



Faculteit Bedrijf en Organisatie

Facturatie optimaliseren door gebruik te maken van Optical Character recognition (OCR) en Machine Learning (ML): onderzoek en proof of concept

Remi Mestdagh

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

Promotor:
Benjamin Vertonghen

Instelling: Append

Academiejaar: 2020-2021

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Facturatie optimaliseren door gebruik te maken van Optical Character recognition (OCR) en Machine Learning (ML): onderzoek en proof of concept

Remi Mestdagh

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

Promotor:
Benjamin Vertonghen

Instelling: Append

Academiejaar: 2020-2021

Tweede examenperiode

Woord vooraf

Als beginnende softwareontwikkelaar ben ik net zoals velen gepassioneerd door de veelzijdigheid van het vak. Maar er zijn weinig onderwerpen in de IT-sector die zo tot de verbeelding spreken als machine learning en OCR. Het zijn beide oplossingen voor problemen waar de mens het langst de bovenhand had. De manier waarop software van voor de computer betekenisloze data omzet naar bruikbare informatie ervaar ik als een kunst.

Ik wil eerst en vooral mijn promotor Benjamin Vertonghen bedanken voor zijn aanstekelijk enthousiasme die mij telkens weer motiveerde. Ook de ontvangen feedback was goud waard. Vervolgens wens ik mijn gezin en vriendin te bedanken voor hun steun doorheen mijn hele opleiding.

Samenvatting

De groeiende nood naar maatwerk tijdens het programmeren zowel als bestaande bottlenecks tijdens het boekhouden dragen bij tot het startschot van deze paper. Het bestaan en de toepassing van software in eigen applicatie die facturen automatisch converteert in de relevante data in sleutel-waardeparen werd onderzocht.

Ten eerste werd de technologie achter OCR besproken. Verder werd in dit onderzoek de meest prominente open source OCR technologie, Tesseract, met de in de literatuur beschouwd als beste betalende variant vergeleken op vlak van accuraatheid. Abbyy komt hier als winnaar uit de boot. De hoge performanties zorgen er echter niet voor dat de gebondenheid aan een maandelijkse kostprijs acceptabel is. Vervolgens wordt voor een combinatie van machine learning en OCR technologie geopteerd om een oplossing aan de casus te bieden. De literatuur wijst uit dat Form Recognizer wellicht de best mogelijke optie is. Hiermee worden enkele experimenten uitgevoerd om de accuraatheid te evalueren. Deze zijn uiterst positief. Ten slotte werd een proof-of-concept gemaakt om aan te tonen dat de casus een oplossing aangeboden kreeg.

De studie heeft zijn limieten bereikt in het kader van de voorziene uren, enkele andere pistes kunnen nog onderzocht worden. De impact van fonts op de nauwkeurigheid van OCR software kan bekijken worden. In de toekomst kan het bouwen van een applicatie met een grotere scope een uitdaging vormen. Software die een groot aantal verschillende facturen kan herkennen, dient geëvalueerd te worden. Eventuele implementaties van dergelijke oplossingen aan de hand van louter open source technologie is ook interessant. Ook de mogelijkheid om een pipeline van AWS services op te zetten om een gelijkaardig resultaat te bekomen, oogt interessant.

Inhoudsopgave

1	Inleiding	13
1.1	Probleemstelling	13
1.2	Onderzoeksvraag	14
1.3	Onderzoeksdoelstelling	14
1.4	Opzet van deze bachelorproef	15
2	OCR onder de loep	17
2.1	Onderzoek naar OCR	17
2.2	Technologieën binnen OCR	18
2.3	Nauwkeurigheid	20
2.3.1	Globale foutenpercentage	20
2.3.2	Weigeringspercentage	21
2.3.3	Herkenningspercentage	21

2.3.4	Betrouwbaarheidsniveau	21
2.4	Eigenschappen en hun invloed	21
2.4.1	Taalondersteuning	21
2.4.2	Aanwezigheid van technieken	22
2.4.3	Bronmateriaal	22
2.5	Beschikbare OCR technologieën	22
2.5.1	Requirements	23
2.5.2	Longlist	24
2.5.3	Marktaanbod	25
2.5.4	Open source route	25
2.6	Experimenten	26
2.6.1	Methodologie van de experimenten	31
2.6.2	Tesseract	34
2.6.3	ABBYY	38
2.7	Vergelijking betalende en open source software	40
2.8	Betekenis voor de casus	40
2.9	Conclusie	41
3	Machine Learning onder de loep	43
3.1	Wat is Machine Learning?	43
3.1.1	Casus	44
3.2	Classificatie	44
3.2.1	Naïve Bayes	44
3.2.2	Dichtste-buur classificatie	45

3.3 Beschikbare ML technologieën	45
3.3.1 Requirements	45
3.3.2 Longlist	46
3.3.3 AWS Textract	47
3.3.4 Azure Form Recognizer	48
3.3.5 Open source route	49
3.3.6 Keuze	50
3.4 Nauwkeurigheid	50
3.4.1 Initiële accuraatheid	50
3.4.2 Accuraatheid met naverwerking	50
3.5 Experimenten	50
3.5.1 Methodologie van de experimenten	51
3.5.2 Benodigdheden	51
3.5.3 Opbouw	52
3.5.4 Analyse	56
3.5.5 Nauwkeurigheid meten	58
3.5.6 Evaluatie	61
3.6 Conclusie en betekenis voor de casus	62
4 Implementatie	63
4.1 Aanspreken in C#	63
4.1.1 Software	63
4.1.2 Verklaring	64
4.2 Conclusie	65

5 Conclusie	67
5.1 Bijdrage	67
5.1.1 Onderzoeksvragen	67
5.1.2 Ondervindingen	69
5.1.3 Verder onderzoek	69
A Onderzoeksvoorstel	71
A.1 Introductie	71
A.2 State-of-the-art	72
A.2.1 Wat is OCR	72
A.2.2 Hedendaagse OCR voor ontwikkelaars	72
A.3 Methodologie	72
A.4 Verwachte resultaten	73
A.5 Verwachte conclusies	74
B Output OCR experimenten	75
B.1 Tesseract	75
B.1.1 Lage DPI	75
B.1.2 Beste omstandigheden	76
B.1.3 Taalondersteuning	77
B.2 ABBYY	78
B.2.1 Beste omstandigheden	78
Bibliografie	79
Woordenlijst	83

Lijst van figuren

2.1	De-skew voorbeeld (Ada, 2013a)	18
2.2	Despeckle voorbeeld(Ada, 2013b)	19
2.3	Belang van dpi	22
2.4	Voorbeeld functioneel project	29
2.5	Nederlands toevoegen	30
2.6	Toevoegen checkbox voor Nederlands	30
2.7	Factuur met lage dpi	32
2.8	Factuur met hoge dpi	32
2.9	Input voor experimenten in onderdeel 2.6	33
2.10	Herkenningspercentage beste omstandigheden	34
2.11	Herkenningspercentage zonder taalondersteuning	35
2.12	Boxplot herkenningspercentage bij lage dpi	37
2.13	Boxplot herkenningspercentage beste omstandigheden (ABBYY)	39
3.1	Azure labeling tool	53
3.2	Azure labeling tool	54
3.3	Nieuw project	54
3.4	Nieuwe verbinding	55

3.5 Voorbeeld document met labels	56
3.6 Eerste analyse	57
3.7 Analyse met fout	58
3.8 Nauwkeurigheid Azure Form Recognizer	59
3.9 Factuur a	59
3.10 Factuur b	60
3.11 Nauwkeurigheid Azure Form Recognizer na veranderingen in model 60	
3.12 Vergelijking output)	61
4.1 Voorbeeldantwoord PoC	65
B.1 Resultaat ABBYY FineReader	78

1. Inleiding

Dit onderdeel vormt een inleiding op de paper. Het bevat eerst en vooral de te onderzoeken probleemstelling. Vervolgens wordt een overzicht van de relevante onderzoeks vragen weergegeven. Daarna wordt de doelstelling van dit onderzoek besproken. Ten slotte wordt de opzet van de paper aangekaart.

1.1 Probleemstelling

Het doel van deze paper is om een veelvoorkomende bottleneck die boekhouders dagelijks ervaren te bespreken en eventuele oplossingen voor te stellen. Niet alleen boekhouders, maar ook softwareontwikkelaars worden ermee geconfronteerd.

Boekhouders en particulieren verkrijgen facturen en andere documenten vandaag vaak op papier of in digitale vorm. In een utopische wereld zou men een binnenkomende factuur kunnen inscannen en zou de inhoud meteen verwerkt worden. De velden komen meteen in de correcte tabel van een databank terecht. In werkelijkheid vormt dit een heuse uitdaging. Een scan is op zich slechts een foto en er is software nodig om de informatie te ontcijferen. Zelfs wanneer een omzetting naar een tekstdocument succesvol is, is het niet bekend voor de computer welke tekstwaarde overeenkomt met een categorie zoals 'totale prijs', 'btw-nummer'... Het manueel toekennen van deze 'categorieën' aan de tekstwaardes is uiterst tijdsintensief.

Wanneer men als programmeur een applicatie die een overzicht biedt van bestellingen, facturen enzovoort van bedrijven wil schrijven, wordt men vrijwel meteen geconfronteerd met de verschillen tussen uitgaande, dus van het bedrijf zelf, en binnenkomende facturen. Niet alleen de vorm van de factuur, waar men de indeling/lay-out mee beschrijft, maar

ook productcodes wijken af. Wanneer men door de boekhouding van eender welk bedrijf bladert, valt een breed scala aan kleuren en lay-outs op. Velden zoals de naam van het bedrijf zullen anders opgemaakt zijn. Hier blijft het echter niet bij. De door de leverancier gehanteerde productcodes worden niet door andere bedrijven overgenomen. Bedrijven kennen een eigen code toe aan producten die ze aanschaffen. Al deze afwijkingen maken het moeilijk om software te schrijven die op een efficiënte en nauwkeurige manier data op facturen kan verwerken.

Deze uitdaging vloeit enerzijds voort uit de manier waarop een factuur verzonden wordt. Wanneer een bedrijf een factuur ontvangt, zal deze vaak enkel op papier bestaan. Een logische volgende stap is deze inscannen. In het beste geval werd de factuur elektronisch bezorgd en kan men deze stap overslaan. Doorgaans beschikt men na deze stap over een pdf-bestand. De uitdaging bestaat er dan uit om de tekst en informatie te extraheren en correct in te delen. Tekst extraheren uit een foto of pdf kan met behulp van Optical character recognition (OCR). Een bijkomend voordeel is dat OCR zorgt dat grote bestanden zoals foto's en pdf niet meer opgeslagen moeten worden. De tekst op zich is in het algemeen enkele kilobytes kleiner. Met behulp van machine learning technieken kan men verdere indelingen maken en de verwerkingssoftware "slim" maken.

1.2 Onderzoeksvoraag

In deze paper zullen volgende onderzoeks vragen uitgewerkt worden:

- Wat is de technologie achter OCR?
- Welke stappen zijn nodig om een werkende applicatie te schrijven die OCR gebruikt om facturen te verwerken?
- Welke software is beschikbaar en wat zijn hun sterkes?
- Wat zijn de tekortkomingen van open source OCR software tegenover betalende?
- Welke OCR software heeft de hoogste snelheid en accuraatheid?
- Welke OCR software wordt aangeraden voor deze use case en waarom?
- Hoe personaliseerbaar zijn de beschikbare varianten?

1.3 Onderzoeksdoelstelling

De doelstelling van dit onderzoek is het vergelijken van beschikbare OCR technologieën en de eventuele mogelijkheid om ze in een C# applicatie te verwerken om het invoeren van facturen te vereenvoudigen. Nadat de performantie, mogelijkheid tot incorporatie in een eigen project en personaliseerbaarheid van de verscheidene beschikbare software onderzocht is, volgt een command line implementatie van een C# applicatie die facturen kan interpreteren. Dit is eerst en vooral een Proof of concept (PoC) waar men de mogelijkheden om een werkende implementatie als ontwikkelaar aantoon.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen OCR, op basis van een literatuurstudie. Vervolgens worden er enkele experimenten uitgevoerd om de invloed op de casus te duiden. De methodologie wordt toegelicht en de gebruikte onderzoekstechnieken worden besproken om een antwoord te kunnen formuleren op de onderzoeks vragen.

In Hoofdstuk 3 wordt een overzicht gegeven van de stand van zaken binnen Machine learning, op basis van een literatuurstudie. Vervolgens worden er enkele experimenten uitgevoerd om de invloed op de casus te duiden. De methodologie wordt toegelicht en de gebruikte onderzoekstechnieken worden besproken om een antwoord te kunnen formuleren op de onderzoeks vragen.

In Hoofdstuk 4 wordt in detail de implementatie van een C#-applicatie beschreven.

In Hoofdstuk 5, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeks vragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2. OCR onder de loep

Optical Character Recognition of OCR is een techniek die voor het eerst in de jaren 1920 toegepast werd. Het werd gebruikt om archieven die toen nog uit microfilm bestonden te doorzoeken. In de loop der jaren heeft OCR meerdere evoluties gekend en staat het zeker niet meer in zijn kinderschoenen. Vrijwel elke tech-gigant heeft ondertussen zijn variant uitgebracht. Deze pakketten bevatten ook vaak “slimme” functionaliteiten zoals patroonherkenning. Deze bewijzen hun nut tijdens het verwerken van facturen. Onder OCR verstaat men een combinatie van meerdere technologieën. Het gaat van het aanpassen van contrasten, het corrigeren van scheve scans tot het trainen van convolutionele neurale netwerken om accuraatheid te verhogen. Het begrijpen van de gebruikte technieken in een stuk OCR software is essentieel bij het maken van een goede keuze wanneer men deze wil toepassen in een eigen applicatie.

Daarna worden de maatstaven voor nauwkeurigheid van een OCR-pakket beschreven. Dan volgt een opsomming van eigenschappen van documenten en software die al dan niet een invloed op deze nauwkeurigheid hebben. Vervolgens worden verschillende pakketten vergeleken op basis van een long- en shortlist. Deze lijsten worden aan de hand van een literatuurstudie samengesteld. Wanneer de shortlist samengesteld is, volgen er enkele experimenten om de nauwkeurigheid en de mogelijkheid om de casus op te lossen te bepalen.

2.1 Onderzoek naar OCR

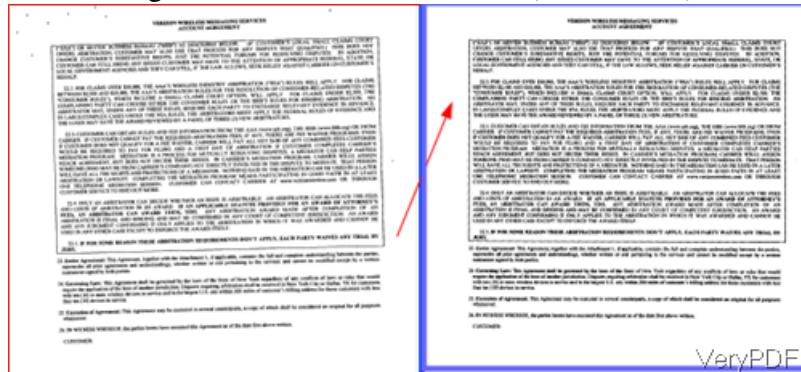
Onderzoek naar OCR is als een meander van een rivier. Instellingen en onafhankelijke onderzoekers splitsen zich af en onderzoeken manieren om betere resultaten te krijgen.

Deze verbeteringen worden dan eventueel later opgenomen in de rivier, de standaard voor OCR. Een voorbeeld hiervan speelde zich af in het begin van de jaren 1960 (Mori e.a., 1992). Iijima begon onderzoek naar mogelijke verbeteringen in de voorverwerking van bronmateriaal. Hij ontdekte dat het wazig maken van bronmateriaal de prestaties van het model verbeterden. De oorzaak hiervan is dat karakters met dezelfde waarde meer op elkaar beginnen te lijken. Iijima bereikte dit door het implementeren van een differentiaal- en integraalvergelijking op een hardwaremodel. De implementatie was een succes maar werd als te duur beschouwd. Het was echter niet voor niets geweest. Na de publicatie ervan in 1968 is zijn onderzoek een basis voor andere academici die het normalisatieprobleem onder handen willen nemen. (Sinha, 2002)

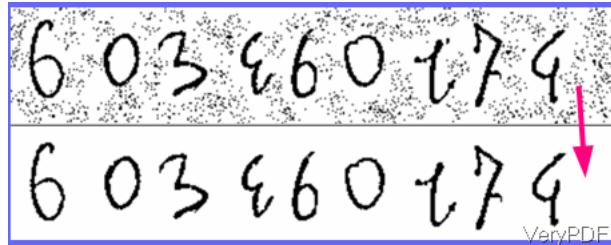
2.2 Technologieën binnen OCR

Zoals eerder vermeld, bestaat OCR uit meerdere technieken die samen het resultaat beïnvloeden. Het is mogelijk deze verder op te delen naargelang de fases waarin ze toegepast worden. Aan de start van het proces staat de voorverwerking die technieken zoals de-skew, despeckle, binarisation, line removal en line and word detection bevat. De-skew is in een groot aantal modellen een van de eerste stappen. Het corrigeert documenten die niet loodrecht ingescand werden. Het is gelimiteerd tot het verschuiven van het document met enkele graden. Despeckle of noise reduction verwijdert noise uit een document. Dit bestaat uit pixels die verkeerd geplaatst zijn. Een derde techniek is binarisation. OCR is in essentie een binair proces. Het herkent dingen die er ofwel zijn of niet zijn. De software zet elke niet-witruimte om naar zwart. Het is een eenvoudige manier om tekst te onderscheiden van de achtergrond. Vervolgens is line removal een techniek die essentieel is bij het verwerken van formulieren. Het verwijdert lijnen en simpele figuren. Deze staan het model vaak in de weg wanneer ze als karakter gezien worden. Ten slotte zal men met behulp van line and word detection pixels opdelen in woorden. („Optical Character Recognition (OCR) – How it works”, g.d.)

