



Faculteit Bedrijf en Organisatie

Intelligente chatbot als begeleidende relatie tussen ontwikkelingsteams en ondersteuningsteam: proof of concept

Hannes Dendoncker

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

Promotor:
Geert Van Boven
Co-promotor:
Jan Delamper

Instelling: Wolters Kluwer

Academiejaar: 2021-2022

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Intelligente chatbot als begeleidende relatie tussen ontwikkelingsteams en ondersteuningsteam: proof of concept

Hannes Dendoncker

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

Promotor:
Geert Van Boven
Co-promotor:
Jan Delamper

Instelling: Wolters Kluwer

Academiejaar: 2021-2022

Tweede examenperiode

Woord vooraf

Artificiële intelligentie leek mij altijd iets zeer toekomstgericht en interessant, maar complex. Al lang wou ik mij erin verdiepen, maar zelfstandig kon ik me er nooit echt aan zetten. Gedurende de laatste twee semesters op HoGent kreeg ik eindelijk de kans om, ondersteund door personen met ruime kennis over het onderwerp, mij te verdiepen in dit prachtig onderdeel van informatica.

Graag dank ik hiervoor mijn lectoren in het eerste semester, vooral de heer Stijn Lievens, om met zoveel interesse en passie mij dit alles bij te leren, en om mijn banale vragen zonder twijfel of minachting te beantwoorden. Graag dank ik Bart Gadeyne, Gilles Vandewiele en Mick Van Den Eeckhout, om mij zo goed te ondersteunen op mijn stageplaats bij Optioryx, en mij te tonen wat er in praktijk allemaal kan gedaan worden met deze technologieën. Als laatste dank ik graag Geert Van Boven en Jan Delamper, mijn promotor en copromotor. Dit is namelijk de eerste maal dat ik een proef als deze schrijf, en beiden toonden erg veel begrip en flexibiliteit. Daarnaast waren ze altijd beschikbaar en behulpzaam wanneer ik hen nodig had.

Samenvatting

Het doel van deze proef is het ontwikkelen van een chatbot, op basis van artificiële intelligentie, die in staat is om gedeeltelijk de taken over te nemen van het ondersteuningsteam. Wolters Kluwer is bedrijf op wereldschaal op vlak van softwareontwikkeling. De software wordt geschreven door vele ontwikkelingsteams, die allemaal de hulp inschakelen van eenzelfde ondersteuningsteam wanneer nodig. Dit is niet in verhouding, en deze proef dient als een mogelijke oplossing voor dit probleem. De bedoeling is dat de chatbot waar mogelijk de taken van het ondersteuningsteam overneemt, en wanneer dit niet mogelijk is toch de taak vereenvoudigt. Binnen deze paper wordt er een eenvoudig prototype van zo een chatbot opgebouwd en beschreven, deze vormt een proof of concept. De proof of concept illustreert welke functionaliteiten de chatbot kan volbrengen binnen de geselecteerde ontwikkelingstools, maar benadrukt ook de beperkingen.

Verder bewijst de proof of concept dat een chatbot weldegelijk in staat is om taken over te nemen van het ondersteuningsteam, zoniet toch deze taken te vereenvoudigen. De bot is met andere woorden in staat om zijn beoogde taak te volbrengen, en kan geïmplementeerd worden als meerwaarde voor Wolters Kluwer.

De proef beperkt zich wel tot een proof of concept, en geen volledige implementatie. Om deze reden wordt er afgesloten met een overzicht aan mogelijke uitbreidingen om de werking van de bot te verbeteren.

