.

De "Detail" tabel en de "**TaxRate**" tabel hebben een one-to-one relatie. Deze relatie bepaald welke belastingtarieven van toepassing zijn op de documentregel. Deze functionaliteit wordt uitsluitend ondersteund door het nieuwe algoritme.

6.7. Foutafhandeling

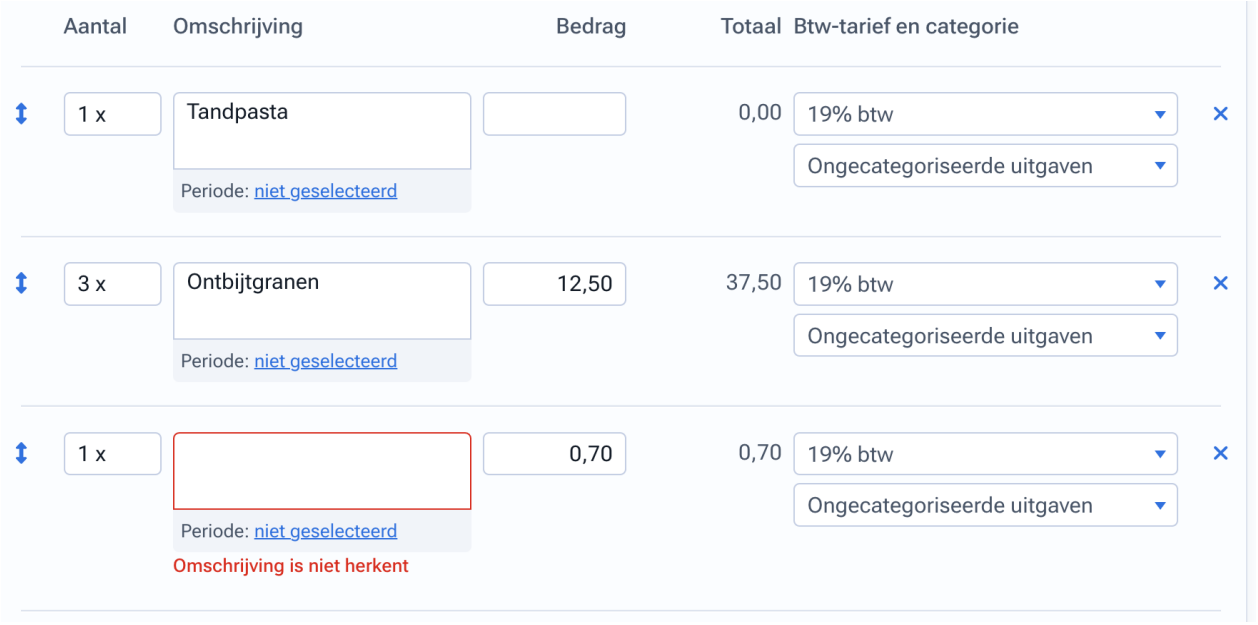
Er is voor gekozen om het tonen van de fouten aan de gebruiker op een minimalistische manier aan te pakken.

Wanneer er zich een fout voordoet aan de kant van Azure, worden er geen voorspellingen weergegeven. De gebruiker kan echter nog steeds doorgaan met zijn werk zonder dat het proces wordt geblokkeerd. Het enige effect dat de gebruiker zal ondervinden, is dat er tijdelijk meer handmatig werk nodig is, aangezien er tijdelijk geen suggesties beschikbaar zijn.
Als er een fout optreedt tijdens het voorspellen van het documenttype of als het documenttype als 'overig' wordt geclassificeerd, wordt altijd het factuurmodel gebruikt.

Als een veld met onvoldoende zekerheid (minder dan 70%) wordt herkend, is ervoor gekozen om het betreffende veld leeg te laten en een kleine waarschuwing onder het veld weer te geven. Het is aan de gebruiker om dit veld handmatig in te vullen.

Figuur 11

Waarschuwing niet of met onvoldoende zekerheid herkent veld



7. Realisatie

7.1. Inleiding

Dit hoofdstuk geeft een overzicht van de technische beslissingen die zijn genomen tijdens de realisatiefase. De functionaliteiten zijn geïntegreerd in de bestaande systemen van Moneybird die zijn ontwikkeld met behulp van het Ruby on Rails framework. Verder is het getest volgens de standaarden bij Moneybird.

7.2. Risico's

De Microsoft Azure Form Recognizer API is een extern component. Om die reden is het belangrijk om te onderzoeken waar de technische en organisatorische risico's liggen. Om deze risico's in kaart te brengen is een risicoanalyse (zie bijlage J) opgesteld.

Een belangrijke bevinding is dat er standaard een limiet van 15 verzoeken per seconde geldt voor Form Recognizer. Dit risico kan voorkomen worden door *scaling* in te schakelen (Microsoft, 2023.-b).

Form Recognizer is opgezet in de regio West-Europa. Hierdoor voldoet het aan de ISO 27001-standaarden (Diepenmaat, 2021) die bij Moneybird toegepast worden. Meer informatie over potentiële risico's, gevolgen en oplossingen zijn te vinden in bijlage J.

7.3. Document classificatie

Om het documenttype vast te stellen is gekozen om gebruik te maken van een Tensorflow classificatie model (Figuur 9, p. 30, Stap 3). Dit model is getraind op een combinatie van bonnetjes en facturen afkomstig van Moneybird, evenals twee openbare datasets (Clovaai, 2019; Mohammad, 2020).

Tabel 8

Classificatie klassen en hun accuraatheid (zie bijlage I)

Class	Size	Accuracy score
Invoice	403	93%
Receipt	1370	100%
Other	647	98%

Noot: de class 'other' is getraind op willekeurige documenten zoals willekeurige afbeeldingen, powerpoint slides en posters.

7.3.1. Training

Het machine learning-model is getraind met Teachable Machine, een project dat ontwikkeld is door Google. Deze keuze is gemaakt vanwege de uitstekende resultaten die het biedt en de

mogelijkheid om modellen te trainen via de webbrowser, zonder dat er veel extra configuratie vereist is (Hyunja, 2020).