Figuur 2.1: De-skew voorbeeld (Ada, 2013a)



Figuur 2.2: Despeckle voorbeeld(Ada, 2013b)



De volgende fase is de daadwerkelijke tekstherkenning. Vandaag kunnen we twee kernalgoritmes onderscheiden om dit doel te verwezenlijken. Enerzijds hebben we matrix matching¹, beter bekend als patroonherkenning (Mantas, 1986). Dit algoritme vergelijkt de beelden in de input met gliefen. Een glief is een individuele markering of teken in een geschreven medium. Een glief kan bijvoorbeeld een enkele schrijfwijze voor het karakter "A" zijn. Meerdere gliefen samen vormen een grafeem. Deze techniek werkt het best met getypte tekst aangezien de kans dan het grootst is dat er een gelijkaardige glief gevonden wordt. Anderzijds hebben we feature extraction² (Deodhare e.a., 2005). Dit algoritme reduceert gliefen in kenmerken zoals lijnen, gesloten cirkels, richting van een lijn, snijpunten van lijnen. Op deze manier verlaagt de techniek de dimensionaliteit van het probleem. Dit zorgt voor efficiëntere verwerking. De features worden dan vergeleken met een vectorvoorstelling van een karakter. Het voordeel van vectoren is aanzienlijk. Het biedt een grote onafhankelijkheid van lettertype en -grootte aan. Matei e.a. (2013) raadt het gebruik van een k-dichtste buren algoritme³ aan. Hun onderzoek toonde een grote snelheid en accuraatheid aan, zelfs met een input van beelden van slechte kwaliteit.

Ten slotte volgt de naverwerking. De nauwkeurigheid van het model kan verhoogd worden als de uitvoer beperkt wordt door een lexicon, zoals het Nederlands woordenboek. Dit kan bijvoorbeeld ook een eerder technische woordenlijst zijn met boekhoudkundige woordenschat. Problemen ontstaan echter wanneer een woord niet in het woordenboek staat, zoals een eigenaam. Op deze manier worden veelvoorkomende fouten zoals het miscategoriseren van een O naar een 0 verholpen. Vandaag kan zelfs Natural Language Processing (NLP) gebruikt worden. NLP kan paragrafen of documenten categoriseren op basis van de gebruikte woordenschat. Recent onderzoek van El Hadi e.a. (2018) wijst uit dat OCR modellen verbeterd kunnen worden door het toepassen van NLP in zowel voor- als naverwerking.

Een meer recente toevoeging is het gebruik van neurale netwerken met miljoenen voorbeelden van karakters. Matei e.a. (2013) gebruikten een neuraal netwerk met backpropagation ter training en het k-dichtste buren algoritme ter toewijzing van input aan de gliefen (Shah, 2017).

¹<https://www.youtube.com/watch?v=RGsrUcWwIXI>

²<https://www.youtube.com/watch?v=bmZiLh1GL18>

³<https://www.youtube.com/watch?v=09mb78oiPkA>

2.3 Nauwkeurigheid

In de meeste gevallen wordt de nauwkeurigheid van OCR-technologie beoordeeld op tekenniveau. Hoe nauwkeurig een OCR-software is op tekenniveau hangt af van hoe vaak een teken correct tegenover fout wordt herkend. Een nauwkeurigheid van 99% betekent dat 1 op de 100 tekens verkeerd herkend werd. Terwijl een nauwkeurigheid van 99,9% betekent dat 1 op de 1000 tekens onzeker is.

Deze nauwkeurigheid wordt gemeten door de uitvoer voor een afbeelding te vergelijken met de originele versie van dezelfde tekst. Je kunt dan ofwel tellen hoeveel tekens correct werden herkend (nauwkeurigheid op tekenniveau), ofwel tellen hoeveel woorden correct werden herkend (nauwkeurigheid op woordniveau). (HarrisHi, 2020)

Deze studie maakt gebruik van de maatstaf voor herkenningsprestaties van Alexandrov (2003). Het model onderstreept vier soorten fouten in het herkenningsproces van OCR, namelijk: substitutie, schrapping, verwerping en toevoeging. Substitutie treedt op wanneer een teken als een ander teken wordt herkend. Substitutie vindt normaal plaats voor structuureel aangrenzende tekens. Schrapping daarentegen vindt plaats wanneer een teken wordt genegeerd omdat de OCR het als ruis ziet. Verwerping gebeurt wanneer het systeem een symbool niet kan onderscheiden of niet zeker is van de herkenning. Toevoeging vindt plaats wanneer de OCR één symbool als twee herkent, of wanneer ruis wordt onderscheiden als een of meerdere tekens. Volgens de maatstaven voor herkenningsprestaties van Alexandrov (2003) kunnen verschillende berekeningen worden gemaakt. De metingen omvatten het globale foutenpercentage, het weigeringspercentage, het herkenningspercentage en de mate van betrouwbaarheid van de OCR-systemen.

Het is opmerkelijk dat Alexandrov (2003) stelt dat het reduceren van het foutenpercentage gepaard gaat met een hoger aantal verwerpingen. Het systeem geeft aan welke karakters of ruis een te lage zekerheid hebben om betrouwbaar herkend te worden. Het voordeel is dat men deze verwerpingen kan opvangen en manueel kan corrigeren. Dit vereist echter kostbare werkuren.

Voor het berekenen van de maatstaven worden volgende parameters gebruikt.

Parameters	
Naam	Verklaring
ne	aantal substitutie, schrapping en toevoegingsfouten
nr	aantal verwerpingsfouten
nc	aantal karakters in de tekst

2.3.1 Globale foutenpercentage

De belangrijkste maatstaf van deze studie is het globale hoofdpercentage of Gerr. Gerr wordt bepaald op basis van het aantal gemaakte fouten en het aantal tekens in de tekst.

$$Gerr = 100(ne/nc) \quad (2.1)$$

2.3.2 Weigeringspercentage

Het weigeringspercentage of Grej, is de maat voor de geregistreerde verwerpingen. Het kan op basis van de volgende formule worden berekend.

$$Grej = 100(nr/nc) \quad (2.2)$$

2.3.3 Herkenningspercentage

Herkenningspercentage wordt vaak gebruikt om de efficiëntie van een OCR-systeem te beschrijven. De vergelijking is als volgt:

$$Grec = 100(nc - nr - ne)/nc \quad (2.3)$$

2.3.4 Betrouwbaarheidsniveau

Het betrouwbaarheidsniveau combineert bovenstaande maten. De vergelijking is als volgt:

$$Grel = 100(nc - nr - ne)/(nc - nr) = Grec/(Grec - Gerr) \quad (2.4)$$

Het doel is Grec te maximaliseren en Gerr te minimaliseren. In het algemeen is het totale percentage van de herkenning gebaseerd op Grec + Gerr.

2.4 Eigenschappen en hun invloed

In dit onderdeel worden factoren beschreven en hun invloed op de uiteindelijke accuraatheid of snelheid van OCR software. Ook wordt de uitbreidbaarheidsvereiste besproken. Deze worden in een volgend onderdeel gebruikt om een shortlist op te stellen.

- Taalondersteuning
- Aanwezigheid van technieken
- Bronmateriaal

2.4.1 Taalondersteuning

Holley (2009) wijst het gebruik van een woordenboek aan als een van de makkelijkste manieren om de nauwkeurigheid van het resultaat van een verwerkt document aan te duiden.

Om de nauwkeurigheid op woordniveau te verbeteren, maken de meeste OCR engines gebruik van extra kennis over de taal die in een tekst wordt gebruikt. Als de taal van de tekst bekend is (bv. Engels), kunnen de herkende woorden vergeleken worden met een woordenboek van alle bestaande woorden (bv. alle woorden van het Engelse woordenboek). Onzekere tekens kunnen dan worden gecorrigeerd door het corresponderende woord te vinden met de grootste overeenkomst.

2.4.2 Aanwezigheid van technieken

In onderdeel 2.2 werden enkele technieken opgesomd. Deze zijn deel van een OCR-software precies omdat ze de nauwkeurigheid verbeteren. Deze verbetering zal een directe invloed hebben op een longlist aangezien een hoge nauwkeurigheid gewenst is.

2.4.3 Bronmateriaal

Als de kwaliteit van het originele bestand goed is, d.w.z. als menselijke ogen het originele materiaal duidelijk kunnen zien, zal het mogelijk zijn goede OCR-resultaten te behalen. Maar als de originele bron zelf niet duidelijk is, zullen de resultaten hoogstwaarschijnlijk fouten bevatten. Hoe hoger de kwaliteit van de originele afbeelding, hoe eenvoudiger het is om tekens te onderscheiden van ruis of andere tekens, hoe hoger de nauwkeurigheid van zal zijn.

Volgens Holley (2009) is een dots per inch (dpi) van 300 aangeraden. Dit wordt verder door „Why is OCR at 300 dpi a standard?” (2017) ondersteund. Het pakket kan met meer dots per inch een hogere mate van nauwkeurigheid bereiken, omdat het meer gegevens heeft om de juiste beslissing over het teken te nemen. Het voorbeeld hieronder staaft dit argument.

Figuur 2.3: Belang van dpi



Het linkse teken is met een dpi van 300 of meer ingescand terwijl het rechtse teken minder informatie bevat. Voor het menselijke oog is het duidelijk dat het in beide gevallen om de letter "B"gaat, maar software kan hier de mist in gaan en het label "8"toekennen.

2.5 Beschikbare OCR technologieën

Dit onderdeel somt de beschikbare software op in een longlist. Deze lijst wordt verder gefilterd aan de hand van relevante requirements.

2.5.1 Requirements

Een requirement is een vereiste waar de elementen van de longlist eventueel aan voldoen. De requirements worden onderscheiden aan de hand van de MoSCoW-methode (McIntyre, 2020). Deze stelt dat requirements onderverdeeld kunnen worden aan de hand van de categorieën "must-have", "should-have" en "could-have".

Ondersteuning Nederlands

Deze requirement komt overeen met wat gezegd werd in 2.4.1. Dit wordt in se als requirement opgenomen aangezien het de nauwkeurigheid verbetert, maar tevens een van de grootste onderscheidende factoren is tussen de verschillende opties. Lang niet alle pakketten beschikken over deze eigenschap. Het is een "must-have".

Open source

Pakketten met een open source licentie zijn vaak niet verbonden aan financiële verplichtingen en kunnen door programmeurs vrij gebruikt worden voor persoonlijke en commerciële doeleinden. Daarom is het een "could-have" requirement.

Hoge nauwkeurigheid

Er is een hoge nauwkeurigheid vereist omdat fouten een grote impact kunnen hebben. Dit is een "must-have".

Uitbreidbaarheidsvereiste

De uitbreidbaarheidsvereiste is een bijkomende parameter die een grote impact op de casus heeft. Deze vereiste stelt dat de pakketten verwerkbaar zijn in een C# applicatie op een snelle, eenvoudige manier. De software bestaat dus zeker niet uit een grafische interface. Dit kan bijvoorbeeld verzorgd worden door het bestaan van een package. Deze vereiste is relevant omdat de afwezigheid van een dergelijk minimum ervoor kan zorgen dat de implementatie van een oplossing voor de casus exponentieel langer kan duren. Deze langere tijdsduur brengt een extra kost teweeg voor de programmeur en/of boekhouder. Dit is een "must-have".

Up-to-date

Het is wenselijk dat software up-to-date is. Software die niet actief onderhouden is, heeft dikwijls de laatste nieuwe verbeteringen van het domein niet geïmplementeerd. Het kan ook voor verdere problemen zorgen i.v.m. comptabiliteit met nieuwe systemen. De knoop wordt doorgehakt op 9 maanden sinds de laatste uitgebrachte versie. Het is een "must-have" requirement.

2.5.2 Longlist

Onderstaande lijst bevat een longlist van allerhande OCR-software. Deze lijst is nog niet gefilterd op relevante vereisten. Het is het resultaat van opzoekingswerk in bestaande studies zoals: Che Abdul Rahman e.a. (2019), Curtis (2010) en Smith (2007)

Lijst van OCR-technologieën				
Naam	Ondersteuning voor Nederlands	Open source	Voldoet aan uitbreidbaarheidsvereiste	Up-to-date
Tesseract	Ja	Ja	Ja	Ja
CuneiForm	Ja	Ja	Ja	Nee
ABBYY FineReader	Ja	Nee	Ja	Ja
Asprise OCR	Ja	Nee	Ja	Ja
OmniPage	Ja	Nee	Ja	Ja
Dynamsoft OCR	Ja	Nee	Ja	Ja
Google Cloud Vision	Ja	Nee	Ja	Ja
AnyDoc Software	Onbekend	Nee	Onbekend	Nee
Ocrad	Nee	Ja	Nee	Nee
OCRopus	Nee	Ja	Onbekend	Ja
OCRFeeder	Nee	Ja	Nee	Ja

Tabel 2.1: Overzicht OCR software

In de onderstaande tabel wordt de longlist gefilterd. Deze lijst is uitgedund door het stellen van twee minimumvereisten. Dit zijn de uitbreidbaarheids- en up-to-datevereiste. Volgende softwares hebben dus allen minstens de mogelijkheid om verwerkbaar te zijn in een C# applicatie en zijn niet louter een GUI.

Lijst van OCR-technologieën				
Naam	Ondersteuning voor Nederlands	Open source	Voldoet aan uitbreidbaarheidsvereiste	
Tesseract	Ja	Ja	Ja	
ABBYY FineReader	Ja	Nee	Ja	
Asprise OCR	Ja	Nee	Ja	
OmniPage	Ja	Nee	Ja	
Dynamsoft OCR	Ja	Nee	Ja	

Tabel 2.2: Overzicht OCR software

Om de onderzoeksvraag i.v.m. de vergelijking tussen open source en betalende variante op te lossen, is een kandidaat uit beide groepen nodig.

Volgens HarrisHi (2020) werd Tesseract beschouwd als het beste open source OCR-pakket. De nauwkeurigheid ervan is out-of-the-box hoog en kan aanzienlijk worden verhoogd met een goede ontworpen Tesseract voorverwerkingspipeline. Als het aankomt op betaalde OCR-pakketten, lijkt het erop dat ABBYY FineReader de pole position grijpt.

Op basis van het voorgaande ziet de shortlist er als volgt uit:

- Tesseract
- ABBYY FineReader

2.5.3 Marktaanbod

ABBYY

ABBYY FineReader is naar verluidt de krachtigste OCR-software die op de markt beschikbaar is voor snelle en nauwkeurige tekstherkenning (Dinita e.a., 2021). Volgens Heliński e.a. (2012) blijkt FineReader op het niveau van karakters marginaal nauwkeuriger te zijn in vergelijking met Tesseract. FineReader is ook in staat om grote hoeveelheden gegevens te verwerken en arbeidsintensieve taken te corrigeren. Deze OCR-software ondersteunt 192 talen. (Che Abdul Rahman e.a., 2019)

2.5.4 Open source route

Zoals Von Hippel (2001) aantoon, kent open source een groot aantal voordelen. Daarom is dit onderzoek niet compleet zonder te analyseren hoe het open source landschap er voor OCR uitziet. Hij stelt dat elke entiteit, of het nu een particulier of een bedrijf is, kan precies creëren wat zij wil. Zij hoeft niet te vertrouwen op een fabrikant. Individuele gebruikers in een gebruikersinnovatiegemeenschap hoeven niet alles wat zij nodig hebben zelf te ontwikkelen, maar kunnen profiteren van de vrijelijk gedeelde innovaties van anderen. De beschikbare broncode zorgt dus voor een grotere mogelijkheid om maatwerk te verrichten.

Tesseract

De wellicht meest prominente kandidaat Tesseract heeft een Apache 2.0 licentie. (Smith, 2007) Een implementatie van deze software in een nieuw pakket staat dus los van financiële verplichtingen.

Het werd oorspronkelijk door Hewlett-Packard in de jaren 1980 ontwikkeld. De broncode is grotendeels in C geschreven. Na 1995 werd er vanuit Hewlett-Packard geen ontwikkeling meer gedaan. Tesseract werd vrijgegeven als open source in 2005 en sinds 2006 wordt de ontwikkeling door Google gesponsord. Na meer dan 10 jaar inactief te zijn geweest, lag Tesseract achter op de toonaangevende commerciële pakketten wat zijn nauwkeurigheid betreft. (Smith, 2007) Verder bestaan GOCR⁴ en CuneiForm⁵ als gerenommeerde

⁴<http://jocr.sourceforge.net>

⁵<https://launchpad.net/cuneiform-linux>

technologieën.

Tesseract is met zijn meer dan 5000 commits, die wekelijks toenemen, zeer goed onderhouden⁶. Dit is anders voor haar open source rivalen. GOCR is al sinds 2011 niet meer bijgewerkt.

Ondersteuning voor de taal waarin men te werk gaat, is van essentieel belang. Het verschil in accuraatheid is niet te verwaarlozen. Dit werd reeds besproken in onderdeel 2.4.1. Tesseract is hier goed uitgerust met ondersteuning voor meer dan 110 talen. Overige noemenswaardige open source pakketten bieden geen ondersteuning voor het Nederlands. Met uitzondering van GOCR.

Gebruiksvriendelijkheid is niet altijd een gegeven. Daarom is het belangrijk te onderzoeken welke stappen men moet ondernemen om OCR toe te passen in een nieuw project. Tesseract kent momenteel de meeste opties. Er bestaan onder andere NuGet packages die het incorporeren van deze OCR software in een C# project vergemakkelijken⁷. Er zijn ook opties voor andere programmeertalen zoals Python⁸.

2.6 Experimenten

In dit onderdeel worden enkele experimenten met Tesseract en ABBYY uitgevoerd. Het doel van deze testen is om de accuraatheid van de concurrerende software te onderzoeken. Deze kan echter ook afhangen van enkele parameters zoals:

- Dpi input
- Ondersteuning taal
- Nauwkeurigheid onder beste omstandigheden

De experimenten met Tesseract werden in de volgende omgeving uitgevoerd. Deze configuratie wordt tevens aangeraden om replicatie te verkrijgen.

- Visual Studio 2019
- .NET Framework 4.5
- Windows 10
- Tesseract Nuget package⁹

De NuGet package voor Tesseract werd gebruikt. Er is van de Tesseract NuGet package gebruikgemaakt. De eenvoudigste manier om deze in een Windows omgeving op te zetten is om deze repository te clonen en de solution te openen in Visual Studio.

Vooraleer men goeie resultaten kan verwachten is het nodig om een getrainde dataset

⁶<https://github.com/tesseract-ocr/tesseract/commits/master>

⁷ <https://www.nuget.org/packages/tesseract/>

⁸<https://pypi.org/project/pytesseract/>

⁹<https://www.nuget.org/packages/tesseract/>

voor het Nederlands te downloaden¹⁰. Na het zip-bestand uitgepakt te hebben, moet het nld.traineddata bestand naar de map tessdata verplaatst worden. Hierna past men in de constructor van TesseractEngine in de OnSubmitFileClicked methode in het pad naar de eerder vermelde map aan zodat het overeenkomt met de bestaande bestandsstructuur.