Inhoudsopgave

1	Inleiding	13
1.1	Probleemstelling	13
1.2	Onderzoeksvraag	13
1.3	Onderzoeksdoelstelling	14
1.4	Opzet van deze bachelorproef	14
2	Stand van zaken	15
2.0.1	Artificiële intelligentie	15
2.0.2	Machine learning	18
2.0.3	Neurale netwerken	21
2.0.4	Deep learning	21
2.0.5	Natural language processing	23
2.0.6	Chatbot	23
2.0.7	Python	24

2.0.8	Proof of concept	24
2.0.9	Knowledge base	25
2.0.10	Wolters Kluwer	25
2.0.11	Frameworks	26
2.0.12	Gebruikte software binnen Wolters Kluwer	28
2.0.13	Bot Framework Emulator	29
2.0.14	QnA Maker	29
2.0.15	Verschillen tussen QnA Maker en LUIS	30
2.0.16	Bot Framework Composer	30

3 Methodologie 33

3.0.1	Onderzoek naar optimale manier van opbouwen van de chatbot	33
3.0.2	Opbouwen van de chatbot deel 1: echoBot	34
3.0.3	Opbouwen van de chatbot deel 2: opzet en implementatie LUIS	35
3.0.4	Opbouwen van de chatbot deel 3: aanvullen met QnA Maker	37
3.0.5	Opbouwen van de chatbot deel 4: integratie Microsoft Teams	38

4 Proof of concept 39

4.0.1	Context	39
4.0.2	Functionaliteiten	39
4.0.3	Mogelijke uitbreidingen	44

5 Conclusie 47

A Onderzoeksvoorstel 49

A.1	Introductie	49
A.2	State-of-the-art	49

A.3	Methodologie	50
A.4	Verwachte resultaten	51
A.5	Verwachte conclusies	51
	Bibliografie	53

Lijst van figuren

2.1	Turing test, gecreëerd door Alan Turing.	17
2.2	Voorbeeld van een gelabelde dataset, met kenmerken en label.	19
2.3	Voorstelling van een neurale netwerk.	22
2.4	Diagram ter illustratie van hoe de verschillende technologieën zich tegenover elkaar verhouden.	22
2.5	Vergelijkende tabel rond R, Python en Julia.	25
2.6	Verduidelijking antwoord LUIS, eigen afbeelding	27
2.7	Voorbeeldzinnen gelabeld met entiteiten, eigen afbeelding	28
2.8	Fragment uit voorgeprogrammeerde knowledge base van QnA maker. Links de vraag, rechts het antwoord. Eigen afbeelding	30
2.9	Grafische interface van Bot Framework Composer.	31
3.1	Voorbeeld van een echogesprek met de bot, eigen afbeelding .	34
3.2	Lijst van triggers binnen de WoltersKluwerBot, eigen afbeelding ..	35
3.3	Verloop van de echodialoog, eigen afbeelding	36
3.4	Voorbeeld van een zelf getrainde vraag en antwoord, eigen afbeelding	37
4.1	Groet van de bot, eigen afbeelding	40

4.2	Opvragen extra informatie aan de hand van twee verschillende commando's, eigen afbeelding	41
4.3	Dialog na herkennen 'ticket' commando, eigen afbeelding	41
4.4	Overzicht van een geregistreerd probleem, eigen afbeelding	42
4.5	Opvragen component aan de hand van een selectlist, eigen afbeelding	43
4.6	Overzicht van een infrastructuur aanvraag, eigen afbeelding	43
4.7	Opvragen van contactgegevens met behulp van QnA Maker, eigen afbeelding	44
4.8	Voorbeeld mogelijkheden LUIS, eigen afbeelding	46

1. Inleiding

1.1 Probleemstelling

Deze proef werd opgesteld op vraag van Wolters Kluwer. Het bedrijf specialiseert zich in het leveren professionele informatie, softwareoplossingen en diensten. Sinds de oprichting in 1836 is het erg gegroeid tot een bedrijf op wereldschaal.

Een bedrijf van dit formaat wordt opgedeeld in verschillende onderdelen. Binnen deze thesis wordt er gefocust op de softwareontwikkelingsteams, het ondersteuningsteam dat voor hen beschikbaar gesteld wordt, maar vooral de relatie tussen deze twee partijen.