De dataset is eerst gefilterd om machine learning-bias te voorkomen. Hierbij zijn dubbele afbeeldingen en afbeeldingen die sterk op elkaar lijken verwijderd. Daarna is de dataset opgedeeld in drie verschillende categorieën, namelijk facturen, bonnetjes en overige afbeeldingen.

7.3.2. Implementatie

Het Tensorflow-model is omgezet naar een ONNX-model, waardoor het mogelijk is om het model in Ruby te gebruiken. Het aanroepen van het model gebeurt met behulp van de ONNX_runtime Gem voor Ruby.

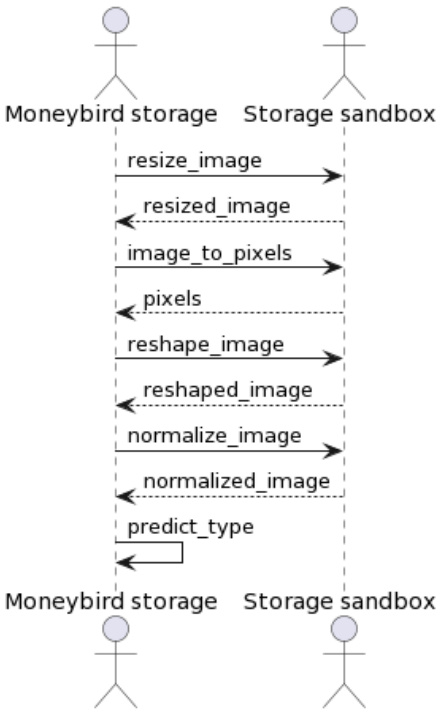
Het onderstaande commando is gebruikt om het oorspronkelijke Tensorflow om te zetten naar een ONNX-model:

```
python -m tf2onnx.convert --opset 10 \
  --saved-model tensorflow_model/saved_model \
  --output model.onnx
```

Voordat het model kan worden aangeroepen, moet de afbeelding eerst worden omgezet in pixels en vervolgens worden genormaliseerd naar een specifiek formaat. De stappen die daarvoor worden gevolgd zijn toegelicht in figuur 12.

Figuur 12

Sequence diagram bewerken afbeelding voor classificatie



Na het bewerken van de afbeelding wordt het model aangeroepen. Het model geeft de voorspelling terug als een integer waarde. Deze waarde correspondeert met de positie in de array die overeenkomt met de bijbehorende classificatie. De onderstaande code illustreert de manier waarop de voorspelling van het model wordt verkregen:

```
prediction = model.predict({ 'sequential_1_input' => prepared_pixels })
labels = %w[invoice receipt other]
predicted_probabilities = prediction['sequential_3'] // 0, 1, 2
label  = labels[predicted_probabilities]
```

7.4. Voorspelling (Form Recognizer)

Zoals in hoofdstuk 6.5 (p. 30, Stap 4 en 5) genoemd, wordt de Form Recognizer API gebruikt om voorspellingen te maken op basis van een document. Het voorspellingsproces bestaat uit drie stappen: het initiëren van de voorspellingsoperatie, het wachten tot de operatie is voltooid en het verkrijgen van de voorspelde resultaten.

Allereerst wordt een POST-request naar de API gestuurd, waarbij de body van het verzoek een base64-string van het document bevat. Als reactie op dit verzoek stuurt de Form Recognizer API een "operation-URL" terug in de headers van het verzoek. De onderstaande code toont hoe de operation-url wordt verkregen:

```
payload = { 'base64Source' => img_base64 }.to_json
Excon.post(url, headers: headers, body: payload).headers['Operation-Location']
```

Om de voorspelling op te halen, wordt de zojuist verkregen operation-URL gebruikt in een GET-request. Als antwoord hierop wordt de status van de voorspellingsoperatie teruggestuurd.

Als de status "succeeded" is, betekent dit dat de voorspelling gereed is en wordt de voorspelling geretourneerd. Als de status "running" of "notStarted" is wordt na 1 seconde nogmaals geprobeerd om de voorspelling op te halen. Deze keuze om 1 seconde te wachten is gebaseerd op de best practices van Microsoft (Microsoft, 2023. -b).

Om te voorkomen dat dit proces resulteert in een oneindige lus, wordt het maximaal 20 keer herhaald. Als het maximaal aantal pogingen wordt overschreden, wordt het proces afgebroken. In dat geval zal de gebruiker geen voorspelling te zien krijgen, maar kan wel verder werken.

De voorspelling kan niet direct worden gebruikt, omdat het formaat van de facturen en bonnetjesmodellen van elkaar verschillen. In het volgende hoofdstuk wordt toegelicht hoe de data wordt getransformeerd naar een uniform formaat.

7.5. Data transformer

Om de verschillen tussen het bonnetjesmodel en het facturenmodel te overbruggen, is er een 'data transformer' ontwikkeld (Figuur 9, p. 30, Stap 6). Deze transformer zorgt ervoor dat de gegevens altijd in een uniform formaat worden teruggegeven.

Om ervoor te zorgen dat deze transformer in de toekomst makkelijk bij te werken is, is het principe van DRY (Don't Repeat Yourself) toegepast om dubbele code te vermijden.

In de onderstaande code wordt de methode "***parse_field***" hergebruikt om specifieke waarden uit de output van Form Recognizer te halen. Om een aanpassing te maken, moeten alleen het doelveld en het pad naar de betreffende waarde worden aangepast.

```
    {
       company_name: parse_field("MerchantName", %w[valueString], document),
       date: parse_field("TransactionDate", %w[valueDate], document),
       address: parse_field("MerchantAddress", %w[valueAddress], document),
       sub_total: parse_field("SubTotal", %w[valueCurrency amount], document),
[..]
}
```

De onderstaande code bevat de methode "parse_field". Deze methode wordt gebruikt om de waarde ("value") en de zekerheid ("confidence") op te halen voor elk veld dat wordt meegegeven.