Vervolgens is het mogelijk om het WebDemo project te starten. Men kan op dat moment navigeren naar een bestand in JPEG of TIFF formaat en deze selecteren. Na op de submit geklikt te hebben krijgt men het resultaat te zien.

```
using System;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace Tesseract.WebDemo
{
    public class DefaultPage : System.Web.UI.Page
    {
        #region Data

        // input panel controls

        protected Panel inputPanel;
        protected HtmlInputFile inputFile;
        protected HtmlButton submitFile;

        // result panel controls
        protected Panel resultPanel;
        protected HtmlGenericControl meanConfidenceLabel;
        protected HtmlTextArea resultText;
        protected HtmlButton restartButton;

        #endregion

        #region Event Handlers

        private void OnSubmitFileClicked(object sender, EventArgs args)
        {
            if (inputFile.PostedFile != null &&
                inputFile.PostedFile.ContentLength > 0)
            {

                using (var engine = new
                    TesseractEngine("E:\\Documenten\\tesseracttest\\src\\Tesseract.Consol
                    "eng", EngineMode.Default))
                {
                    using (var image = new
```

¹⁰https://github.com/tesseract-ocr/tessdata_fast/releases/tag/4.1.0

```
System.Drawing.Bitmap(imageFile.PostedFile.InputStream))
{
    using (var pix = PixConverter.ToPix(image))
    {
        using (var page = engine.Process(pix))
        {
            meanConfidenceLabel.InnerText =
                String.Format("{0:P}",
                page.GetMeanConfidence());
            resultText.InnerText = page.GetText();
        }
    }
}
inputPanel.Visible = false;
resultPanel.Visible = true;
}

private void OnRestartClicked(object sender, EventArgs args)
{
    resultPanel.Visible = false;
    inputPanel.Visible = true;
}

#endregion

#region Page Setup
protected override void OnInit(EventArgs e)
{
    InitializeComponent();
    base.OnInit(e);
}

private void InitializeComponent()
{
    this.restartButton.ServerClick += OnRestartClicked;
    this.submitFile.ServerClick += OnSubmitFileClicked;
}

#endregion
}
```

Figuur 2.4: Voorbeeld functioneel project

The screenshot shows a simple web form titled "File Upload". It has a single input field labeled "File:" with a placeholder "Choose File" and a status message "No file chosen". Below the input field is a note "The file to be processed." and a "Submit" button at the bottom.

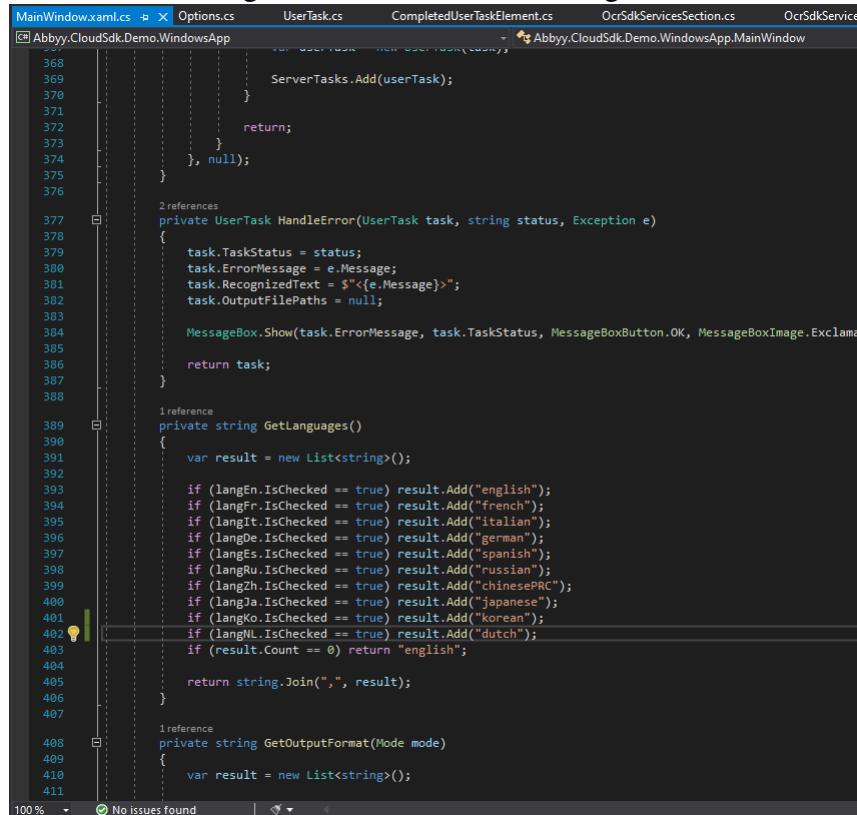
De experimenten met ABBYY werden in de volgende omgeving uitgevoerd. Deze configuratie wordt tevens aangeraden om replicatie te verkrijgen.

- Visual Studio 2019
- .NET Framework 4.5
- Windows 10
- Demo-applicatie¹¹

Er zijn enkele wijzigingen aan de demo-applicatie nodig om ondersteuning voor Nederlandstalige documenten te voorzien.

¹¹<https://github.com/abbyy/cloudsdk-demo-dotnet>

Figuur 2.5: Nederlands toevoegen



```

368     }
369     {
370         ServerTasks.Add(userTask);
371     }
372     return;
373 }, null);
374 }
375
376 }

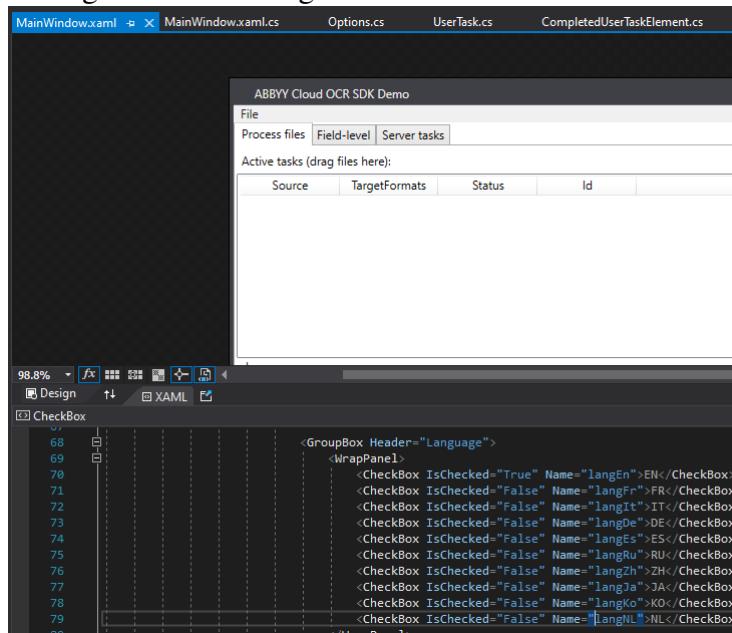
377 2 references
378 private UserTask HandleError(UserTask task, string status, Exception e)
379 {
380     task.TaskStatus = status;
381     task.ErrorMessage = $"{e.Message}";
382     task.RecognizedText = $"<{e.Message}>";
383     task.OutputFilePaths = null;
384
385     MessageBox.Show(task.ErrorMessage, task.TaskStatus, MessageBoxButton.OK, MessageBoxImage.Exclamation);
386
387     return task;
388 }

389 1 reference
390 private string GetLanguages()
391 {
392     var result = new List<string>();
393
394     if (langEn.IsChecked == true) result.Add("english");
395     if (langFr.IsChecked == true) result.Add("french");
396     if (langIt.IsChecked == true) result.Add("italian");
397     if (langDe.IsChecked == true) result.Add("german");
398     if (langEs.IsChecked == true) result.Add("spanish");
399     if (langRu.IsChecked == true) result.Add("russian");
400     if (langZh.IsChecked == true) result.Add("chinesePRC");
401     if (langKo.IsChecked == true) result.Add("korean");
402     if (langNL.IsChecked == true) result.Add("dutch");
403
404     if (result.Count == 0) return "english";
405
406     return string.Join(", ", result);
407 }

408 1 reference
409 private string GetOutputFormat(Mode mode)
410 {
411     var result = new List<string>();

```

Figuur 2.6: Toevoegen checkbox voor Nederlands



Ten eerste moet men op lijn 402 in MainWindow.xaml.cs overnemen wat op 2.5 staat.

Vervolgens moet een extra knop toegevoegd worden zodat men de taaloptie kan selecteren. In MainWindow.xaml kan men op lijn 79 de waarde die op 2.6 te zien is, overnemen. Wanneer de applicatie opgestart wordt, kan men 'NL' aanklikken om aan te duiden dat documenten in de wachtrij als Nederlandstalige behandeld moeten worden.

2.6.1 Methodologie van de experimenten

In de volgende onderdelen worden enkele experimenten uitgevoerd om de nauwkeurigheid van OCR-pakketten te valideren. Deze studie maakt gebruik van een kwantitatief onderzoek om deze uit te voeren. In deze paper worden de beschikbare producten getest. Het is vanzelfsprekend voor de casus dat facturen als invoer gebruikt worden. Facturen zijn inherent anders dan gewone documenten. Ze bevatten dikwijls tabellen en figuren zoals logo's die het proces bemoeilijken. Daarom is het relevant om na te gaan of de in de literatuur vermelde nauwkeurigheid overdraagbaar is naar facturen.

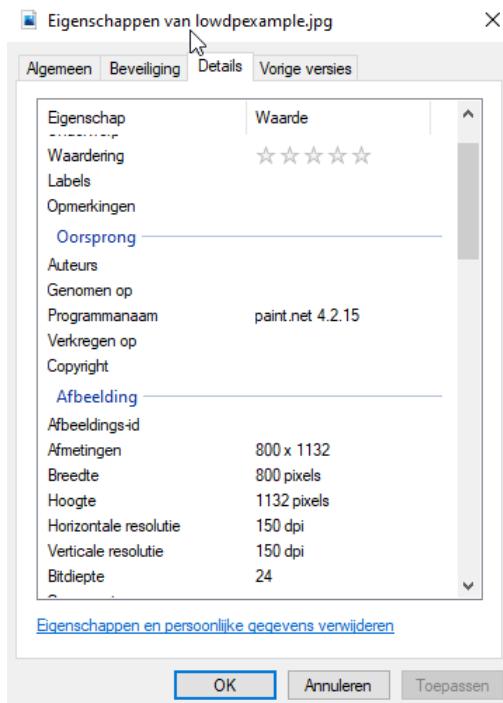
Het vergaren van de data verloopt als volgt: een demoproject wordt opgestart, de gebruiker kiest het gewenste bestand waar de test op uitgevoerd wordt en de output wordt in een tekstbestand opgeslagen. Op deze data worden de formules in 2.3 uitgevoerd. Deze maatstaven worden opgeslagen voor verdere verwerking.

De test wordt 30 keer herhaald zodat de dataset voldoende groot is. Het gaat om de accuraatheid van een OCR-pakket zoals uitgedrukt in 2.3. Deze cijfers worden vervolgens vergeleken met resultaten van voorgaande studies.

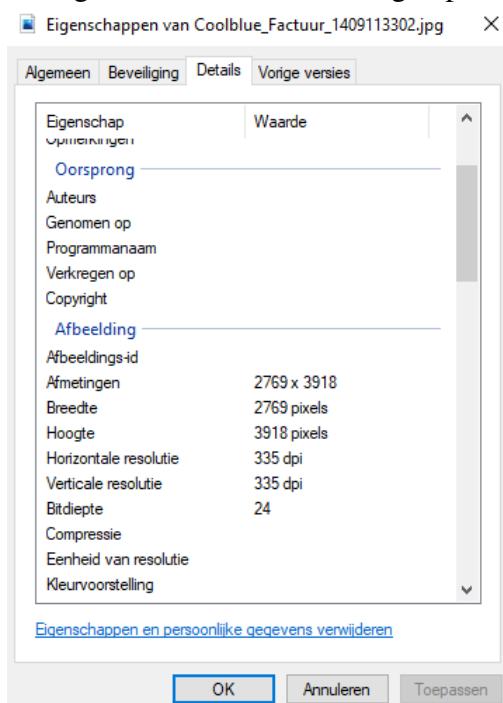
Ten slotte worden de data geanalyseerd door middel van statistische methodes in R. Er wordt een boxplot gemaakt om informatie zoals gemiddelde, minimum, maximum, mediaan en het eerste quartiel op een eenduidige manier weer te geven.

Onderstaande figuren geven de eigenschappen van de gebruikte factuur weer. Deze factuur is voldoende complex om het pakket te testen. Het heeft enkele figuren en is niet louter tekst. Dit is wenselijk omdat facturen vaak enkele figuren of andere afbeeldingen bevatten. Het is belangrijk dat de software deze niet als tekst interpreteert. Er zijn 2 versies van dezelfde factuur. De Dots per inch (dpi) is bij de alternatieve versie veranderd om de impact ervan te evalueren.

Figuur 2.7: Factuur met lage dpi



Figuur 2.8: Factuur met hoge dpi



Figuur 2.9: Input voor experimenten in onderdeel 2.6

FACTUUR.
Dat betaal ik zelf wel.

remi mestdagh
aartrijkestraat 39
8820 torhout

Factuurnummer: 1409113302
Klantnummer: 21752541
Factuurdatum: 24 maart 2021
Ordernummer: 52517004
Orderdatum: 24 maart 2021

Coolblue N.V.
Borsbeeksebrug 28
2600 Berchem
België
www.coolblue.be

RPR Antwerpen
BTW BE0867686774
IBAN BE07310160125666
BIC BBRUBEBB

Artikel	Aantal	Prijs per stuk	BTW	Prijs incl. BTW
Tello Drone (powered by DJI) Incl. Recupel: 07.01 Speelgoed - huishoudelijk	1	€ 99,00	21%	€ 99,00 € 0,04

Exclusief BTW	€ 81,82	Subtotaal	€ 99,00
BTW 21%	€ 17,18		
Totaal	€ 99,00	Totaal	€ 99,00

"Bancontact" op 24 maart 2021



Printen? Niet nodig. Je kunt je factuur altijd terugvinden in je mail of in je Mijn Coolblue-account.

Altijd minimaal 2 jaar garantie. Doen we niet moeilijk over.

**cool
blue**

De gegenereerde grafieken die het herkenningspercentage plotten bevatten volgende infor-

matie:

- op de X-as wordt het herkenningspercentage in % weergegeven
- op de Y-as wordt onderscheid gemaakt tussen het niveau waarop deze herkenningspercentage toepassing heeft

2.6.2 Tesseract

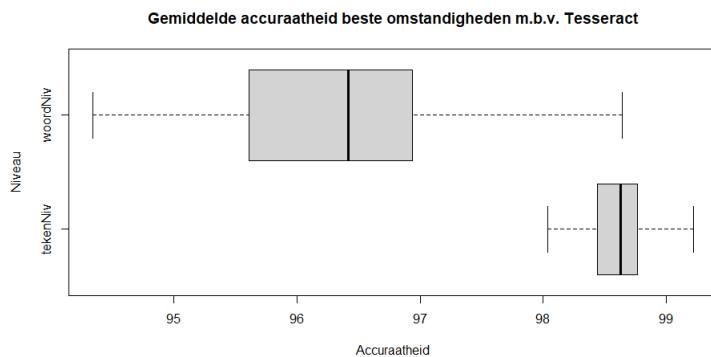
Nauwkeurigheid onder beste omstandigheden

Onder de beste omstandigheden verstaat men enkele zaken. Ten eerste moet men over een exemplaar van de factuur beschikken met een voldoende hoge dpi. In een voorgaand hoofdstuk stelt men dat dit 300 of meer moet zijn. Ook is het aan te raden dat de software een dataset voor het Nederlands heeft. Voor deze test wordt figuur 2.9 gebruikt als input.

Na het uitvoeren van de meermaals herhaalde tests blijkt dat het gemiddelde herkenningspercentage op tekenniveau 98,59% is. Op woordniveau bedraagt dit 96,30%. In B.1.2 is een voorbeeld te vinden.

Fouten bevatten bijvoorbeeld een "O" die als een "0" herkend wordt.

Figuur 2.10: Herkenningspercentage beste omstandigheden



Opvallend is dat het herkenningspercentage op woordniveau lager ligt dan deze op tekenniveau. Dit is vanzelfsprekend aangezien er minder woorden dan tekens zijn. Omdat een enkel verkeerd teken in een woord ervoor zorgt dat het woord onbruikbaar is, heeft een verkeerd teken een grotere procentuele invloed op woordniveau.

De software haalt zelfs onder optimale omstandigheden geen 100% nauwkeurigheid.

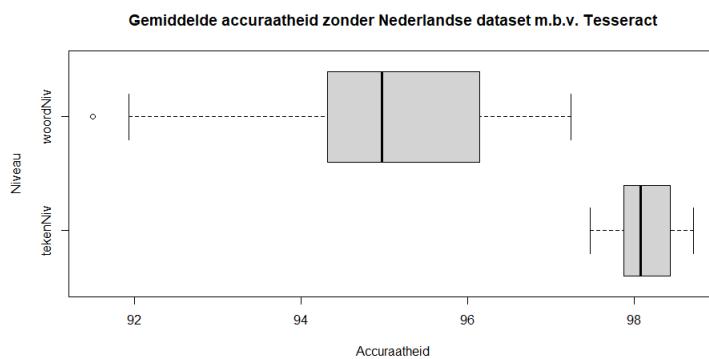
Invloed ondersteuning taal

In bijlage B.1.3 is het resultaat te vinden. De gehanteerde taal kan in het bestand Default.aspx.cs op lijn 38 aangepast worden. Voor de test is 'nld' door 'eng' vervangen. Dit

zorgt dat Tesseract een Engelse dataset gebruikt. Het document in 2.9 wordt als input gebruikt.

De behaalde gemiddelde nauwkeurigheid op tekenniveau is 98,11%. Op woordniveau is dit 94,99%. Dit is lager dan het gemiddelde van de vorige test.

Figuur 2.11: Herkenningspercentage zonder taalondersteuning



Vergelijking tekenniveau Om te onderzoeken of dit verschil in nauwkeurigheid significant is, wordt de t-test uitgevoerd. De nulhypothese stelt dat deze gemiddelen gelijk zijn. De alternatieve hypothese stelt dat het gemiddelde onder de beste omstandigheden groter is dan deze wanneer de taal niet ondersteund wordt.

α is de kans dat de geteste hypothese ten onrechte wordt verworpen. Men kiest voor een waarde van 5%.