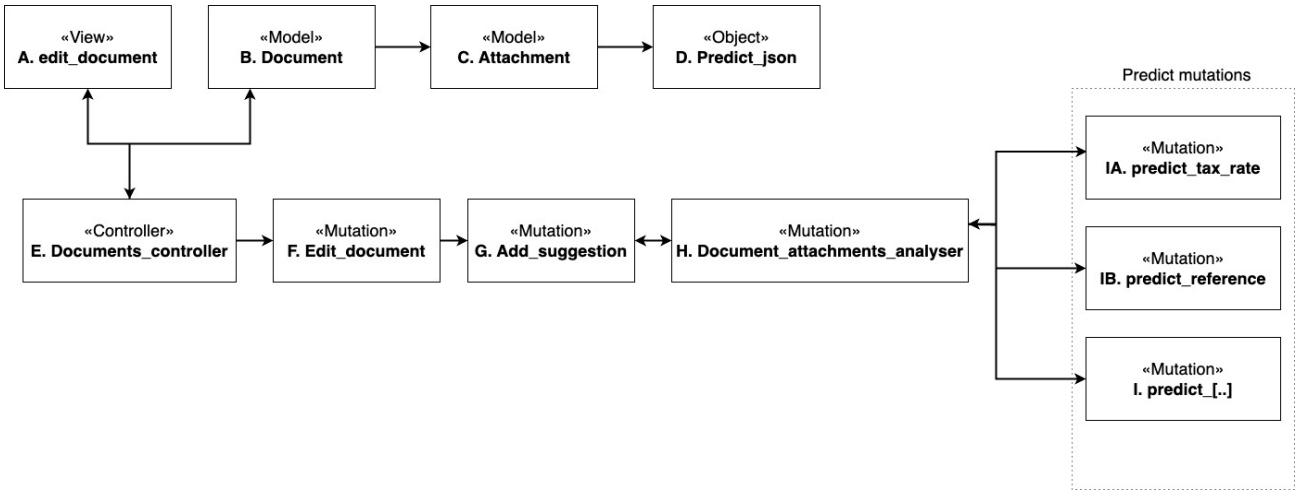
```
   def parse_field(external_key, target_field_path, document)
      return nil if document[external_key].nil?

      value = document.dig(external_key, *target_field_path)

      {
        value: value,
        confidence: document[external_key]["confidence"]
      }
    end
```

7.6. Mappen en tonen van de data in MB2

Voordat de individuele gegevens van Form Recognizer getoond worden, moet de data eerst om worden gezet in een voorspelling per veld. Deze stappen worden uitgevoerd in het Moneybird 2 (MB2) systeem (Figuur 9, p. 30, Stap 7 en 8).

Figuur 13

Overzicht componenten Moneybird 2 - document analyse



Het bovenstaande diagram toont de componenten die gebruikt zijn voor de implementatie en hun onderlinge samenhang. Elk vierkant vertegenwoordigt een component in het systeem, samen met het bijbehorende type.

De '**documents_controller**' (Figuur 13, E) is verantwoordelijk voor het aanleveren van de juiste data voor de '**edit_document**' view (Figuur 13, A). Deze controller maakt gebruik van het '**document'-model** (Figuur 13, B), dat op zijn beurt het **'attachment'**-model (Figuur 13, C) gebruikt.

Het 'attachment'-model zorgt voor toegang tot de voorspellingen. De voorspellingen zijn opgeslagen in een object genaamd '**predict_json**' (Figuur 13, D), Dit is eigenlijk een uitgebreid JSON-object met hulpfuncties.

7.6.1. Data mappen met predict mutaties

Er is besloten om voor elke voorspelling een aparte mutatie te maken (Figuur 13, I). Deze mutaties zijn gebruikt om de output van Form Recognizer om te zetten naar een individuele voorspelling per veld.

In de mutaties wordt rekening gehouden met het ontbreken van een herkend veld door meerdere alternatieve benaderingen te gebruiken, waar mogelijk.

Neem bijvoorbeeld het voorspellen van het documentkenmerk (Figuur 13, IB). In eerste instantie wordt geprobeerd om het veld 'reference' op te halen. Als dit veld niet beschikbaar

is, wordt het 'purchase order' (PO-nummer) veld gebruikt als alternatief. Uiteindelijk hebben beide velden betrekking op een identificatienummer en voldoen dus aan de use-case.

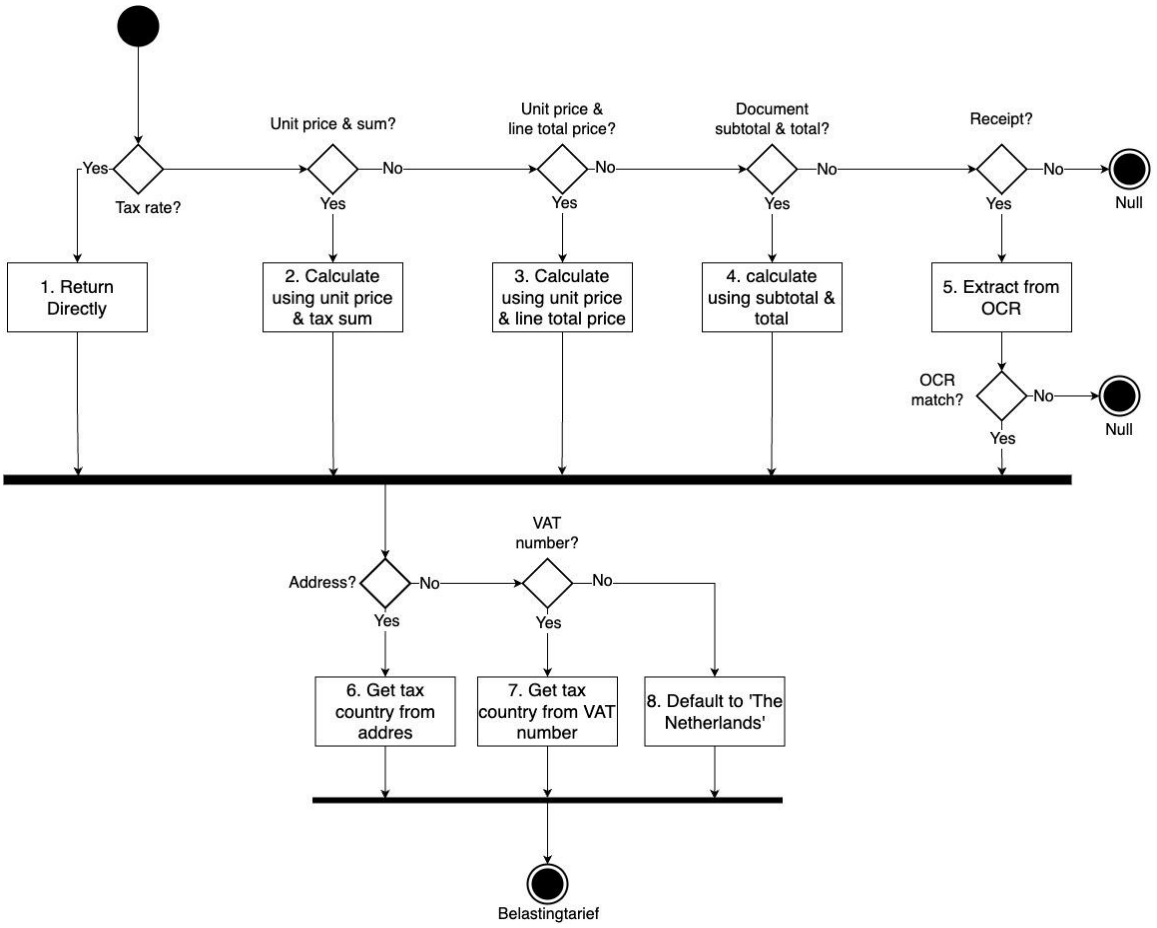
```
# In case of receipt -> company name + date of receipt
return receipt_reference if predict_json['type_of'] == 'receipt'
return predict_json.value('reference') if predict_json.value?('reference')
predict_json.value('purchase_order') if predict_json.value?('purchase_order')
```

7.6.1.1. Belastingstarief.

Het bepalen van het belastingtarief wordt gedaan door de 'predict_tax_rate' mutatie (Figuur 13, p. 37, IA). Het proces bestaat uit meerdere stappen die afhankelijk zijn van de geïdentificeerde velden.

Figuur 14

Activity diagram voorspelling belastingtarief



Het bovenstaande diagram illustreert de verschillende benaderingen voor het bepalen van het belastingtarief. Dit proces kan worden onderverdeeld in twee onderdelen: het bepalen van het belastingpercentage en het bepalen van het belastingland. Met behulp van deze informatie kan het uiteindelijke belastingtarief worden vastgesteld.

Hieronder worden de verschillende scenario's stapsgewijs toegelicht, waarbij de nummering overeenkomt met de scenario's in figuur 14:

1. Het belastingpercentage is direct beschikbaar. Er zijn geen verdere berekeningen nodig.
2. Het belastingpercentage wordt berekend op basis van de bruto stukprijs, de hoeveelheid en het belastingtarief.
3. Het belastingpercentage wordt berekend op basis van de bruto stukprijs, de hoeveelheid en het totaalbedrag (van de regel).
4. Het belastingpercentage kan niet worden bepaald op basis van individuele regels op het document. Het belastingtarief wordt berekend aan de hand van de totale prijs van het algehele document en het subtotaal van het algehele document.
5. In dit scenario wordt geprobeerd om het belastingtarief te extraheren met behulp van reguliere expressies (regex). Deze extractie kan alleen worden uitgevoerd op bonnetjes, omdat alleen het bonnetjesmodel een array retourneert met belastinginformatie in tekstformaat. Een voorbeeld van de teruggegeven tekst is: "21% BTW".

Nadat het belastingpercentage is vastgesteld, is de volgende stap het bepalen van het belastingland.

6. Het belastingland wordt bepaald op basis van het adres.
7. Het belastingland wordt bepaald op basis van het VAT-nummer. Binnen Europa beginnen VAT-nummers altijd met de shortcode van het land in kwestie (Alvara, 2020).
8. Het belastingland kan niet gevonden worden. 'Nederland' wordt als belastingland teruggegeven.

Op basis van het belastingland en het belastingpercentage wordt vervolgens het bijbehorende belastingtarief in het systeem gezocht. Indien niet gevonden, wordt 'nill' teruggegeven.

7.6.2. De data tonen

Nu de gegevens gemapt zijn, kunnen de voorspellingen getoond worden. De '**add_suggestion**'-mutatie (Figuur 13, p. 37, G) wordt gebruikt om de voorspellingen zichtbaar te maken en op te slaan in de database. De '**document_attachment_analyser**' mutatie (Figuur 13, H) fungeert hierin als interface tussen de 'add_suggestion' mutatie en de losse 'predict' mutaties (Figuur 13, I).

De onderstaande code controleert of de voorspelde prijzen inclusief of exclusief belasting zijn. Door het attribuut 'prices_are_incl_tax' van het documentmodel aan te passen, wordt het resultaat aan de gebruiker getoond.

```
total_price = document_lines.sum do |line|
      document_line_quantity(line).to_f * line[:unit_price]
```

```
      end

# Is incl tax if the sum of the document line prices is equal to the total price
inc_tax? = total_price == document_details[:total_price]

# Save result and show
document.prices_are_incl_tax = inc_tax?
```

7.6.3. Suggesties ongedaan maken

Zowel in de huidige situatie als in de nieuwe situatie heeft de gebruiker de mogelijkheid om voorspellingen ongedaan te maken. In de nieuwe situatie is ervoor gekozen om hiervoor een aparte mutatie te creëren, zodat de logica gescheiden blijft. Deze mutatie wordt gebruikt om de ingevulde voorspellingen te verwijderen, zodat de gebruiker de details handmatig kan invullen. De onderstaande code illustreert hoe de voorspellingen van de documentregels verwijderd worden:

```
document.details.clear
# Make sure there is at least one empty detail
document.details.push(empty_detail)
```

7.7. Uitdagingen

Over het algemeen verliep de implementatie soepel, wel waren er enkele uitdagingen. In dit hoofdstuk worden deze uitdagingen toegelicht.

Voor het manipuleren van de afbeelding voor het classificatiemodel was aanvankelijk gekozen voor de Numpy.rb library (Murata, n.d.). Dit bleek later een slechte keuze te zijn, omdat deze bibliotheek afhankelijkheid bleek te hebben met de Python programmeertaal. Dit veroorzaakte problemen omdat het build-systeem van Moneybird geen ondersteuning voor Python heeft. Hierdoor faalde alle testen ondanks dat het lokaal wel werkte.