De gebruikte R dataframes zijn online te vinden.¹²

$$H_0 : \mu_0 = \mu_1$$

$$H_1 : \mu_0 > \mu_1$$

$$\alpha = 0,05$$

```
var.test(tessBeste$tekenNiv, tessTaal$tekenNiv, alternative =
  "two.sided")

t.test(tessBeste$tekenNiv, tessTaal$tekenNiv, conf.level = .05,
  alternative = "greater")
```

Ten eerste dient men de F-test uit te voeren om te onderzoeken of de variantie niet significant afwijkt. Dit is een voorwaarde voor de t-test. De output van deze eerste functie geeft een p-waarde aan van 0,271. Dit is hoger dan α dus is er geen significant verschil.

¹²<https://drive.google.com/drive/folders/1VGRbWU1j8XTQPHFY9RHX9cmUhstofLvh?usp=sharing>

Vervolgens voert men de t-test uit. De p-waarde van deze functie is 1,251e-07. Deze is lager dan α . We kunnen dus stellen dat de nulhypothese verworpen is.

Uit dit alles besluiten we dat er een significant verschil op tekenniveau is tussen beide instellingen. De gemiddelde nauwkeurigheid is significant lager als de gewenste taal niet ondersteund wordt.

Vergelijking woordniveau Dezelfde test wordt gebruikt om significante verschillen op woordniveau te onderzoeken.

```
var.t.test(tessBeste$woordNiv, tessTaal$woordNiv, alternative =
  "two.sided")
t.test(tessBeste$woordNiv, tessTaal$woordNiv, conf.level = .05,
  alternative = "greater")
```

De p-waarde van de F-test is 0,2136. Dit wijst op een niet significant verschillende variantie. De p-waarde bij de t-test wijst echter wel op een significant verschil (8,334-e05). We kunnen besluiten dat de gemiddelde nauwkeurigheid op woordniveau significant lager is.

Besluit In overeenkomst met onderdeel 2.4.1 behaalt men een minder hoge nauwkeurigheid op woord- en tekenniveau wanneer het OCR-pakket de taal van het inputdocument niet ondersteunt. De verkeerd herkende tekens bevatten deze die kenmerkend aan het Nederlands zijn zoals de trema. Zo wordt 'België' consistent als 'Belgié' geschreven. Andere fouten zoals het substitueren van een O door een O, zijn echter niet te wijten aan de gehanteerde dataset. Verder onderzoek is vereist om aan te tonen dat de invloed van de taal eventueel een kleiner verschil teweegbrengt zolang men een taal gebruikt die relatief nauw aansluit bij het Nederlands.

Invloed van dpi

Voor dit experiment worden dezelfde facturen gebruikt, maar verandert de dpi. Dit is zichtbaar in figuur 2.7.

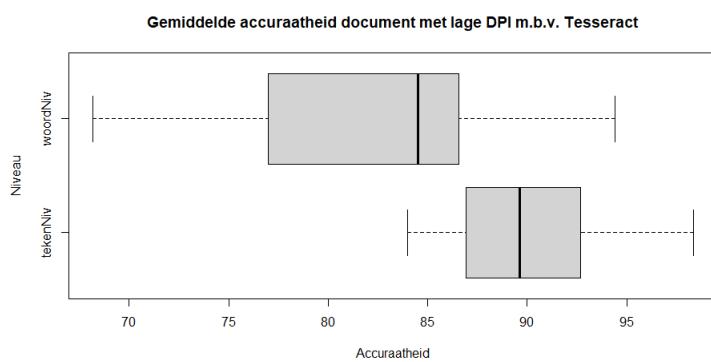
Het gemiddelde herkenningspercentage op tekenniveau is 89,80%. Op woordniveau bedraagt deze 82,26%. De vraag rijst of dit resultaat voldoende is voor gebruik in een applicatie. Dit resultaat kan een invloed hebben op de efficiëntie van het pakket. In het resultaat staan frequent op het eerste zicht willekeurige tekens. Dit bemoeilijkt verdere verwerking aangezien de waarde 'pi Borsbeeksebrug 25' tevoorschijn kwam, terwijl het eigenlijk 'Borsbeeksebrug 25' moet zijn. Dit kan grote fouten teweegbrengen wanneer er bijvoorbeeld een willekeurig getal aan de sequentie die de prijs voorstelt toegevoegd zou worden.

Onderstaand voorbeeld heeft een dpi van 150. In bijlage B.1.1 staan de verkregen output van de eerste test en de algemene instellingen vermeld.

Wanneer men de maatstaven van Alexandrov (2003) gebruikt, krijgt men meer inzicht in de kenmerken van het systeem. Ten eerste meet men een globaal foutenpercentage van 10,20%. Opmerkelijk is dat het weigeringspercentage 0% is in de huidige configuratie. Tesseract maakt geen verwerpingen. Het poogt steeds om een karakter te herkennen. Dit leidt echter tot grotere foutenpercentages. Het herkenningspercentage komt overeen met de accuraatheid op tekenniveau. Het finale betrouwbaarheidsniveau bedraagt tevens 89,80%.

Enkele fouten kunnen geweten worden aan het falen van 'line-removal'. Deze techniek zorgt ervoor, zoals eerder vermeld, dat lijnen verwijderd worden. Dit wordt voorzien omdat ze als tekens kunnen herkend worden, zoals hier gebeurd is.

Figuur 2.12: Boxplot herkenningspercentage bij lage dpi



Vergelijking tekenniveau Om te onderzoeken of dit verschil in nauwkeurigheid significant is, wordt de t-test uitgevoerd. De nulhypothese stelt dat deze gemiddelen gelijk zijn. De alternatieve hypothese stelt dat het gemiddelde onder de beste omstandigheden groter is dan wanneer de facturen een lagere dpi hebben.

$$H_0 : \mu_0 = \mu_1$$

$$H_1 : \mu_0 > \mu_1$$

$$\alpha = 0,05$$

```

var.test(tessBeste$tekenNiv, tessTaal$tekenNiv, alternative =
  "two.sided")

t.test(tessBeste$tekenNiv, tessTaal$tekenNiv, conf.level = .05,
  alternative = "greater", var.equal = FALSE)

```

Ten eerste dient men de F-test uit te voeren om te onderzoeken of de variantie niet significant afwijkt. Dit is een voorwaarde voor de t-test. De output van deze eerste functie geeft een p-waarde aan die lager is dan 2,2e-16. Dit is lager dan α dus is er een significant verschil.

Vervolgens voert men de t-test uit. Deze moet ingesteld worden op het bestaan van ongelijke variantie. Dit doet men door "var.equal = FALSE" als parameter toe te voegen. De p-waarde van deze functie is 2.223e-14. Deze is lager dan α . We kunnen dus stellen dat de nulhypothese verworpen is.

Uit dit alles besluiten we dat er een significant verschil op tekenniveau is tussen beide instellingen. De gemiddelde nauwkeurigheid is significant lager.

Vergelijking woordniveau Dezelfde test wordt gebruikt om significante verschillen op woordniveau te onderzoeken.

```
var.test(tessBeste$woordNiv, tessDPI$woordNiv, alternative =
  "two.sided")
t.test(tessBeste$woordNiv, tessDPI$woordNiv, conf.level = .05,
  alternative = "greater", var.equal = FALSE)
```

De p-waarde van de F-test is 6,098e-16. Dit wijst op een significant verschillende variantie. Dit wil zeggen dat dit opnieuw moet aangeven door de parameter "var.equal = FALSE" toe te voegen. Ook bij de t-test wijst de p-waarde op een significant verschil (8,932-e13). We kunnen besluiten dat de gemiddelde nauwkeurigheid op woordniveau significant lager is.

Besluit In overeenstemming met onderdeel 2.4.3 behaalt men een minder hoge nauwkeurigheid op woord- en tekenniveau wanneer het inputdocument een lagere dpi-waarde heeft.

2.6.3 ABBYY

In dit onderdeel wordt ABBYY FineReader gebruikt om de documenten te verwerken.

Nauwkeurigheid onder beste omstandigheden

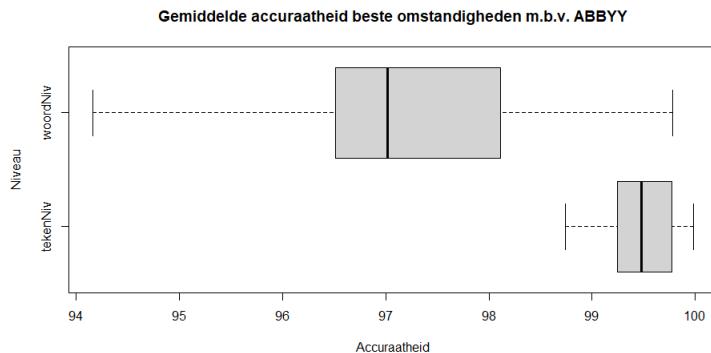
In B.2.1 is het eerste resultaat van dit pakket te zien. Na de test 30 maal te herhalen is het gemiddelde herkenningspercentage 99,47% op tekenniveau en 97,01% op woordniveau. Dit is op het eerste zicht een uiterst hoge score.

Vergelijking tekenniveau Om te onderzoeken of dit verschil in nauwkeurigheid significant is, wordt de t-test uitgevoerd. De nulhypothese stelt dat deze gemiddelden gelijk zijn. De alternatieve hypothese stelt dat de gemiddelde nauwkeurigheid bij Abbyy hoger ligt dan Tesseract.

$$H_0 : \mu_0 = \mu_1$$

$$H_1 : \mu_0 < \mu_1$$

Figuur 2.13: Boxplot herkenningspercentage beste omstandigheden (ABBYY)



$$\alpha = 0,05$$

```
var.test(tessBeste$tekenNiv, abbyyBeste$tekenNiv, alternative =
  "two.sided")

t.test(tessBeste$tekenNiv, abbyyBeste$tekenNiv, conf.level = .05,
  alternative = "less")
```

Ten eerste dient men de F-test uit te voeren om te onderzoeken of de variantie niet significant afwijkt. Dit is een voorwaarde voor de t-test. De output van deze eerste functie geeft een p-waarde aan van 0,267. Dit is hoger dan α dus is er geen significant verschil.

Vervolgens voert men de t-test uit. De p-waarde van deze functie is 5,416e-16. Deze is lager dan α . We kunnen dus stellen dat de nulhypothese verworpen is.

Uit dit alles besluiten we dat er een significant verschil op tekenniveau is tussen beide pakketten. De gemiddelde nauwkeurigheid van Abbyy ligt significant hoger.

Vergelijking woordniveau Dezelfde test wordt gebruikt om significante verschillen op woordniveau te onderzoeken.

```
var.test(tessBeste$woordNiv, abbyyBeste$woordNiv, alternative =
  "two.sided")

t.test(tessBeste$woordNiv, abbyyBeste$woordNiv, conf.level = .05,
  alternative = "greater")
```

De p-waarde van de F-test is 0,079. Dit wijst op niet significant verschillende variantie. De p-waarde van de t-test bedraagt 0,021. Deze is lager dan α . Er is opnieuw een significant verschil. De nulhypothese is verworpen. Dit wil zeggen dat de gemiddelde nauwkeurigheid ook op woordniveau significant beter is bij Abbyy.

2.7 Vergelijking betalende en open source software

Abbyy en Tesseract liggen op de uitersten van het softwarelandschap op vlak van openbaarheid van broncode. Zoals in tabel 2.2 te zien, zijn de overige kenmerken in beide pakketten zeer vergelijkbaar. Ze kennen beide ondersteuning voor het Nederlands. Verder zijn ze allebei relatief eenvoudig te implementeren in een C#-applicatie door de beschikbaarheid van NuGet packages.

Op vlak van herkenningspercentage scoren ze beide hoog. Abbyy behaalt echter een significant beter resultaat op zowel teken- als woordniveau. Op woordniveau haalde de t-test een p-waarde van 0,021. Het is voor volgend onderzoek interessant om te onderzoeken of een lagere α , of grotere n voor andere resultaten zorgt.

2.8 Betekenis voor de casus

De verkregen output zoals te zien in B.1.3 bevat nog geen metadata. Het is nog niet duidelijk welke soort data het betreft. Voor een boekhouder of particulier is het in een groot aantal gevallen wel duidelijk dat de sequentie van woorden "Totaal €99,00" de totale prijs van een factuur betreft, maar dit is voor een computerprogramma niet zomaar een zekerheid.

Het is mogelijk om op basis van zoekmethodes die in programmeertalen ingebouwd zijn de tekst die na expliciete labels zoals "totale prijs" te extraheren, maar dit is veel minder vanzelfsprekend voor de waarde van bijvoorbeeld de naam van de verkoper. De labels van dit soort informatie zijn op een groot aantal facturen niet expliciet aanwezig. Onderstaand codevoorbeeld illustreert deze uitdaging. Het maakt gebruik van een reguliere expressie om prijzen te vinden in een bepaalde input.

Listing 2.1: Code voorbeeld zoeken naar prijs

```
Regex rx = new Regex(@"\s? [0-9] [0-9]*,[0-9]");  
  
string input = "Tello Drone (powered by DJI) 1 €99,00 21% €99,00 Incl.  
Recupel: 07.01 Speelgoed - huishoudelijk 1 €0,04 Exclusief BTW  
C81,82 Subtotaal €99,00BTW 21 % €17,18Totaal €99,00";  
MatchCollection matches = rx.Matches(input);  
  
// Report the number of matches found.  
Console.WriteLine("{0} matches found in:\n {1}",  
matches.Count,  
input);  
  
// Report on each match.  
foreach (Match match in matches)  
{  
    GroupCollection groups = match.Groups;
```

```
        Console.WriteLine("'{0}', repeated at positions {1} and {2}",
    groups[0].Value,
    groups[0].Index,
    groups[1].Index);
}
```

Listing 2.2: Output 2.1

```
6 matches found in:
Tello Drone (powered by DJI) 1 €99,00 21% €99,00 Incl. Recupel: 07.01
    Speelgoed - huishoudelijk 1 €0,04 Exclusief BTW C81,82 Subtotaal €
        99,00BTW 21 % €17,18Totaal €99,00
'€ 99,0' repeated at positions 31 and 0
'€ 99,0' repeated at positions 43 and 0
'€ 0,0' repeated at positions 100 and 0
'€ 99,0' repeated at positions 138 and 0
'€ 17,1' repeated at positions 154 and 0
'€ 99,0' repeated at positions 168 and 0
```

Aan de output van code voorbeeld 2.1 te zien, bestaan er meerdere opties die voldoen aan de formatering van een prijs. Dit spreekt voor zich want op facturen komen deze vaak voor. Deze manier van werken is niet geschikt om op zoek te gaan naar de finale totale prijs van een factuur. Het is onbegonnen werk om alle mogelijke sequenties die een totale prijs eenduidig uitdrukken te ondersteunen. Deze kan oneindig veel vormen aannemen. Voorbeelden hiervan zijn "Totale prijs €99,00", "Totaal €99,00", "Te betalen €99,00".... Hier stopt het echter niet. Wanneer men rekent op sequenties zoals "Totale prijs" als ankerpunt om de uiteindelijke prijs te vinden, kan men op een muur botsen. Wanneer de factuur op een dusdanige manier opgesteld is dat het ankerpunt niet op dezelfde lijn te vinden is als de eigenlijke prijs, verliest men de mogelijkheid een reguliere expressie te gebruiken om het probleem op te lossen.

2.9 Conclusie

Zowel Abbyy als Tesseract scoren hoog op de relevante maatstaven. Beide softwarepakketten beschikken over ondersteuning voor het Nederlands. Tevens op vlak van toegankelijkheid maakt de open source variant een stevige indruk. Deze gelijkaardige prestaties maken de keuze niet moeilijker. Het lijkt vanzelfsprekend om voor Tesseract te kiezen wetende dat het over een Apache-2.0 licentie beschikt.

Met de verworven kennis over OCR is het duidelijk dat er aanvullende technologieën nodig zijn om het doel te bereiken. De verkregen output van een OCR-proces volstaat niet. In het volgende onderdeel wordt Machine learning (ML) geëvalueerd als mogelijk aanvulling.

3. Machine Learning onder de loep

Traditionele OCR-software kan de inhoud van artikels lezen. Associaties en koppelingen tussen sleutel- en waardeparen zijn niet mogelijk. Formulieren, zoals facturen, bevatten deze. De relaties kunnen op een natuurlijke manier worden overgebracht door bijvoorbeeld uitlijning, inspringing, marge en regellijnen. Sleutelteksten kunnen worden onderscheiden van de bijbehorende inhoud door visuele stijlen zoals lettertype, kleur of arcering. Aangezien de beeld- en tekstinformatie met elkaar verweven zijn, moet bij de ontwikkeling van een formulerverwerkingsysteem vaak gebruik worden gemaakt van zowel computervisie als natuurlijke taalverwerkingstechnieken. (XU e.a., 2021)

In dit onderdeel wordt ML aangekaart als potentiële opvulling van deze leegte.

3.1 Wat is Machine Learning?

Mitchell (1997) definieert ML als volgt: Machine learning (ML) is de studie van computeralgoritmen die automatisch beter worden door ervaring en door het gebruik van gegevens. Het wordt gezien als een onderdeel van kunstmatige intelligentie. ML-algoritmen bouwen een model op basis van steekproefgegevens, bekend als "trainingsdata", om voorspellingen te doen of beslissingen te nemen zonder daarvoor expliciet geprogrammeerd te zijn. (Gero, 1996)

ML is verder op te delen naargelang de manier waarop trainingsdata gebruikt worden. Er zijn in het algemeen twee soorten gegevens, die radicaal verschillend behandeld moeten worden. Voor het eerste type is er een speciaal toegewezen attribuut beschikbaar en het doel van het ML-model is om dit attribuut te voorspellen voor onbekende gevallen. De aanwezigheid van zo'n attribuut of label zorgt dat ze gelabelde gegevens genoemd worden.

Machine learning met behulp van dit soort gegevens wordt ook gesuperviseerd leren genoemd. De afwezigheid van dit label wijst op een ongelabelde dataset. De techniek die dit soort data gebruikt, wordt ongesuperviseerd leren genoemd.

Indien het toegewezen attribuut categorisch van aard is, d.w.z. dat het een gelimiteerd aantal verschillende waarden kan aannemen, zoals 'zeer goed', 'goed' of 'slecht', wordt de taak classificatie genoemd. (Bramer, 2020)

3.1.1 Casus

Deze casus kent een typisch classificatieprobleem. Het indelen van data in vaste categorieën zoals 'totale prijs' of 'BTW-nummer' is hier een duidelijk voorbeeld van. Het is wenselijk om een gelabelde dataset te maken zodat we gesuperviseerd leren kunnen gebruiken. Deze heeft een hogere accuraatheid en er is geen extra input nodig om achteraf de klassen te interpreteren.