Het probleem is opgelost door de implementatie te herschrijven met de numo library (Ruby Numo, n.d.). Numo biedt vergelijkbare mogelijkheden als Numpy.rb maar heeft geen afhankelijkheden van andere programmeertalen.

Verder zijn er enkele velden waarvoor geen voorspellingen worden teruggegeven maar alleen tekst. Er is voor gekozen om voor deze gevallen terug te vallen op reguliere expressies. Dit is bijvoorbeeld het geval bij bonnetjes waarop "btw laag" of "btw hoog" vermeld staat, evenals wanneer het belastingpercentage in tekstformaat wordt teruggegeven.

Hieronder volgt een voorbeeld van hoe het belastingpercentage uit de tekst wordt gehaald en omgezet wordt naar een integer.

```
# Finds the percentage, removes the '%' and casts that to an integer
text.scan(/\d+(?:[.,]\d+)?%/)&.first&.gsub(/[%\s,.].*/, "")&.to_i
```

Een soortgelijke aanpak is gekozen voor het ophalen van het KvK-nummer:

```
regex = /k\.? ?[uvwy]\.? ?k|\bh\. ?r\b|\bc\.? ?o\.?
?c\b|commerce|koophandel|handelsregist|registreerd|registered|ingeschreven/i
```

De bovenstaande reguliere expressie zoekt naar patronen zoals 'k.v.k' en 'c.o.c' (chamber of commerce), evenals specifieke patronen zoals 'kamer van koophandel'.

7.8. Eindresultaat

Het eindproduct voldoet aan alle vooraf opgestelde 'must-have'- en 'should-have'-requirements, zoals gedocumenteerd in bijlage K. Er waren geen aanpassingen aan de requirements en de opdracht nodig om het eindproduct succesvol te implementeren.

In vergelijking met de vorige implementatie biedt het eindproduct nu de functionaliteit om documentregels en hun bijbehorende kenmerken te voorspellen. Deze kenmerken omvatten onder andere de beschrijving van het product of de dienst, de hoeveelheid, de kosten en het BTW-tarief. Bovendien wordt bepaald of het BTW-tarief binnen of buiten Europa moet worden toegepast.

Daarnaast is er een classificatiemodel getraind en ontwikkeld, waardoor het juiste model van Form Recognizer automatisch wordt gekozen en de gebruiker niet langer handmatig het documenttype hoeft te selecteren.

Dit alles samen heeft geleid tot aanzienlijk minder interacties die nodig zijn om een document in te boeken, zoals zichtbaar in bijlage l.

8. Testen

8.1. Inleiding

De implementatie is getest volgens de principes van Test Driven Development (TDD). Hierbij zijn tests geschreven voordat nieuwe code is toegevoegd. Als de tests falen, wordt pas nieuwe code geschreven en vervolgens wordt opnieuw getest. Test Driven Development helpt ontwikkelaars zich te concentreren op de gestelde requirements nog voordat het daadwerkelijke ontwikkelproces is gestart (Pilchenko, 2021).

Het RSpec-framework (RSpec, n.d.) wordt gebruikt voor het schrijven van de tests, waarbij de implementatie van de vernieuwde documentherkenning is getest met behulp van Rspec integratietesten.

De geschreven tests hebben altijd een direct of indirect verband met de requirements. Onderzoek toont bijvoorbeeld aan dat een classificatiemodel nodig is om het juiste Form Recognizer-model aan te spreken. In dit geval heeft het classificatiemodel een indirecte invloed op het voldoen aan meerdere requirements. Raadpleeg bijlage K voor een overzicht van de requirements die gedekt worden door de tests.

8.2. Classificatie

Het classificatiemodel (zie hoofdstuk 7.3) is getest met integratietesten waarbij afbeeldingen van verschillende bonnetjes en facturen zijn gebruikt. Met de testen wordt getest of het model in staat is om het documenttype correct te classificeren en of de fouten goed afgehandeld worden.

```
it "Classifies type of a bad picture of an receipt" do
   params[:stored_file] = bad_receipt
   transformed_file = transform.run!(params)
   transformed_file_json = JSON.parse(read_file(transformed_file))
   expect(transformed_file_json["type_of"]).to eq("receipt")
 end
```

Bovenstaande test heeft betrekking op requirement F2 (zie hoofdstuk 2.1.).

De bovenstaande test voert de classificatie uit op een bonnetje dat onleesbaar is en wordt vastgehouden. De test verwacht dat het model het bonnetje ondanks de gebreken markeert als bonnetje.

8.3. Voorspelling (Form Recognizer API)

Bij Moneybird wordt de VCR GEM (benoittgt, n.d.) gebruikt om externe API's te testen. De VCR GEM legt de HTTP-interacties vast tijdens het testen, zodat de API niet telkens opnieuw hoeft te worden aangeroepen wanneer een test wordt uitgevoerd.

De 'predict' stap (Figuur 9, p. 30, Stap 5) is getest met als invoer diverse bonnetjes en facturen om te verifiëren of de data transformer (zie hoofdstuk 7.5) de voorspellingen correct verwerkt, ook in situaties waarin het document niet herkend kan worden.

```
it "request should return error if input is incorrect" do
   transform_file_params[:transformations] = bad_transformations
   allow(circuit_breaker).to receive(:request).and_call_original
   transformed_file = Transform::TransformFile.run!(transform_file_params)
   expect(JSON.parse(read_file(transformed_file))).to have_key('error')
 end
```

Bovenstaande test heeft betrekking op requirement F2 en F10 (zie hoofdstuk 2.1.).

In de bovenstaande test wordt een onverwachte invoer gegeven aan de 'predict' stap. De verwachting is dat het resultaat een JSON-object bevat met de key 'error'. In Moneybird wordt de aanwezigheid van de 'error'-key gebruikt om een fout in de voorspelling op te vangen.

8.4. Mappen van de voorspellingen

Net als dat er voor elk te herkennen veld een mutatie geschreven is (zie hoofdstuk 7.6.1), zijn er ook voor elk te herkennen veld testen geschreven. Deze testen testen of alle mogelijke scenario's en fouten op een goede manier worden afgevangen.

```
it 'Will output the purchase order id, if invoice id is not found' do
   invoice_expected_values.set('reference', nil)
   invoice_expected_values.set('purchase_order', 'value' => 'PO1212')
   expect(mutation.run!(predict_json: invoice_expected_values)).to eq("PO1212")
 end
```

Bovenstaande test heeft betrekking op requirement F5.