3.2 Classificatie

Classificatie is een van de meest voorkomende toepassingen. Het komt overeen met taken die vaak voorkomen in het dagelijkse leven. In wezen gaat het over het verdelen van voorwerpen in een aantal uitputtende en uitsluitende categorieën, die vaak klassen worden genoemd. Een ziekenhuis kan bijvoorbeeld patiënten classificeren in personen die een hoog, gemiddeld of laag risico lopen een bepaalde ziekte te krijgen. (Bramer, 2020) Men kan bijvoorbeeld ook een veld van een formulier indelen in velden met het label 'totale prijs' of 'naam verkoper'.

In dit onderdeel worden de relevante classificatie algoritmes besproken.

3.2.1 Naïve Bayes

Naive Bayes is een classificator die gebruikmaakt van het Bayes-theorema. Het voorspelt de waarschijnlijkheid dat een voorwerp deel is van een bepaalde klasse. De klasse met de hoogste waarschijnlijkheid wordt als de meest waarschijnlijke klasse beschouwd. Dit wordt ook wel Maximum A Posteriori genoemd. (Bramer, 2020)

- $\Pr(A|B)$ is de waarschijnlijkheid van klasse A gegeven B.
- $\Pr(A)$ is de waarschijnlijkheid dat het resultaat de klasse A is.

$$\Pr(A|B) = \frac{\Pr(B|A) \Pr(A)}{\Pr(B|A) \Pr(A) + \Pr(B|\neg A) \Pr(\neg A)} \quad (3.1)$$

Bovenstaand theorema is gebaseerd op voorwaardelijke kansen.

$$\Pr(A|B) = \frac{\Pr(B|A) \Pr(A)}{\Pr(B)} \quad (3.2)$$

3.2.2 Dichtste-buur classificatie

Het idee van het dichtste-buur algoritme is om de classificatie van een ongeziene instantie te schatten aan de hand van de klasse van de instantie of instanties die er het dichtst bij liggen. Classificatie is niet vanzelfsprekend. Er kan op geen volledig bevredigende manier met categorische attributen omgegaan worden. Een mogelijkheid is te zeggen dat het verschil tussen twee identieke waarden van het kenmerk nul is en dat het verschil tussen twee verschillende waarden 1 is. In feite komt dit erop neer dat bijvoorbeeld rood - rood = 0, rood - blauw = 1, blauw - groen = 1... Er zijn enkele manieren om de afstand te meten tussen twee instanties aan de hand van de waarde van hun attributen. De meest populaire is bij uitstek de Euclidische vergelijking. (Bramer, 2020)

Parameters	
Naam	Verklaring
x_1	Coördinaat 1 van x
x_2	Coördinaat 2 van x
y_1	Coördinaat 1 van y
y_2	Coördinaat 2 van y

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (3.3)$$

3.3 Beschikbare ML technologieën

Dit onderdeel somt de beschikbare software in een longlist op. Deze lijst wordt verder gefilterd aan de hand van relevante requirements.

3.3.1 Requirements

Een requirement wordt conform onderdeel 2.5.1 gedefinieerd. Volgende vereisten worden overgenomen en eventueel uitgediept.

- Ondersteuning Nederlands
- Open source
- Hoge nauwkeurigheid
- Uitbreidbaarheidsvereiste
- Personalisatie

Volgende vereisten zijn uniek voor dit hoofdstuk.

- Opvangen fouten

Bijkomende en uitgediepte requirements worden in dit onderdeel besproken

Hoge nauwkeurigheid

Deze parameter, uitgedrukt in percentage geeft aan in welke mate de classificatie correct gebeurt. Het hangt dus in tegenstelling tot louter OCR niet meer enkel af van de correcte interpretatie van de tekst door een OCR-engine maar ook van het Machine learning (ML) model. Het model dient tekstblokken correct in te delen in bijhorende klassen.

Opvangen fouten

Na het toepassen van de software kan in sommige gevallen manuele input nodig zijn. De manuele input slaat op deze die door een gebruiker nodig is als er verkeerde classificaties gemaakt werden. Deze hangt dus van de nauwkeurigheid af. Elke fout kan voor een grote vertraging zorgen in het proces. Niet alleen moeten de fouten gecorrigeerd worden, ook moet er een omgeving opgezet worden die deze fouten kan corrigeren.

Het is onwaarschijnlijk dat software elk document met een nauwkeurigheid van 100% analyseert. Het is daarom gewenst dat er mogelijkheden zijn om deze input te voorzien, maar ook te minimaliseren. Dit nemen we op als een "should-have" requirement.

Personalisatie

Omdat de te verwerken facturen per persoon of bedrijf anders zijn maar ook omdat ze onderling inherent uniek zijn, stijgt de nood aan personalisatie. Een Machine learning (ML) model dient getraind te worden op trainingsdata die nauw genoeg aansluiten bij de te verwerken data voor de op te lossen casus. Daarom is het gewenst dat software deze mogelijkheid aanbiedt.

3.3.2 Longlist

Volgende longlist bevat technologieën die OCR en ML gebruiken om metadata aan een ingelezen document te koppelen.

- Azure Form Recognizer¹
- Tesseract
- AWS Textract²
- DocParser³
- Google Cloud Auto ML Vision Object Detection⁴

XU e.a. (2021) beschrijft de nauwkeurigheid van enkele potentiële oplossingen. De paper bevat een goed beginpunt voor de meest interessante opties.

¹<https://azure.microsoft.com/nl-nl/services/cognitive-services/form-recognizer/>

²<https://aws.amazon.com/textract/>

³<https://docparser.com>

⁴<https://cloud.google.com/vision>

Lijst van ML-technologieën				
Naam	Ondersteuning voor Nederlands	Open source	Voldoet aan uitbreidbaarheidsvereiste	
Tesseract	Ja	Ja	Ja	
Azure Form Recognizer	Ja	Nee	Ja	
AWS Textract	Nee	Nee	Ja	

Tabel 3.1: Overzicht ML software

3.3.3 AWS Textract

Amazon Textract is een op AI gebaseerde cloudservice die automatisch tekst en gegevens uit formulieren haalt. Voor eindgebruikers is Textract eenvoudig te gebruiken. Ontwikkelaars hoeven geen aangepaste code bij te werken voor een specifiek formuliertype wanneer het formaat of de lay-out verandert. Textract is getraind op miljoenen documentafbeeldingen uit verschillende sectoren. Het kan sleutel-waardeparen detecteren en de samenstelling van in tabellen opgeslagen gegevens bewaren. Ondersteuning voor Nederlands is er momenteel niet. (XU e.a., 2021)

The screenshot shows the AWS Textract interface. On the left, there is a scanned document image of a receipt from 'BOELEN ORCHIDEE'. The receipt contains text and some tables. On the right, the extracted data is presented in a structured format:

Leveringsdatum	Aantal	Omschrijving	BTW %	EP. (Incl)	Totaal (Incl)
14/03/2021	1,00	BOEKET	6 %	€ 27,00	€ 27,00
14/03/2021	1,00	DECORATIE	21 %	€ 75,00	€ 75,00
14/03/2021	1,00	PLANTEN	6 %	€ 19,20	€ 19,20

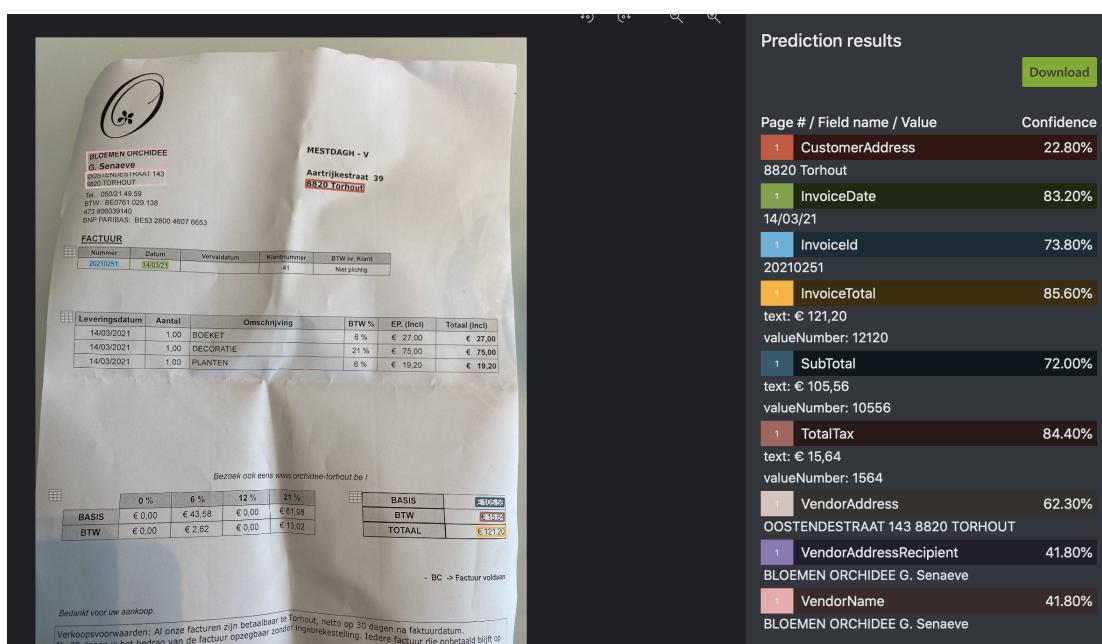
Textract genereert geen nieuwe data. Daarmee bedoelt men dat de gedetecteerde sleutel-waardeparen het eindproduct zijn. Er worden geen sleutels voorspeld op basis van de plaats lay-out en waarden. Dit is iets wat andere pakketten wel aanbieden.

Het is echter mogelijk om meerdere services van AWS in een pipeline te combineren om de gewenste functionaliteit te benaderen. De kost hiervan is echter een verlies aan gebruiksvriendelijkheid die door andere oplossingen wel voorzien is. Een andere bijkomende kost vloeit voort uit de bijkomende te investeren tijd in het opstellen van zo'n pipeline. Dit zou voor de casus een extra financiële kost betekenen. Een voorbeeld van een volledig uitgewerkte pipeline is in de literatuur momenteel niet beschikbaar, maar Engdahl (2008) illustreert het aanmaken van een pipeline die informatie uit documenten haalt en er vervolgens algoritmes op toepast. Eventueel later onderzoek kan hier verder op ingaan.

3.3.4 Azure Form Recognizer

Azure Form Recognizer is net zoals Textract een cloudservice. Men kan ermee gegevensverwerkingssoftware bouwen met behulp van machine learning-technologie. De functionaliteiten omvatten onder andere tekst extraheren, sleutel- waardeparen identificeren, tabelgegevens en relaties onderscheiden. Het pakket is aanspreekbaar via een REST-api of m.b.v. een SDK. Dit wijst op een lage toegangsdrempel.

Het is out-of-the-box in staat om onderstaande analyse uit te voeren.



In bovenstaande foto kent het softwarepakket een sleutel toe aan de data op het document. Dit oogt krachtiger dan de variant van Amazon aangezien deze de sleutels niet out-of-the-box voorspelt. Een ander voordeel is de ondersteuning van documenten in het Nederlands.

Men merkt ook op dat niet alle gegevens correct gelabeld werden. Training van het model is dus vereist. Microsoft voorziet hiervoor een eenvoudig te gebruiken tool die los staat van besturingssysteem. Het is namelijk beschikbaar in Docker.¹ Voor een minimale implementatie van Form Recognizer moeten er enkele services op Azure geconfigureerd worden. Ten eerste hebben we uiteraard Form Recognizer nodig. Ten tweede moet men over een Storage Account beschikken.² Deze BLOB storage wordt gebruikt om de trainingsdata voor het model op te slaan.

Het resultaat van een werkend model is een JSON bestand met de door OCR geëxtraheerde tekst en de voorgespelde labels door de ML model. Het voordeel van dit formaat is de herbruikbaarheid in systemen zoals een eigen stuk software. Telkens wanneer een bestand door de pipeline vloeit, wordt de zekerheid van de gemaakte classificatie vermeld. Deze

¹<https://www.docker.com/>

²<https://azure.microsoft.com/en-us/services/storage/blobs/>

extra data kan eventueel gebruikt worden om een grens te stellen voor menselijke controle. (Farley, 2021)

```
"documentResults": [
  {
    "docType": "custom:second",
    "modelId": "9473bccc-dd21-4eb7-9ad8-2bd723434bcf",
    "pageRange": [
      1,
      1
    ],
    "fields": {
      "naam verkoper": {
        "type": "string",
        "valueString": "Schoonmaakbedrijf Degryse",
        "text": "Schoonmaakbedrijf Degryse",
        "page": 1,
        "boundingBox": [
          251,
          47,
          581,
          47,
          581,
          74,
          251,
          74
        ],
        "confidence": 1,
        "isText": true
      }
    }
  }
]
```

Op vlak van personalisatie oogt de software veelbelovend. Het is mogelijk eigen documenten te gebruiken tijdens de trainingsfase. Dit zou ervoor kunnen zorgen dat de nauwkeurigheid op gelijkaardige documenten enorm toeneemt.

Deze, op artificiële intelligentie gebaseerd dienst, komt, net zoals andere diensten Azure, met een gratis en een betalende versie. Het is gratis tot 500 pagina's per maand. Wanneer men meer capaciteit nodig heeft, is er een optie die €43,15 per 1000 bladzijdes kost.

3.3.5 Open source route

Tesseract is een OCR pakket. Het bevat op zich dus geen Machine Learning technieken om classificaties van data te maken. De metadata blijven met enkel de toepassing van Tesseract dus een vraagteken. Ha e.a. (2018) demonstreert OCRMiner, een toepassing die op de door Tesseract gegenereerde data werkt. Het doel van dit pakket is metadata voor de data te voorzien. Deze is in 2021 niet vrijgegeven. De implementatie van een pipeline die vergelijkbaar is met het resultaat van Ha e.a. (2018) vergt niet alleen veel tijd, maar de beschreven accuraatheid bereikt slechts 80,1%. Deze nauwkeurigheid betekent dat voor 19,9% van de blokken een menselijke input nodig is, ook dit kost tijd.

3.3.6 Keuze

Aan de hand van vorige onderdelen lijkt het duidelijk dat Azure Form Recognizer de meest interessante optie is. De ondersteuning voor Nederlands, de mogelijkheid tot personalisatie en de toegankelijkheid spreken boekdelen.

3.4 Nauwkeurigheid

Dit onderdeel somt relevante manieren op om de accuraatheid van de gebruikte classificatiealgoritmes te beschrijven.

3.4.1 Initiële accuraatheid

De initiële accuraatheid beschrijft de mate waarin software aanwezige informatie uit een factuur extraheert en correct classificeert. Om dit te berekenen gebruikt men volgende parameters:

Parameters	
Naam	Verklaring
ne	aantal verkeerde of overgelaten classificaties
nb	aantal tekstblokken op document
a	accuraatheid in %

$$a = 100 - (ne/nb) \quad (3.4)$$

3.4.2 Accuraatheid met naverwerking

De maatstaf die in dit onderdeel beschreven wordt, dient als hulpmiddel om op zoek te gaan naar kleine aanpassingen aan het resultaat van de software die voor een zekere verbetering zorgen. Het is waardevol om op zoek te gaan naar fouten van deze aard omdat ze op een eenvoudige manier verbetering kunnen brengen.

In sommige gevallen is het mogelijk dat kleine aanpassingen, zoals het verwijderen van een karakter dat herhaaldelijk verkeerd voorkomt, voor een verhoogde nauwkeurigheid zorgen. Dit is echter alleen mogelijk wanneer de fout voorspelbaar van aard is.

3.5 Experimenten

In dit onderdeel worden enkele experimenten met Azure Form Recognizer uitgevoerd. Het doel is om na te gaan hoe accuraat, gebruiksvriendelijk en personaliseerbaar het is.

3.5.1 Methodologie van de experimenten

In de volgende onderdelen worden enkele experimenten uitgevoerd om de nauwkeurigheid van Azure Form Recognizer te valideren. Deze studie maakt gebruik van een kwantitatief onderzoek om deze uit te voeren. Het is vanzelfsprekend voor de casus dat facturen als invoer gebruikt worden. De verklaring hiervoor is dezelfde als in onderdeel 2.6.

Het vergaren van de data verloopt als volgt: een demoproject⁵ wordt opgestart, de gebruiker kiest het gewenste bestand waar de test op uitgevoerd wordt en de output wordt in een tekstbestand opgeslagen. Op deze data worden de formules in 3.4 uitgevoerd. Deze maatstaven worden opgeslagen voor verdere verwerking.

De test maakt gebruik van 31 verschillende facturen als testdata. Het gaat om de accuraatheid zoals uitgedrukt in 3.4. Deze cijfers worden vervolgens vergeleken met resultaten van voorgaande studies.

Ten slotte worden de data geanalyseerd door middel van statistische methodes in R. Er wordt een boxplot gemaakt om informatie zoals gemiddelde, minimum, maximum, mediaan en het eerste quartiel op een eenduidige manier weer te geven.

Tijdens deze experimenten wordt inclusief trainingsdata 36 facturen gebruikt.⁶

3.5.2 Benodigdheden

In deze sectie worden de architectuur en benodigdheden voor het uitvoeren van de experimenten zowel als het implementeren van de PoC besproken. Er worden meerdere tools gebruikt. Om deze experimenten/implementatie na te bootsen zijn deze dan ook vereist.

Azure

Ten eerste moet men zich registreren voor een account op het Azure platform⁷. Wanneer de registratie voltooid is, dient men naar het portaal te navigeren⁸. Dan kan men een resource toevoegen. Men kiest Form Recognizer. Daarna wordt gevraagd om een resourcegroep, regio en naam te kiezen. Deze kan men vrij kiezen.

De volgende vereiste is een opslagaccount op Azure. Deze kan men aanmaken door op het portaal resource maken te kiezen en vervolgens opslagaccount te kiezen. Voor de beste performantie, kiest men dezelfde resourcegroep en regio als de Form Recognizer.

⁵De manier van opzetten wordt in onderdelen 3.5.3 en 3.5.3 beschreven

⁶<https://drive.google.com/drive/folders/1lhG7IZjDhgBEn9ayXMuBQRr77V7Ac2jn?usp=sharing>

⁷<https://www.azure.com>

⁸<https://www.portal.azure.com>

Dataset

Zoals elke ML toepassing is een voldoende grote dataset nodig. In deze casus zijn er facturen van 5 bedrijven. Deze facturen hebben een verschillende lay-out. Om optimale resultaten te bekomen, hebben we meerdere exemplaren per bedrijf nodig. Verder in dit hoofdstuk wordt de invloed van de grootte van de trainingsdataset beoordeeld.

Om deze dataset van labels te voorzien, bestaat de 'sample labeling tool'⁹.

Software

Om de 'sample labeling tool' op te kunnen starten is Docker¹⁰ vereist. Hiernaast is Azure Storage Explorer aan te raden aangezien het de communicatie met het opslagaccount vergemakkelijkt. Het maakt het genereren van een handtekening voor gedeelde toegang eenvoudiger.

3.5.3 Opbouw

Deze sectie bevat de verschillende stappen die nodig zijn om een werkende applicatie te bouwen.

Klaarstomen data

Om facturen in het opslagaccount op te slaan, opent men de Azure Storage Explorer en meldt men zich aan met het Azure account. Nu is het mogelijk om een BLOB container aan te maken. Wanneer deze aangemaakt is onder standaardinstellingen is het mogelijk om hier de PDF of fotobestanden van facturen toe te voegen. Vervolgens klikt men met de rechtermuisknop op de container en kiest men voor: "handtekening voor gedeelde toegang ophalen". Deze bewaart men voor de volgende stap. In de volgende stap open men Docker Desktop en voert men onderstaand commando uit.

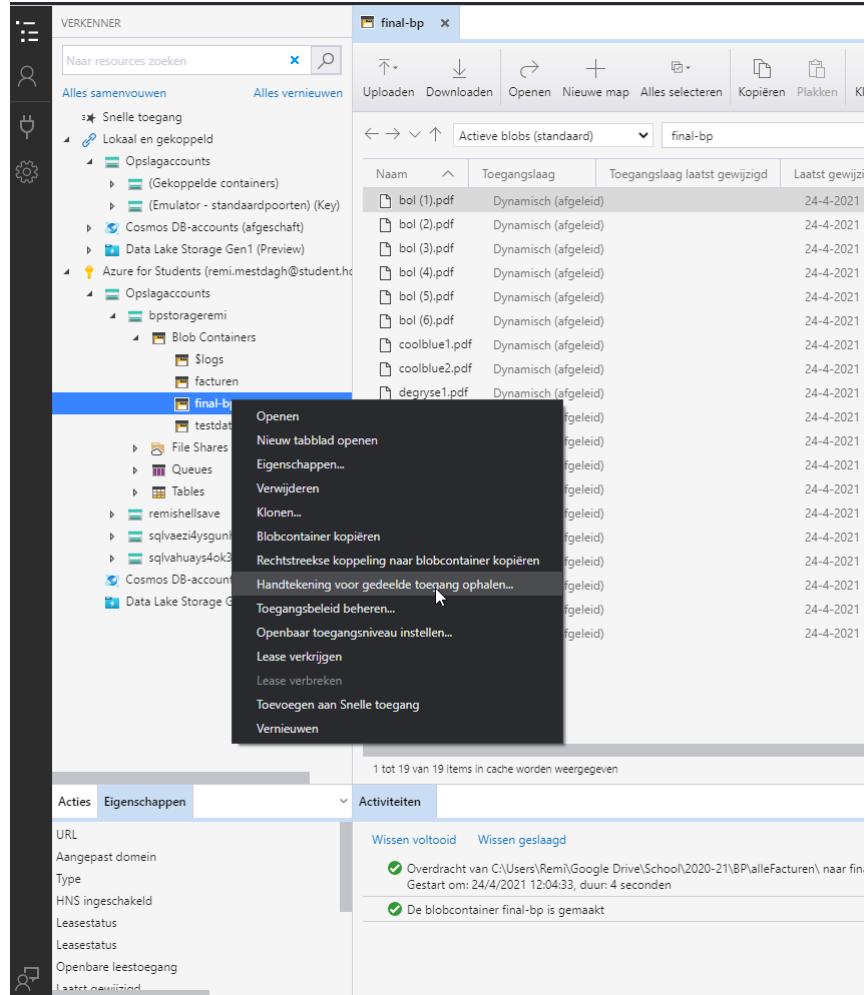
```
docker pull  
mcr.microsoft.com/azure-cognitive-services/  
custom-form/labeltool:latest-preview
```

Nu is het mogelijk om met de 'sample labeling tool' aan de slag te gaan. Wanneer deze in de browser geopend is, bestaat de mogelijkheid om een aangepast model te bouwen. Hier is het vereist om een nieuw project aan te maken. De naam en beschrijving zijn vrij te kiezen, maar de overige informatie is van groter belang. Als source connection dient men een nieuwe verbinding aan te maken. In het veld SAS URI vult u de in de vorige stap verkregen handtekening in. Wanneer deze verbinding geconfigureerd is, dient men de

⁹<https://docs.microsoft.com/en-us/azure/cognitive-services/form-recognizer/quickstarts/label-tool?tabs=v2-1#set-up-the-sample-labeling-tool>

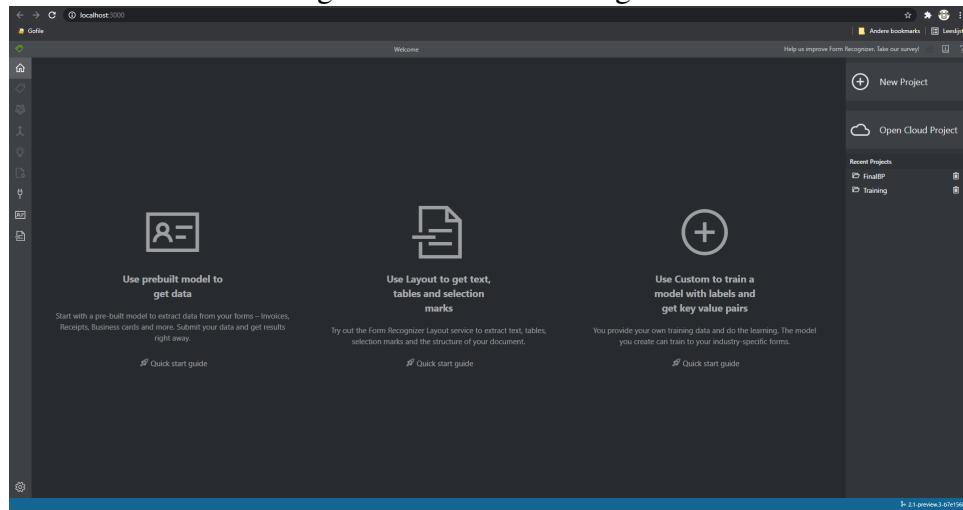
¹⁰<https://www.docker.com>

Figuur 3.1: Azure labeling tool

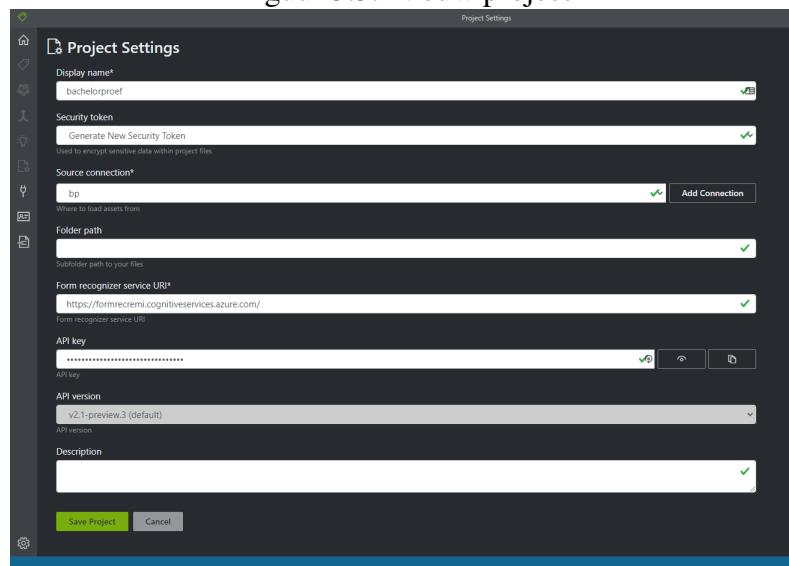


overige info zoals de URI van de API en de sleutel in te vullen. Deze zijn beiden te vinden op het Azure portaal wanneer men de Recognizer service selecteert. Nadat deze informatie ingevuld is, kan het project opgeslagen worden.

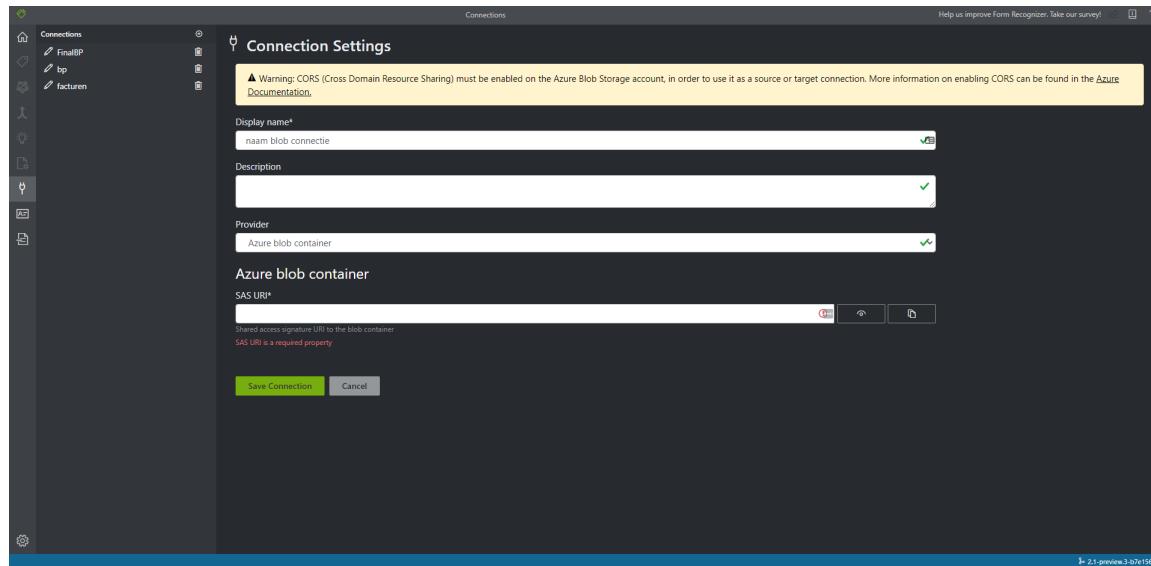
Figuur 3.2: Azure labeling tool



Figuur 3.3: Nieuw project



Figuur 3.4: Nieuwe verbinding

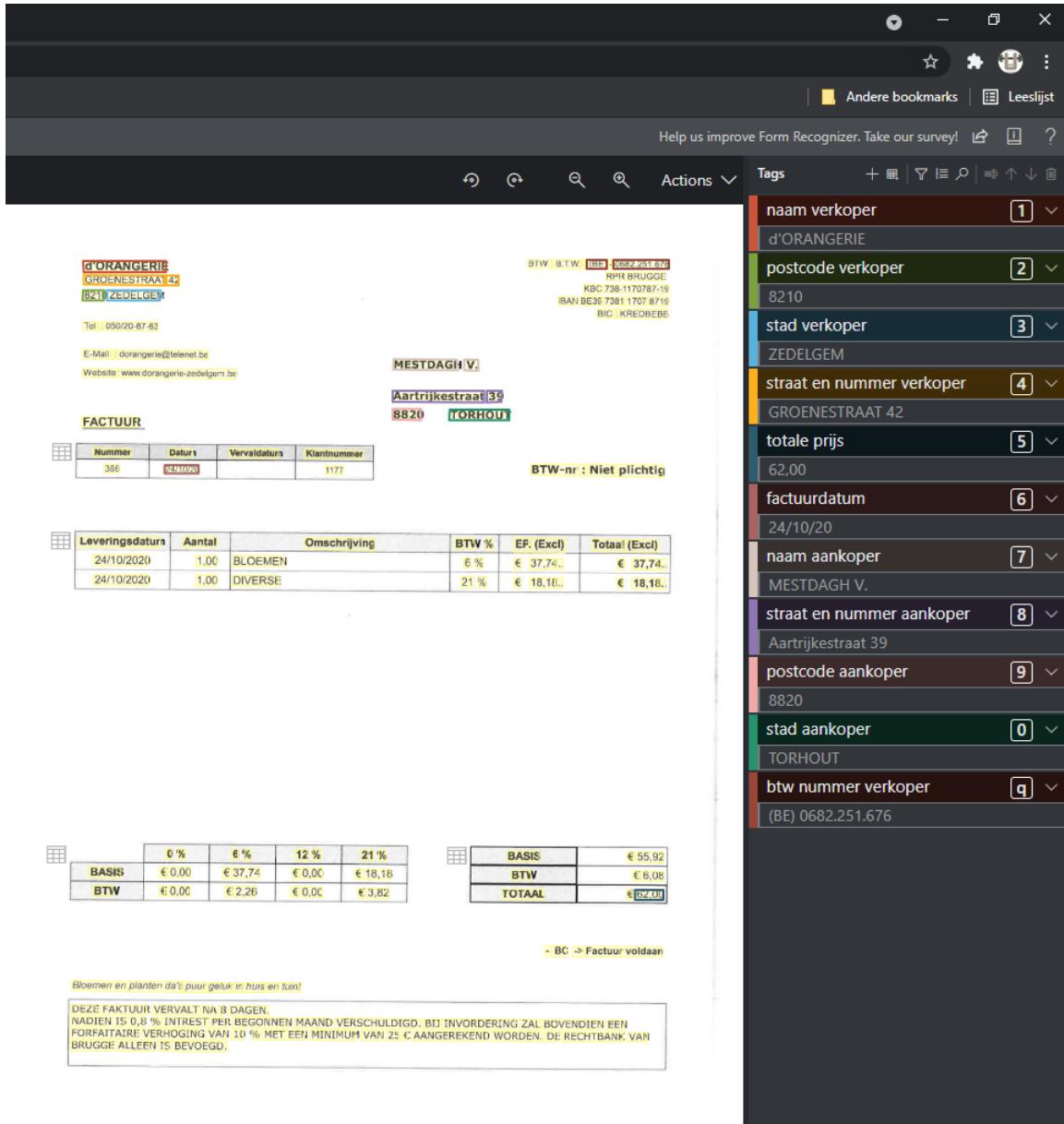


Nadat het project aangemaakt is, worden alle facturen door de software geanalyseerd op basis van de lay-out. Het resultaat hiervan is een in logische tekstblokken opgedeeld document. Nu is het mogelijk om blokken te selecteren en aan te duiden over welk soort data het precies gaat. Dit doet men door op het icoon naast tags te klikken en een klasse, zoals "naam aankoper" aan te maken. Wanneer men op het gemaakte label klikt, is het mogelijk aan te geven dat het om tekst of cijfers gaat. Dit is aangeraden in de mate waarin het mogelijk is. Door een tekstblok aan te klikken en vervolgens op de nieuwe klasse aan de rechterkant te klikken, wordt deze gedefinieerd als een object ervan. Deze stappen herhaalt men totdat alle documenten en alle gewenste klassen een verbinding hebben.

Model trainen

Nadat de data klaargestoomd zijn, moet een model getraind worden. Dit kan bereikt worden door op de 'sample labeling tool' het derde icoon aan te klikken. Nu kan men een naam aan het model geven en op 'Train' klikken. Hier zijn minstens 5 gelabelde documenten nodig. Tijdens deze experimenten wordt van 4 bedrijven telkens 1 factuur gebruikt. Hier wordt echter een uitzondering op gemaakt bij de facturen van Bol.com omdat zodat aan de minimumvereiste voldaan wordt.

Figuur 3.5: Voorbeeld document met labels



3.5.4 Analyse

In een volgende stap kan het model getest worden. Dit kan door op "Analyse" te klikken. Men kiest een lokaal document waarop OCR en classificatiealgoritmes uitgevoerd worden.

In 3.6 werd een factuur van Bol geanalyseerd. Tijdens de trainingsfase werden er reeds van dit bedrijf gebruikt. Aan de rechterkant van de figuur zijn de geëxtraheerde waarden zichtbaar. Deze zijn correct behalve de straat van de verkoper. Dit komt doordat een komma in het tekstblok van de trainingsdata staat. Hier is in de huidige staat van de software niet veel aan te veranderen. Het is niet mogelijk om manueel regio's te tekenen

die als tekstblokken herkend moeten worden.

Figuur 3.6: Eerste analyse

The screenshot shows the Microsoft Form Recognizer interface with a dark theme. On the left, a receipt from bol.com is displayed. At the top of the receipt, it shows delivery information: Dhr. dries mestdagh, Aartrijkstraat 39, 8820 Torhout, België. Below this, it shows company details: bol.com b.v., Papendorpseweg 100, 3528 BJ Utrecht, KVK Utrecht: 32147382, BTW nr. BE: BE0824148721, IBAN: NL27INGB0000026500, BIC: INGBNL2A. The main section is titled "Factuur" with the tagline "alles op een rijtje". It lists a single item: Assassin's Creed: Odyssey - PS4, quantity 1, price € 27,99, with a 21% discount. The total amount is shown as € 27,99. At the bottom, a thank you message reads: "Bedankt voor je aankoop bij bol.com". On the right side of the interface, there is an "Analysis" tab with a "Run analysis" button. Under "Prediction results", it shows the confidence levels for various extracted fields. The results table includes columns for "Page # / Field name / Value" and "Confidence". Some entries have a yellow background, indicating they are likely correct. The confidence values range from 94.50% to 99.50%.