Bij het verkrijgen van het documentkenmerk wordt eerst gezocht naar het veld 'reference'. Als er geen voorspelling voor 'reference' wordt gevonden, wordt er vervolgens gekeken naar het veld 'purchase_order'.

In de bovengenoemde test wordt de waarde van 'reference' opzettelijk ingesteld op 'nil' om te testen of in een dergelijke situatie de waarde van 'purchase_order' als alternatief wordt gebruikt.

8.5. Mappen van voorspellingen met betrekking tot belasting

Voor het herkennen van het belastingtarief zijn veel scenario's mogelijk (Figuur 14, p. 38). Voor elk van deze scenario's zijn tests opgesteld om te verifiëren of alle situaties en fouten correct worden afgehandeld.

De tests die in dit hoofdstuk worden besproken, zijn gerelateerd aan de requirements F1, F10 en F11 (zie hoofdstuk 2.1.).

```
 it 'Will output 21% tax (from quantity, unit_price, tax_total)' do
    values = [
      'quantity' => { 'value' => 2 },
      'unit_price' => { 'value' => 10.0 },
      'tax_total' => { 'value' => 4.20 }
    ]
    invoice_expected_values.set("lines", values)

    output = mutation.run!(predict_json: invoice_expected_values,
administration: administration)
    expect(output[:lines][0][:tax_id]).to eq(high_tax)
  end
```

In de bovenstaande test worden de hoeveelheid ('quanty'), de stukprijs ('unit price') en het totale belastingtarief ('tax total') meegegeven aan de test. De test verwacht dat op basis van deze informatie het hoge belastingtarief (21%) wordt teruggegeven.

```
 it 'Will output 9% tax (receipt, regex extraction (BTW laag) text' do
    values = [
      'content' => 'Btw laag $ 12,50'
    ]
    receipt_expected_values.set("tax_details", values)

    output = mutation.run!(predict_json: receipt_expected_values,
administration: administration)
    expect(output[:lines][0][:tax_id]).to eq(reduced_tax)
  end
```

In de bovenstaande test wordt het belastingtarief als tekst ('Btw laag $ 12,50') meegegeven. De test verwacht dat de uitkomst het verlaagde btw-tarief (9%) is.

```
 it 'Will output 9% tax detail (from tax rate)' do
    invoice_expected_values.set("lines", ["tax_rate" => { "value" => "9%" }])
    output = mutation.run!(predict_json: invoice_expected_values,
administration: administration)
    expect(output[:lines][0][:tax_id]).to eq(reduced_tax)
  end
```

De bovenstaande test simuleert een situatie waar het belastingpercentage meteen gevonden wordt. De test verwacht dat de uitkomst het verlaagde BTW-tarief (9%) is.

9. Conclusie

In dit onderzoek is antwoord gezocht op de vraag: `*Hoe kan het algoritme voor documentanalyse verbeterd worden zodat deze de algehele structuur van een document beter herkent?*`

Om de structuur van een document te herkennen is een bestaand of zelf getraind model voor documentherkenning nodig. Het trainen van een model heeft niet de gewenste resultaten opgeleverd en daarom is gekozen om gebruik te maken van een framework.

Om dit te automatiseren is gekeken naar verschillende bestaande machine learning oplossingen. Hierbij zijn verschillende frameworks vergeleken. Hieruit bleek dat Microsoft Form Recognizer de beste prijs-kwaliteitverhouding biedt. Daarnaast zijn de frameworks praktisch met elkaar vergeleken met een benchmark, uit de resultaten is gebleken dat Azure Form Recognizer de meest nauwkeurige document voorspellingen levert, met een algemene nauwkeurigheidspercentage van 94.21%.

In tegenstelling tot de oude implementatie kan de nieuwe implementatie de factuurregels herkennen. Hoe meer factuurregels, hoe meer interacties er bespaard worden. Hiermee is de doelstelling: *"Het doel van de afstudeeropdracht is om in 5 maanden het documentanalyse algoritme zo veel mogelijk te verbeteren zodat de gebruiker minder interacties nodig heeft om een document in te boeken."* behaald.

Azure Form Recognizer maakt gebruik van specifieke modellen voor bonnetjes en facturen, daarom moet op voorhand bekend zijn welk document type er geüpload wordt. hiervoor is een classificatiemodel getraind en geïmplementeerd (zie bijlage I). Door deze toevoeging wordt automatisch het juiste model gekozen.

Kortom kan gesteld worden dat een oplossing met als basis de Form Recognizer API en het gebruik van een classificatiemodel een effectieve en kosteneffectieve oplossing is voor het voorspellen van de algehele structuur van een document.

10. Aanbevelingen

Met de huidige implementatie is de ondernemer al aanzienlijk ontlast, maar er zijn nog enkele aanbevelingen die zowel Moneybird als de ondernemer verder zouden kunnen helpen.

10.1. Voorspelling op de achtergrond uitvoeren

De praktijk heeft aangetoond dat het in de nieuwe situatie ongeveer 3 tot 4 seconden duurt (afhankelijk van de documentkwaliteit) voordat de voorspelling gereed is, wat langer is dan de oorspronkelijke 1 seconde in de oude situatie. De vertraging wordt veroorzaakt door de Form Recognizer API, een extern component waar Moneybird slechts beperkte invloed op heeft. Het gevolg is dat hoewel de voorspelling flink verbeterd is, de wachttijd voor de gebruiker iets is toegenomen.

Om deze vertraging voor de ondernemer te verminderen, is het een optie om de voorspelling op de achtergrond uit te voeren terwijl de ondernemer nog bezig is met het selecteren van het contact. Hoewel de voorspelling nog steeds even snel is, lijkt de wachttijd korter voor de gebruiker.

10.2. Ontwikkelingen in AI gebied

Hoewel er tot nu toe nog geen geschikte dataset of *pre-trained* model is die aan de use-case voldoet, zijn er veelbelovende ontwikkelingen gaande op het gebied van kunstmatige intelligentie. Websites zoals Kaggle (Kaggle, n.d.) en Huggingface (Huggingface, n.d.) brengen regelmatig nieuwe open source- modellen en datasets uit, waardoor er in de toekomst mogelijk een passende oplossing beschikbaar komt.

10.3. Ingebruikname

Op dit moment wordt de nieuwe implementatie aangeboden als een bèta-functie, die alleen beschikbaar is voor gebruikers binnen de organisatie Moneybird.

Voor een volledige ingebruikname wordt aanbevolen om de functie geleidelijk uit te rollen naar de verschillende administraties. Op deze manier kunnen de kosten en prestaties in een gecontroleerde omgeving bijgehouden worden.

In de code is een boolean-vlag opgenomen om te controleren of de gebruiker de bèta-functie heeft ingeschakeld. Om het product volledig in gebruik te nemen, moet de bèta-vlag worden verwijderd en de oude, irrelevante code verwijderd worden.

11. Beperkingen

Benchmark

De benchmark vergeleek verschillende frameworks door hun output te vergelijken met een dataset van verwachte gegevens. Aangezien deze dataset handmatig is opgesteld, bestaat de mogelijkheid dat er menselijke fouten in zijn geslopen die van invloed kunnen zijn op de resultaten van de benchmark. Het resultaat van de benchmark moet worden gezien als goede indicatie van de accuraatheid maar is niet perfect.

Trainen model voor documentherkenning

Het trainen van een eigen model bleek niet succesvol te zijn. Dit kan mogelijk toegeschreven worden aan de relatief kleine dataset met weinig variatie. Verder is de data geautomatiseerd verwerkt, hierdoor kan geen garantie worden gegeven dat de trainingsdata geen fouten bevatten. Deze factoren hebben mogelijk bijgedragen aan het gebrek aan succes bij het trainen van het eigen model.

Vergelijking van frameworks

De frameworks die zijn opgenomen in de vergelijking, zijn frameworks voor documentherkenning die vaak genoemd worden in literatuur. Het zou kunnen zijn dat er kleinere, minder bekende frameworks bestaan die ook goede resultaten behalen. Een struikelblok hierin zijn de kosten die ontstaan om dergelijke frameworks te testen.

12. Reflectie

In het afgelopen halfjaar heb ik veel bij Moneybird geleerd. Moneybird maakt gebruik van het Ruby on Rails framework. Ik had nog geen ervaring met het framework, daarom heb ik de eerste twee weken van mijn afstudeerproject besteed aan het leren van het framework en het verkennen van de specifieke eigenschappen van Ruby on Rails. Hier heb ik veel profijt van gehad. Ik merkte dat ik de concepten en de benoemingen die binnen Moneybird gebruikt worden hierdoor snel doorhad.

Vanuit school ben ik gewend om met Scrum te werken. Het werken met Kanban was even wennen, maar omdat het veel overeenkomsten met Scrum heeft, voelde het al snel vertrouwd. Wat ik de volgende keer beter zou doen, is het specifieker *scopen* van de tickets. In het begin van het afstudeertraject waren mijn wijzigingen in de code groot. Waardoor het voor mijn begeleiders moeilijk was om mijn werk goed te beoordelen.

Bij Moneybird was ik onder de indruk van de begeleiding die ik kreeg. Mijn begeleiders waren kritisch en behulpzaam, wat resulteerde in een positieve omgeving waarin zelfontwikkeling werd gestimuleerd. Ik heb veel geleerd over de *best-practises* van Ruby, maar ook over het bedrijf Moneybird en zelfs enkele basisprincipes van boekhouden. De begeleiding heeft bijgedragen aan mijn persoonlijke groei en ik ben erg tevreden met de ervaring die ik daarbij heb opgedaan.

In de onderzoeksfase is er achteraf gezien te veel nadruk gelegd op het onderzoeken en trainen van machine learning modellen voor documentanalyse. Er is aanzienlijke tijd besteed aan het voorbereiden van de datasets en het opzetten van de machine learning frameworks. Hoewel er geleidelijk vooruitgang is geboekt (van 12% naar 34% nauwkeurigheid), bleef het gewenste resultaat uit. De volgende keer zou ik dergelijke situatie vermijden door ook voor de onderzoeksfases Kanban of Scrum toe te passen. Dit stelt me in staat om de voortgang beter te volgen en mezelf strikte deadlines te stellen.

De implementatie verliep over het algemeen voorspoedig. Dankzij het ontwerp en de risicoanalyse had ik een duidelijk plan voor de implementatie en wist ik welke stappen ik moest nemen om toekomstige risico's te vermijden.

De implementatie van het classificatiemodel verliep helaas minder voorspoedig. Ik had aanvankelijk gekozen voor een bibliotheek die afhankelijk bleek te zijn van de programmeertaal Python, waardoor alle tests faalden op het build-systeem. Het gevolg hiervan is dat ik veel tijd kwijt was met het vinden van het probleem en het herschrijven van de implementatie met een ander framework. Achteraf gezien had ik me beter moeten verdiepen in de afhankelijkheden van de bibliotheek.

Daarentegen was ik heel blij met de conclusie van zowel de benchmark als de vergelijking van frameworks. De benchmark gaf tastbare resultaten, ik merkte dat mijn collega's baat hadden bij deze resultaten. Bovendien werd de kostenraming die is opgesteld, gewaardeerd vanuit een zakelijk perspectief. Ik heb daardoor het gevoel dat ik een weloverwogen keuze heb gemaakt voor Moneybird.

Al met al kan ik zeggen dat ik veel heb geleerd en persoonlijk gegroeid ben. Dankzij de inhoudelijke feedback, heb ik geleerd hoe ik mijn code efficiënter kan schrijven. Ik durf te beweren dat dit mij heeft geholpen om een betere software ontwikkelaar te worden. Ook ben ik blij dat ik het product heb kunnen integreren in het Moneybird systeem en dat de nieuwe implementatie daadwerkelijk in productie gaat komen.