Page # / Field name / Value	Confidence
1 naam verkoper	99.50%
bol.com b.v.	
1 postcode verkoper	99.50%
3528	
1 stad verkoper	99.50%
Utrecht	
1 straat en nummer verkop...	99.50%
text: Papendorpseweg 100,	
valueString: Papendorpseweg100	
1 totale prijs	99.50%
27,99	
1 factuurdatum	99.50%
18 december 2019	
1 naam aankoper	94.50%
dries mestdagh	
1 straat en nummer aanko...	99.50%
Aartrijkstraat 39	
1 postcode aankoper	99.50%
8820	
1 stad aankoper	99.50%
Torhout	
1 btw nummer verkoper	99.50%
BE 0824148721	

Figuur 3.7: Analyse met fout

Page # / Field name / Value	Confidence
1 naam verkoper	99.00%
d'ORANGERIE	
1 postcode verkoper	99.00%
050/20-87-63	
1 stad verkoper	99.50%
ZEDELGEM	
1 straat en nummer verkoper	99.00%
GROENESTRAAT 42	
1 totale prijs	99.50%
44,00	
1 factuurdatum	99.50%
25/11/20	
1 naam aankoper	68.90%
MESTDAGH V.	
1 straat en nummer aankoper	99.50%
Aartrijkestraat 39	
1 postcode aankoper	99.50%
8820	
1 stad aankoper	99.00%
TORHOUT	
1 btw nummer verkoper	99.00%

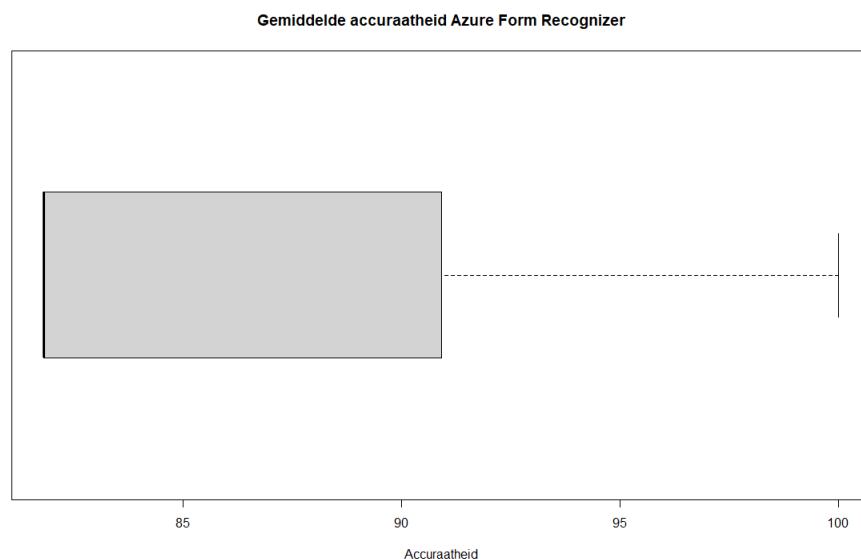
3.5.5 Nauwkeurigheid meten

Om de nauwkeurigheid te meten gaat men als volgt te werk: er wordt een testdataset voorzien van 31 facturen. Deze lijken goed op de getrainde dataset omdat ze vanuit 1 van de 4 bedrijven komen.

Vervolgens kan men deze facturen toevoegen aan het opslagaccount waar de door Form Recognizer gebruikte bestanden te vinden zijn. Daarna kan men de labeling tool gebruiken om alle nieuwe bestanden automatisch van labels te voorzien.

Het gemiddelde resultaat ziet er als volgt uit:

Figuur 3.8: Nauwkeurigheid Azure Form Recognizer



Zoals op de figuur te zien is, bedraagt de gemiddelde accuraatheid 86,65%.

Wanneer men de gemaakte fouten onder de loep neemt, ziet men dat deze van een gelijke aard zijn. Zo wordt in facturen van Coolblue vaak een extra komma toegevoegd aan het adresveld. Dit vloeit voort uit een tekortkoming van de "sample labeling tool". Het is momenteel niet mogelijk om regio's te tekenen om aangepaste tekstblokken aan te maken. Gebruikers van de software zijn gebonden aan de gegenereerde blokken.

Wanneer men naar de output kijkt, ziet men dat in sommige gevallen het tekstblok voor "naam aankoper" verkeerd geclasseerd wordt. Bij nader onderzoek, ziet men dat sommige testdocumenten er anders uitzien. Onderstaande figuren illustreren dit.

Figuur 3.9: Factuur a



Figuur 3.10: Factuur b



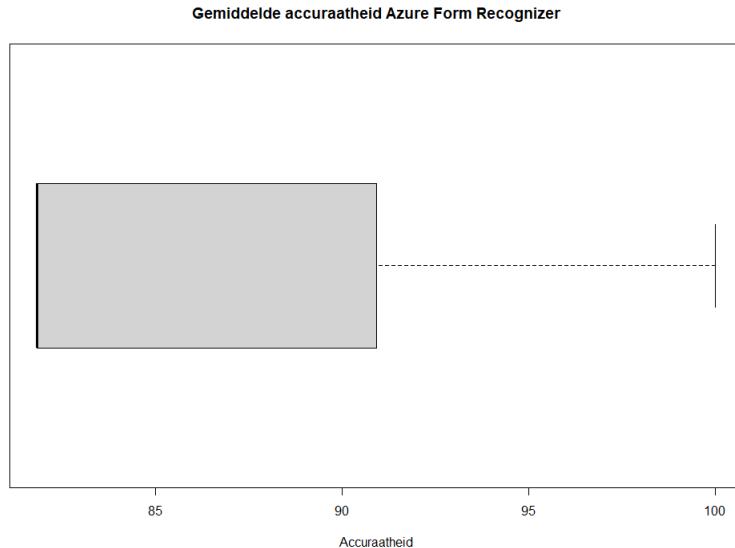
Het is duidelijk dat de lay-out in beide facturen verschillend is. Dit is hoogstwaarschijnlijk de oorzaak voor de lage vertrouwenswaarde en/of verkeerde classificatie.

Een mogelijke oplossing is om een applicatie te ontwikkelen die de API van de Form Recognizer aanspreekt en op het resultaat enkele bewerkingen uit te voeren die deze extra tekens verwijdert. Dit wordt in een volgend hoofdstuk onderzocht.

Remedie

Wanneer men een factuur die voldoet aan de lay-out in figuur 3.10 opneemt als trainingsdata voor ons model en voorgaande test opnieuw uitvoert, krijgt men volgend resultaat.

Figuur 3.11: Nauwkeurigheid Azure Form Recognizer na veranderingen in model



Zoals op de figuur te zien, bedraagt de gemiddelde accuraatheid 86,93%. Dit is een kleine verbetering, maar daarmee is de kous nog niet af. Onderstaande figuur toont een vergelijking tussen de output van het vorige en huidige model. Het valt op dat nog steeds fouten gemaakt worden tijdens het classificeren van "naam aankoper". In een volgend hoofdstuk tracht men dit te verhelpen.

Figuur 3.12: Vergelijking output)

Field 'straat en nummer aankoper: Value: 'Staatsbaan 48B Confidence: '0,995	Field 'stad aankoper: Value: 'HANDZAME Confidence: '0,995
Field 'straat en nummer verkoper: Value: 'Papendorpsweg 100, Confidence: '0,995	Field 'straat en nummer aankoper: Value: 'Staatsbaan 48B Confidence: '0,995
Field 'postcode verkoper: Value: '3528 Confidence: '0,995	Field 'naam aankoper: Value: 'ROJO bvba Confidence: '0,995
Field 'stad verkoper: Value: 'Utrecht Confidence: '0,99	Field 'factuurdatum: Value: '27 november 2020 Confidence: '0,995
Field 'postcode aankoper: Value: '8610 Confidence: '0,995	Field 'totale prijs: Value: '341,00 Confidence: '0,995
Field 'totale prijs: Value: '341,00 Confidence: '0,995	Field 'naam verkoper: Value: 'bol.com b.v. Confidence: '0,995
Field 'stad aankoper: Value: 'HANDZAME Confidence: '0,99	Field 'postcode verkoper: Value: '3528 Confidence: '0,99
Field 'btw nummer verkoper: Value: 'BE 0824148721 Confidence: '0,99	Field 'straat en nummer verkoper: Value: 'Papendorpsweg 100, Confidence: '0,995
Field 'naam aankoper: Value: 'Dhr. Joost Roose Confidence: '0,216	Field 'postcode aankoper: Value: '8610 Confidence: '0,995
Field 'factuurdatum: Value: '27 november 2020 Confidence: '0,995	Field 'stad verkoper: Value: 'Utrecht Confidence: '0,99
Field 'naam verkoper: Value: 'bol.com b.v. Confidence: '0,99	Field 'btw nummer verkoper: Value: 'BE 0824148721 Confidence: '0,995

Hieruit leren we dat in sommige gevallen facturen van eenzelfde bedrijf genoeg kunnen afwijken zodat additionele trainingsdata vereist zijn.

3.5.6 Evaluatie

In dit onderdeel wordt het verkregen resultaat besproken. Tevens wordt Azure Form Recognizer beoordeeld op vlak van parameters zoals:

- Personalisatie
- Nauwkeurigheid
- Opvangen fouten

Personalisatie

Zoals vermeld in onderdeel 3.3.4 is een mogelijkheid tot personalisatie aanwezig. Na het uitvoeren van enkele analyses met facturen die van dezelfde verkoper komen als de

trainingsdata zien we meteen een sprong in algemeen vertrouwen en accuraatheid van de software.

Nauwkeurigheid

Ook de algemene nauwkeurigheid laat het niet afweten. De experimenten wijzen uit dat een goede naverwerking tot een uitstekend resultaat leiden.

Oppangen fouten

Er bestaat geen ingebouwde waarschuwing of equivalent om mogelijke fouten te corrigeren. De software bezit echter wel de mogelijkheid om deze zelf te implementeren. Na elke analyse is een niveau van vertrouwen in de classificatie zichtbaar. Verdere implementaties kunnen hiervan gebruikmaken om fouten te vermijden en mogelijk extra manuele input te verlagen.

Een laag niveau van vertrouwen kan wijzen op facturen die van nieuwe leveranciers komen. Er zijn dus nog geen getrainde exemplaren. Een goede implementatie kan dus ook op deze manier gebruikmaken van dit vertrouwensniveau om de applicatie robuust te maken en de gebruiker aan te sporen het model te trainen.

3.6 Conclusie en betekenis voor de casus

De combinatie van OCR en ML technologieën oogt veelbelovend als oplossing voor de casus. In het volgende hoofdstuk volgt een implementatie van Azure Form Recognizer. Deze tracht een oplossing aan de casus te geven.

4. Implementatie

In dit hoofdstuk wordt de implementatie van een C# applicatie besproken. Het doel is een oplossing aan de casus te bieden. Op basis van voorgaande hoofdstukken wordt Azure Form Recognizer gebruikt om de problematiek in de casus aan te kaarten. Dit is eerst en vooral een Proof of concept (PoC). De initiële opbouw volgt onderdelen 3.5.2 en 3.5.3.

4.1 Aanspreken in C#

Na vorige stappen te hebben doorlopen, kan een console applicatie in Visual Studio 2019 aangemaakt worden.

4.1.1 Software

Deze implementatie¹ maakt gebruik van Visual Studio 2019 in een Windows 10 omgeving. Deze editor wordt verrijkt met enkele packages. Ten eerste is het ten zeerste aangeraden om Azure.AI.FormRecognizer te gebruiken om de communicatie met de API van de form recognizer te vergemakkelijken. Vervolgens is het importeren van Azure.Storage.Blobs ook een must. Het versnelt het uploaden en downloaden van foto's van facturen die later in het proces gebruikt zullen worden.

¹<https://github.com/remimestdagh/bp-implementation-presentation>

4.1.2 Verklaring

Nadat het project aangemaakt is en de gebruikte NuGet packages toegevoegd zijn, moeten enkele attributen gedefinieerd worden. Ten eerste dient men de namen van de storage containers die de testdata en getrainde data bevatten in te vullen. Daarna moet de verbindingsreeks voor het opslagaccount toegevoegd worden. Om aanvragen naar de Form Recognizer te sturen, dient men de unieke url en sleutel toe te voegen. Deze zijn allen op Azure onder de desbetreffende service te vinden.

Wijzigingen die in de "AnalysePdfForm"methode gemaakt werden, zorgen voor het verwijderen van problemen die zich in het vorige hoofdstuk voordeden. Dit gaat, ter herhaling, om de redundante komma's of "Dhr." in het "naam aankoper"veld. Nadat deze doorgevoerd werden is de nauwkeurigheid 100% op de testdata.

De applicatie bestaat uit een menu met enkele beschikbare opties.

- 1: Toont uitleg rond het gebruik van de applicatie
- 2: Hoofdfunctie van de applicatie, geeft de mogelijkheid om een lokaal bestand te selecteren om aan de hand van de getrainde modellen te classificeren
- 3: Geeft alle getrainde modellen weer
- 4: Toont alle modellen en geeft de optie om te verwijderen
- 5: Toont de inhoud van de testdata blob container waar de geteste data opgeslagen wordt
- 0: Programma afsluiten

De classificatieprocedure verloopt als volgt: men plaatst een factuur in de data map van het project. Daarna kiest men optie 2 wanneer het programma draait. Dan wordt men gevraagd om de naam van het bestand in te voeren. Dit bestand wordt vervolgens naar een blob container uploadt. Dan kiest men het gewenste getrainde model. Dit stuurt een API-aanvraag naar de Form Recognizer. Ten slotte ontvangt men een antwoord van de service met daarin een JSON in onderstaande vorm.

Figuur 4.1: Voorbeeldantwoord PoC

```

Form of type: custom:2meiOptimal
Field 'postcode verkoper':
    Value: '3528
    Confidence: '0,99
Field 'postcode aankoper':
    Value: '8610
    Confidence: '0,995
Field 'totale prijs':
    Value: '37,00
    Confidence: '0,995
Field 'straat en nummer aankoper':
    Value: 'Staatsbaan 48B
    Confidence: '0,995
Field 'naam aankoper':
    Value: 'ROJO bvba
    Confidence: '0,995
Field 'naam verkoper':
    Value: 'bol.com b.v.
    Confidence: '0,995
Field 'factuurdatum':
    Value: '12 februari 2021
    Confidence: '0,995
Field 'straat en nummer verkoper':
    Value: 'Papendorpseweg 100,
    Confidence: '0,995
Field 'stad aankoper':
    Value: 'HANDZAME
    Confidence: '0,995
Field 'btw nummer verkoper':
    Value: 'BE 0824148/21
    Confidence: '0,995
Field 'stad verkoper':
    Value: 'Utrecht
    Confidence: '0,99
Table data:
Table 0 has 3 rows and 8 columns.
Cell (0, 0) contains text: 'Omschrijving'
Cell (0, 1) contains text: 'Aantal'
Cell (0, 2) contains text: 'Prijs/st'
Cell (0, 3) contains text: 'Korting'
Cell (0, 4) contains text: 'Bedrag'
Cell (0, 5) contains text: 'BTW'
Cell (0, 6) contains text: 'BTW'
Cell (1, 0) contains text: 'Dopper Steel Drinkfles - 800 ml - Roestvrij staal'
Cell (1, 1) contains text: '1'
Cell (1, 2) contains text: '? 24,50'
Cell (1, 3) contains text: ''
Cell (1, 4) contains text: '? 24,50'
Cell (1, 5) contains text: '21%'
Cell (1, 6) contains text: '?'
Cell (1, 7) contains text: '4,25'
Cell (2, 0) contains text: 'Dopper Original Drinkfles - 450 ml - Sea Green'
Cell (2, 1) contains text: '1'
Cell (2, 2) contains text: '? 12,50'
Cell (2, 3) contains text: ''
Cell (2, 4) contains text: '? 12,50'
Cell (2, 5) contains text: '21%'
Cell (2, 6) contains text: '?'
Cell (2, 7) contains text: '2,17'

```

Dit is gelijkaardig aan de output van de experimenten in het voorgaande hoofdstuk.

4.2 Conclusie

Deze implementatie vormt een voldoende en krachtige oplossing voor de casus. De beschikbare packages maken de implementatie intuïtief en zorgen voor een kortere duur. De behaalde accuraatheid is nagenoeg perfect. Het is echter op dit moment niet onderzocht of dit vertaalbaar is naar software die een groter aantal soorten facturen dient te verwerken.

5. Conclusie

5.1 Bijdrage

In dit onderdeel wordt de bijdrage van deze paper aan het onderzoeks veld bekeken.

5.1.1 Onderzoeks vragen

In dit onderdeel wordt een antwoord op de onderzoeks vragen geformuleerd.

Vergelijking open source met betalende OCR

De vergelijking van open source met betalende OCR technologie diende gemaakt te worden. In de literatuur zijn reeds gelijkaardige vergelijkingen beschikbaar. Wat deze paper onderscheidt van voorgaand onderzoek is het gebruik van facturen als bronmateriaal. Deze hebben, zoals besproken, unieke eigenschappen. De besprekking is in onderdeel 2.5 te vinden.

De studie wees uit dat de performantie gelijkaardig is, met marginaal hogere nauwkeurigheid voor de betalende variant. Deze marginaal hogere nauwkeurigheid wordt overheerst door de open source licentie van de gratis variant, Tesseract. Deze heeft echter weinig impact op te casus aangezien OCR op zich niet volstaat.

Wat is de technologie achter OCR?

In 2.2 worden onderdelen, die in hedendaagse OCR software te vinden zijn, beschreven. OCR bestaat uit een groeiend aantal technologieën als de-skew, despeckle, binarisation, line removal en line and word detection... Deze worden sequentieel uitgevoerd om documenten klaar te stomen voor een volgende fase, de daadwerkelijke tekstherkenning. Patroonherkenning en feature extraction zijn de meest prominente.

Welke stappen zijn nodig om een werkende applicatie te schrijven die OCR gebruikt om facturen te verwerken?

Dit onderzoek wijst uit dat louter OCR niet voldoende is om dit doel te bereiken. Als aanvulling opteert deze studie voor Machine learning (ML).

Welke software is beschikbaar en wat zijn hun sterktes?

Volgende lijst bevat alle besproken software:

- Tesseract
- CuneiForm
- ABBYY FineReader
- Asprise OCR
- OmniPage
- Dynamsoft OCR
- Google Cloud Vision
- AnyDoc Software
- Ocrad
- OCropus
- OCRFeeder
- Azure Form Recognizer
- Tesseract
- AWS Textract
- DocParser
- Google Cloud Auto ML Vision Object Detection

De sterktes werden beschreven aan de hand van de requirements voor de casus. Zo werden de technologieën opgedeeld aan de hand van hoge nauwkeurigheid, ondersteuning voor Nederlands, uitbreidbaarheid, up-to-date en type licentie.

In onderdeel 2.5.2 wordt deze toetsing verricht.

Welke OCR heeft de hoogste snelheid en accuraatheid?

ABBYY FineReader behaalde tijdens experimenten de hoogste nauwkeurigheid. Deze wordt door de literatuur ondersteund.

Welke OCR software wordt aangeraden voor deze use case en waarom?

Zoals eerder vermeld is aanvullende software nodig voor deze use case. Onderzoek wees uit dat Azure Form Recognizer de ideale kandidaat is. Het voldoet aan een groot aantal requirements. Zo staat het open voor personalisatie, is het toegankelijk in C#, biedt het ondersteuning voor Nederlandse documenten, heeft het een hoge nauwkeurigheid en is er een mogelijkheid om eventuele fouten op te vangen.

Hoe personaliseerbaar zijn de beschikbare varianten?

Deze studie toont aan dat Azure Form Recognizer een grote mogelijkheid tot personalisatie heeft. Andere varianten beschikken in mindere mate over deze optie.