13. Bronnenlijst

Amazon. (n.d.-a). *Amazon Textract FAQs*. Amazon.com. Retrieved May 19, 2023, from https://aws.amazon.com/textract/faqs/

Amazon. (n.d.-b). *Texttract pricing*. Amazon.com. Retrieved March 13, 2023, from https://aws.amazon.com/textract/pricing/

Azure Applied AI Services. (2023, March 17). *Create and use managed identities with Form Recognizer - Azure Applied AI Services*. Azure Applied AI Services. https://learn.microsoft.com/en-us/azure/applied-ai-services/form-recognizer/managed-identities

benoittgt. (n.d.). *vcr: Record your test suite's HTTP interactions and replay them during future test runs for fast, deterministic, accurate tests*. Retrieved May 20, 2023, from https://github.com/vcr/vcr

Biswal. (2021, April 27). *The complete guide on overfitting and underfitting in machine learning*. Simplilearn.com; Simplilearn. https://www.simplilearn.com/tutorials/machine-learning-tutorial/overfitting-and-underfitting

C3. (2021, April 7). *Ground truth*. C3 AI. https://c3.ai/glossary/machine-learning/ground-truth/

cloveai. (n.d.). *SynthDoG 🐶: Synthetic Document Generator*. Retrieved March 30, 2023, from https://github.com/clovaai/donut

Diepenmaat, J. (2021, September 28). Moneybird is ISO 27001 gecertificeerd. *Moneybird*. https://www.moneybird.nl/blog/moneybird-is-iso-27001-gecertificeerd/

Dullin, T. (n.d.). *Machine Learning for Unstructured Document Analysis: a guide*. Machine Learning for Unstructured Document Analysis. Retrieved April 3, 2023, from https://kili-technology.com/data-labeling/machine-learning/machine-learning-for-unstructured-document-analysis

Google Developers. (n.d.-a). *Machine learning glossary*. Google Developers. Retrieved April 3, 2023, from https://developers.google.com/machine-learning/glossary#fine-tuning

Google Developers. (n.d.-b). *The size and quality of a data set*. Google Developers. Retrieved March 13, 2023, from https://developers.google.com/machine-learning/data-prep/construct/collect/data-size-quality

HBO-I. (n.d.). *ICT research methods*. Ictresearchmethods.Nl. Retrieved May 23, 2023, from https://ictresearchmethods.nl/

Huggingface. (n.d.). *Hugginface - Models*. Huggingface.Co. Retrieved May 31, 2023, from https://huggingface.co/models

Kaggle. (n.d.). *Kaggle - models*. Kaggle.com. Retrieved May 31, 2023, from https://www.kaggle.com/models

Kim, G., Hong, T., Yim, M., Nam, J., Park, J., Yim, J., Hwang, W., Yun, S., Han, D., & Park, S. (2021). OCR-free document understanding transformer. In *arXiv [cs.LG]*. http://arxiv.org/abs/2111.15664

Mavuduru, A. (2021, February 17). *What is GPT-3 and why is it so powerful?* Towards Data Science. https://towardsdatascience.com/what-is-gpt-3-and-why-is-it-so-powerful-21ea1ba59811

Microsoft. (n.d.). *Pricing - Form Recognizer API*. Microsoft.com. Retrieved March 13, 2023, from https://azure.microsoft.com/en-gb/pricing/details/form-recognizer/

Microsoft. (2022, May 12). *Bestedingslimiet voor Azure*. Microsoft.com. https://learn.microsoft.com/nl-nl/azure/cost-management-billing/manage/spending-limit

Microsoft. (2023a, February 14). *Language support - Form Recognizer - azure applied AI services*. Microsoft.com. https://learn.microsoft.com/en-us/azure/applied-ai-services/form-recognizer/language-support?view=form-recog-3.0.0

Microsoft. (2023b, March 13). *Form Recognizer quotas and limits - Azure Applied AI Services*. Microsoft.com. https://learn.microsoft.com/en-us/azure/applied-ai-services/form-recognizer/service-limits?view=form-recog-3.0.0

Microsoft. (2023c, March 13). *Form Recognizer service quotas and limits*. Microsoft. http://ttps://learn.microsoft.com/en-us/azure/applied-ai-services/form-recognizer/service-limits

Microsoft. (2023d). *Service Level Agreement for Microsoft Online Services*. Azure Cognitive Services. https://wwlpdocumentsearch.blob.core.windows.net/prodv2/OnlineSvcsConsolidatedSLA(WW)(English)(May2023)(CR).docx?sv=2020-08-04&se=2123-05-31T19:35:30Z&sr=b&sp=r&sig=XAJij3WfVjtV8P3xrHgrMTVj5oSzlO3aEVmnWJenykM%3D

Mohammad, S. (2020). *Real World Documents Collections* [Data set]. https://www.kaggle.com/datasets/shaz13/real-world-documents-collections

Moneybird. (n.d.). *Wil je weten wie de mensen achter Moneybird zijn?* Moneybird. Retrieved May 23, 2023, from https://www.moneybird.nl/over-ons/

Moneybird. (2023). *Afstudeeroprdacht: Betere bonnetjesscanner & factuurherkenning*.

Murata, K. (n.d.). *numpy.rb: Numpy wrapper for Ruby*.

Nitinme, S. D. V.-A. (2022, July 19). *Data, privacy, and security for Form Recognizer*. Microsoft.com. https://learn.microsoft.com/en-us/legal/cognitive-services/form-recognizer/fr-data-privacy-security

Ordelman, S. (n.d.). [personal communication].

Pilchenko, E. (2021, July 8). *How test-driven development help business: benefits of TDD*. Forte Group. https://fortegrp.com/test-driven-development-benefits/

RSpec. (n.d.). *Rspec*. https://rspec.info/

Ruby Numo. (n.d.). *Ruby Numo (NUmerical MOdule)*. Retrieved June 13, 2023, from https://github.com/ruby-numo

Singh, S. (2020, July 1). *MVC framework: A modern web application development approach and working*. Cloudfront.net. https://d1wqtxts1xzle7.cloudfront.net/61982966/IRJET-V7I11120200203-92100-xttlvw-libre.pdf

Tensorflow. (n.d.). *Image classification*. TensorFlow. Retrieved May 18, 2023, from https://www.tensorflow.org/tutorials/images/classification

van Essen, T., & de Weerdt, M. (n.d.). *De impact van algoritmes*. TU Delft. Retrieved May 17, 2023, from https://www.tudelft.nl/stories/articles/de-impact-van-algoritmes