5.1.2 Ondervindingen

Na de succesvolle implementatie van een oplossing voor de casus rijst de vraag of deze toepasbaar is voor projecten die ondersteuning dient te bieden voor een groter aantal leveranciers. De toename in soorten facturen zou voor andere resultaten kunnen zorgen.

Tijdens de experimenten en implementatie werden enkele tekortkomingen van Azure Form Recognizer duidelijk. Er is namelijk geen eenvoudige manier om de grootte van de tekstblokken te wijzigen. Dit kan voor ongewenste karakters zorgen.

5.1.3 Verder onderzoek

In de toekomst kunnen enkele zaken besproken worden. De mogelijkheid om een pipeline op te stellen van AWS services of de open source tegenhangers kan onderzocht worden. Er bestaat een mogelijkheid dat deze de nauwkeurigheid of kosten kunnen verbeteren.

Zoals eerder vermeld is een grotere scope van facturen een interessant startpunt voor onderzoek. Dit kan tot een andere evaluatie leiden.

A. Onderzoeksvoorstel

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

A.1 Introductie

De grootste bottleneck tijdens het dagelijkse werk van boekhouders is het inscannen en verwerken van facturen die ze op papier ontvangen. De boekhouder moet in dit scenario dan vaak manueel de informatie die op deze facturen te vinden is, invoeren op een digitaal platform. Vandaag gebruikt men software dat het verwerken hiervan vergemakkelijkt. Het probleem hierbij is echter dat deze software vaak niet gratis is. Hedendaagse beschikbare tools hebben mogelijkheden om de achterliggende Machine learning (ML) modellen bij te werken naar persoonlijke behoeftes. Hoewel de meeste facturen gelijkaardige eigenschappen hebben, zijn er vaak verschillen met de norm. Daarom is personalisatie een must. In het dagelijks leven hebben veel bedrijven baat bij het sneller inlezen en digitaliseren van facturen. Het is echter niet vanzelfsprekend om als ontwikkelaar hier software voor te schrijven. Nagenoeg elke speler op de markt heeft een eigen variant uitgebracht. Deze hebben vaak hun eigen sterktes en zwaktes, zo bekomt de Tesseract engine¹ niet altijd de juiste uitkomst wanneer het schrift van de norm afwijkt (Akram e.a., 2014). Uit dit alles blijkt dat het maken van een keuze niet makkelijk is.

In deze paper zullen volgende onderzoeks vragen uitgewerkt worden:

- Wat is de technologie achter OCR?

¹<https://tesseract-ocr.github.io/>

- Welke stappen zijn nodig om een werkende applicatie te schrijven die OCR gebruikt om facturen te verwerken?
- Welke software is beschikbaar en wat zijn hun sterktes?
- Wat zijn de tekortkomingen van open source OCR software tegenover betalende?
- Welke OCR software heeft de hoogste snelheid en accuraatheid?
- Welke OCR software wordt aangeraden voor deze use case en waarom?
- Hoe personaliseerbaar zijn de beschikbare varianten?

A.2 State-of-the-art

A.2.1 Wat is OCR

OCR zit vandaag in een groot aantal applicaties verwerkt. Het is een Engelstalig letterwoord dat voor *Optical Character Recognition* staat. In het kort betekent dit dat men softwarematig tekst uit afbeeldingen haalt. De meeste personen zullen het zien als een magisch stukje software dat bij het inscannen van een document, de aanwezige tekst herkent en aanpasbaar maakt. OCR staat ondertussen niet meer in zijn kinderschoenen en er zijn tal van toepassingen op basis van dit oude idee gemaakt, zoals uitbreidingen naar het herkennen van gezichten, voorwerpen... In de paper komt er een overzicht van de evolutie van deze technologie. Vervolgens wordt de technologie achter hedendaagse varianten bestudeerd. (Nagy e.a., 1999)

A.2.2 Hedendaagse OCR voor ontwikkelaars

Volgens Kae e.a. (2010) is het extraheren van tekst uit ongestructureerde, *noisy* documenten een uitdaging. Daaronder vallen ook documenten die niet gescand werden met een hoge resolutie. Daarom bieden uitgevers zoals Microsoft, Google en de open source community alternatieven de mogelijkheid aan om eigen bronmateriaal te gebruiken om het model te perfectioneren. Form Recognizer² van Microsoft is een meer recente technologie die aan populariteit wint. De integratie met hedendaagse programmeertalen zoals C# is aantrekkelijk. Ook Google biedt een oplossing onder de naam *Vision AI*³ aan. Het onderzoek biedt een vergelijking tussen de capaciteiten van de software en de aangeboden prijsmodellen. Objectief gezien kan deze prijs een obstakel vormen voor het ontwikkelen van applicaties.

A.3 Methodologie

In een eerste fase van het onderzoek wordt OCR zelf onder de loep genomen. De achterliggende theorie en technologie wordt bekeken. Ook de meest recente evoluties en verbeteringen worden hier bestudeerd.

²<https://azure.microsoft.com/nl-nl/services/cognitive-services/form-recognizer/>

³<https://cloud.google.com/vision>

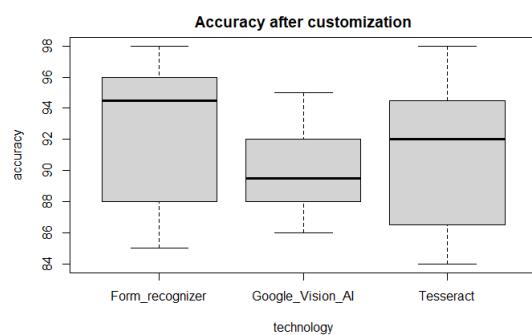
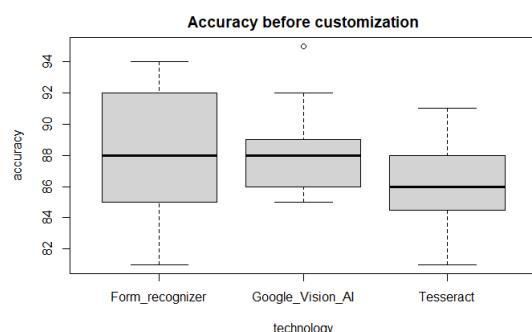
Bij een tweede luik wordt een *long list* van beschikbare OCR tools opgesomd. Hierna wordt deze verfijnd tot een *short list* van de beste kandidaten op basis van gedocumenteerde performantie op basis van de literatuur.

Na het opstellen van de kandidaten zullen er experimenten gedaan worden om de accuraatheid en snelheid te bepalen. Om deze testen uit te voeren, zullen de testmodules van de uitgevers van de software gebruikt worden. De meest prominente open source varianten hebben simulatiesites waar je deze tests kan uitvoeren. Er zal ook een vergelijking gemaakt worden tussen de uitbreidbaarheid aan de hand van een *use case*. De mogelijkheid om deze software in een C# applicatie te verwerken is ook een must.

Nadat de experimenten de beste kandidaat uitgewezen hebben, zal er een C# applicatie geprogrammeerd worden die de mogelijkheid heeft om gescande documenten te verwerken. Hiervoor zal Visual Studio 2019 gebruikt worden.

Ten slotte zal de accuraatheid van de applicatie onderzocht worden aan de hand van experimenten. In deze zullen facturen van verschillende bedrijven gebruikt worden.

A.4 Verwachte resultaten



A.5 Verwachte conclusies

Er wordt verwacht dat open source OCR software zoals deze van Tesseract de beste keuze is bij het programmeren van een C# applicatie om facturen te herkennen en te verwerken. *Out-of-the-box* is de accuraatheid echter niet goed genoeg en is er bij 30% van facturen nog menselijke input nodig om het gewenste resultaat te bekomen. Deze fouten zullen waarschijnlijk bestaan uit verkeerde classificaties. Het is aan te raden om een betalend platform te gebruiken wanneer de tijd die nodig is om de applicatie te maken belangrijk is. Er wordt verwacht dat de experimenten *Form Recognizer* van Microsoft als beste onder de betalende platformen verklaren op vlak van accuraatheid en snelheid. De intuïtieve manier om deze te personaliseren met eigen facturen is een enorme meerwaarde.

Wanneer Tesseract echter succesvol gemodificeerd wordt dan is het competitief met alternatieven van marktspelers. Met als direct voordeel dat er geen kosten hangen aan het volume verwerkte documenten. Nog een voordeel is dat facturen niet blootgesteld worden aan de servers van derden. Het nadeel is echter dat het uitdagend is om de maximale performantie te bereiken. Het trainen van het ML model is tijdrovender dan bij de betalende varianten.

Het schrijven van een OCR applicatie in C# is uitdagend, maar wordt tot op zekere hoogte vergemakkelijkt dankzij het grote aanbod aan tools. Onderzoek voor het maken van keuzes over de tools is een must voor het bekomen van een werkend resultaat.

B. Output OCR experimenten

B.1 Tesseract

B.1.1 Lage DPI

De taal van Tesseract staat ingesteld op Nederlands voor dit voorbeeld. De dpi van de afbeelding is 143,99.

FACTUUR. coolblue NV.

pi Borsbeeksebrug 25 Dat betaal ik zelf wel. Daan Berten

België remi mestdagh wauw coolblue.be aartrijkestraat 39 8820 torhout RPR Antwerpen

Btw Be0a67686774 Factuurnummer: 1409113302 IBAN Be07310160125666

Klantnummer: 21752541 Factuurdatum: 24 maart 2021 Ordernummer: 52517004 Orderdstum: 24 maart 2021

Bic geruBees

Artikel Aantal Prijsperstuk _BTW Prijs incl. BTW

Tello Drone (powered by DI) 1 €9900 21% €99,00

Incl Recupel: 07.01 Speelgoed -huishouseijk 1 coo

Excusief BTW esra 'subtotaal €59,00, BTW 21% €718

—_ Totaal €99,00,

Totmal €900 “Bancontact” op 24 maart 2021 €99,00

Sg Printen? Niet nodig, Je kunt je factuur altijd terugvinden in je mail of in je Mijn Coolblue account

Vibe EREN EAT

B.1.2 Beste omstandigheden

De taal van Tesseract staat ingesteld op Nederlands voor dit voorbeeld. De dpi van de afbeelding is 335.

FACTUUR. Coolblue N.V.

: Borsbeeksebrug 28

Dat betaal ik zelf wel. 2600 Berchem.

België

remi mestdagh www.coolblue.be

aartrijkestraat 39

8820 Torhout RPR Antwerpen

BTW BEO867686774

Factuurnummer: 1409113302 [BAN BEO7310160125666

Klantnummer: 21752541 BIC BBRUBEBB

Factuurdatum: 24 maart 2021

Ordernummer: 52517004

Orderdatum: 24 maart 2021

Artikel Aantal Prijs per stuk BTW Prijs incl. BTW

Tello Drone (powered by DJI) 1 € 99,00 21% € 99,00

Incl. Recupel: 07.01 Speelgoed - huishoudelijk 1 € 0,04

Exclusief BTW €81,82 Subtotaal € 99,00

BTW 21% € 17,18

Totaal € 99,00

Totaal € 99,00 “Bancontact” op 24 maart 2021 € 99,00

Printen? Niet nodig. Je kunt je factuur altijd terugvinden in je mail of in je Mijn Coolblue-account.

Altijd minimaal 2 jaar garantie. Doen we niet moeilijk over.

B.1.3 Taalondersteuning

De taal van Tesseract staat ingesteld op Engels voor dit voorbeeld. De dpi van de afbeelding is 335.

FACTUUR. Coolblue N.V.

; Borsbeeksebrug 28

Dat betaal ik zelf wel. 5600 Berchem

Belgié

remi mestdagh www.coolblue.be

aartrijkestraat 39

8820 torhout RPR Antwerpen

BTW BE0867686774

Factuurnummer: 1409113302 IBAN BE07310160125666

Klantnummer: 21752541 BIC BBRUBEBB

Factuurdatum: 24 maart 2021

Ordernummer: 52517004

Orderdatum: 24 maart 2021

Artikel Aantal Prijs per stuk BTW Prijs incl. BTW

Tello Drone (powered by DjJI) 1 € 99,00 21% € 99,00

Incl. Recupel: 07.01 Speelgoed - huishoudelijk 1 €0,04

Exclusief BTW € 81,82 Subtotaal € 99,00

BTW 21% € 17,18

—__—. Totaal € 99,00

Totaal € 99,00 "Bancontact" op 24 maart 2021 € 99,00

mos Printen? Niet nodig. Je kunt je factuur altijd terugvinden in je mail of in je Mijn Coolblue-account.

Altijd minimaal 2 jaar garantie. Doen we niet moeilijk over.

B.2 ABBYY

B.2.1 Beste omstandigheden

Figuur B.1: Resultaat ABBYY FineReader



Bibliografie

- Ada, A. (2013a). Convert deskew PDF to editable word document. <http://www.verypdf.com/wordpress/201305/convert-deskew-pdf-to-editable-word-document-36628.html>
- Ada, A. (2013b). Despeckle PDF image and then convert scan PDF to word document. <http://www.verypdf.com/wordpress/201305/despeckle-pdf-image-and-then-convert-scan-pdf-to-word-document-36634.html>
- Akram, Q., Hussain, S., Niazi, A., Anjum, U. & Irfan, F. (2014). Adapting Tesseract for Complex Scripts: An Example for Urdu Nastalique. *2014 11th IAPR International Workshop on Document Analysis Systems (DAS)*, 191–195. <https://doi.org/10.1109/DAS.2014.45>
- Alexandrov, V. (2003). Error evaluation and applicability of OCR systems. <https://doi.org/10.1145/973620.973671>
- Bramer, M. (2020). *Principles of Data Mining*. <https://doi.org/10.1007/978-1-4471-7493-6>
- Che Abdul Rahman, A. N., Ho Abdullah, I., Zainuddin, I. & Jaludin, A. (2019). THE COMPARISONS OF OCR TOOLS. *MALAYSIAN JOURNAL OF COMPUTING*, 4, 335. <https://doi.org/10.24191/mjoc.v4i2.5626>
- Curtis, J. (2010, maart 30). Top OCR Software. <https://web.archive.org/web/20170223213719/http://ocrworld.com/software/5-in-depth/149-top-ocr-software.html>
- Deodhare, D., Suri, N. R. & Amit, R. (2005). Preprocessing and Image Enhancement Algorithms for a Form-based Intelligent Character Recognition System. *IJCSA*, 2(2), 131–144.
- Dinita, M., *, N., says: K. S., says: D. & Says: D. (2021). 7 best OCR software for Windows 10. <https://windowsreport.com/ocr-software-windows-10/>
- El Hadi, A., Ayachi, R., Erritali, M. & Baslam, M. (2018). Improved results of an OCR via NLP using MapReduce framework. *International Journal of Grid and Distributed Computing*, 11, 13–28. <https://doi.org/10.14257/ijgdc.2018.11.11.02>

- Engdahl, S. (2008). Blogs. <https://aws.amazon.com/blogs/machine-learning/deriving-conversational-insights-from-invoices-with-amazon-textract-amazon-comprehend-and-amazon-lex/>
- Farley, P. (2021). *Quickstart: Use the Form Recognizer client library or REST API* (Microsoft, Red.). <https://docs.microsoft.com/en-gb/azure/cognitive-services/form-recognizer/quickstarts/client-library?pivots=programming-language-rest-api&tabs=preview,v2-1#analyze-invoices>
- Gero, J. (1996). *Artificial intelligence in design '96*. Kluwer.
- Ha, H. T., Nevěřilová, Z., Horák, A. e.a. (2018). Recognition of ocr invoice metadata block types. *International Conference on Text, Speech, and Dialogue*, 304–312.
- HarrisHi, A. J. (2020). Improve OCR Accuracy With Advanced Image Preprocessing. <https://docparser.com/blog/improve-ocr-accuracy/>
- Heliński, M., Kmieciak, M. & Parkoła, T. (2012). Report on the comparison of Tesseract and ABBYY FineReader OCR engines.
- Holley, R. (2009). How good can it get? Analysing and improving OCR accuracy in large scale historic newspaper digitisation programs. *D-Lib Magazine: The Magazine of the Digital Library Forum*, 15.
- Kae, A., Huang, G., Doersch, C. & Learned-Miller, E. (2010). Improving state-of-the-art OCR through high-precision document-specific modeling. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1935–1942. <https://doi.org/10.1109/CVPR.2010.5539867>
- Mantas, J. (1986). An overview of character recognition methodologies. *Pattern Recognition*, 19(6), 425–430. [https://doi.org/10.1016/0031-3203\(86\)90040-3](https://doi.org/10.1016/0031-3203(86)90040-3)
- Matei, O., Pop, P. C. & Vălean, H. (2013). Optical character recognition in real environments using neural networks and k-nearest neighbor. *Applied intelligence*, 39(4), 739–748.
- McIntyre, J. (2020). MoSCoW or Kano models - How do you prioritize? <https://www.hotpmo.com/management-models/moscow-kano-prioritize/>
- Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.
- Mori, S., Suen, C. Y. & Yamamoto, K. (1992). Historical review of OCR research and development. *Proceedings of the IEEE*, 80(7), 1029–1058. <https://doi.org/10.1109/5.156468>
- Nagy, G., Nartker, T. & Rice, S. (1999). Optical character recognition: an illustrated guide to the frontier. *Proceedings of SPIE - The International Society for Optical Engineering*, 3967, 58–69. <https://doi.org/10.1117/12.373511>
- Optical Character Recognition (OCR) – How it works. (g.d.). <https://www.nicomsoft.com/optical-character-recognition-ocr-how-it-works/>
- Shah, S. (2017). Object character recognition in C# using Tesseract. *Journal of Electrical Engineering and Technology*, 6.
- Sinha, A. (2002). An Improved Recognition Module for the Identification of Handwritten Digits.
- Smith, R. (2007). An overview of the Tesseract OCR engine. *Ninth international conference on document analysis and recognition (ICDAR 2007)*, 2, 629–633.
- Von Hippel, E. (2001). Learning from open-source software. *MIT Sloan management review*, 42(4), 82–86.

- Why is OCR at 300 dpi a standard? (2017). <https://scansnapcommunity.net/why-is-ocr-at-300-dpi-a-standard-2/>
- XU, T., ZHANG, Y., WU, X. & MING, W. (2021). Intelligent Document Processing.

Woordenlijst

dpi Dots per inch. 11, 22, 30, 33, 36

ML Machine learning. 1, 40, 41, 44, 45, 50, 72

NuGet NuGet (uitgesproken als "New Get") is een pakketbeheerder die is ontworpen om ontwikkelaars in staat te stellen herbruikbare code te delen.. 23, 39, 61, 69

OCR Optical character recognition. 1, 14, 44

PoC Proof of concept. 14, 49, 61