Momenteel werkt deze relatie als volgt. Wanneer een lid van een van de ontwikkelings-teams hulp nodig heeft, dit gaat van het oplossen van een probleem tot het bestellen van hardware, contacteert hij of zij het ondersteuningsteam. Een lid van dit team helpt hem of haar dan verder tot ze samen tot een oplossing komen. Wat dit moeilijk maakt, is dat het ondersteuningsteam erg klein is in verhouding met de verschillende ontwikkelingsteams.

1.2 Onderzoeksvraag

Het doel van deze thesis is het komen tot een efficiëntere, meer gestroomlijnde relatie tussen beide teams. Dit wordt onderzocht aan de hand van een chatbot, die onder andere gebruik maakt van artificiële intelligentie. De proef beperkt zich tot een proof of concept, een beperkte voorbeeldimplementatie, maar toch kunnen er enkele vragen gesteld worden als richtlijnen voor het bereiken van het doel. In hoeverre kan een chatbot de taken van het ondersteuningsteam overnemen? Verder, hoe kunnen taken die niet overgenomen kunnen worden, toch vereenvoudigd worden voor het ondersteuningsteam? Op welke manier kan het gebruiksgemak van de chatbot gemaximaliseerd worden, opdat het gebruik ervan

gemotiveerd wordt?

1.3 Onderzoeksdoelstelling

Zoals hiervoor beschreven is het uiteindelijke doel van deze proef het stroomlijnen en assisteren van de relatie tussen de ontwikkelingsteams en het ondersteuningsteam, aan de hand van een chatbot. Een eenvoudige versie van deze chatbot wordt ook geproduceerd, als proof of concept. De bedoeling is dus dat de chatbot de taken van het ondersteuningsteam overneemt of vereenvoudigt. Concreet moet de bot ten eerste in staat zijn om eenvoudige vragen te beantwoorden. Een voorbeeld hiervan is het opvragen van contactinformatie van een collega. Daarnaast bij meer complexe vragen of taken die de bot niet zelfstandig kan voltooien, is het toch de bedoeling dat de bot de taak vereenvoudigt voor het ondersteuningsteam, door bijvoorbeeld al alle benodigde informatie te verzamelen. Wanneer de bot bij voorbeeld niet in staat is om een probleem zelfstandig op te lossen, is het de bedoeling dat de bot alle informatie rond dit probleem verzamelt, en deze dan opslaat in de vorm van een ticket. Dit ticket kan dan gecommuniceerd worden naar het ondersteuningsteam. Een tweede voorbeeld van een complex probleem is het aanvragen van infrastructuur of hardware. Ook hier is het de bedoeling dat de bot alle nodige informatie verzamelt, om deze dan in een gestructureerde vorm door te geven aan het ondersteuningsteam.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 4 wordt de functionaliteit en de mogelijkheden van een chatbot geïllustreerd aan de hand van een beperkte implementatie.

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

2. Stand van zaken

Zoals in de inleiding vermeld wordt er in deze bachelorproef geprobeerd om een chatbot op te bouwen op basis van artificiële intelligentie, om zo de relatie tussen de ontwikkelingsteams en het ondersteuningsteam te begeleiden.

Het opbouwen van een chatbot is complex. Er wordt gebruik gemaakt van verschillende concepten, ontwikkelingstools en andere hulpmiddelen. Vooraleer er verdergegaan wordt, is het van groot belang van deze termen te verduidelijken. Daarnaast verklaart deze literatuurstudie de verschillende keuzes gemaakt doorheen deze proef.

2.0.1 Artificiële intelligentie

Introductie

Artificiële intelligentie is het eerste concept dat besproken wordt. Het is een van de nieuwe technologieën die volgde in de decennia na wereldoorlog twee. Wel is de definitie, de manier van implementatie en het gebruik ervan ondertussen erg veranderd. Als eerste wordt er een korte geschiedenis besproken. Vervolgens wordt een actuele stand van zaken gegeven en geïllustreerd aan de hand van enkele huidige implementaties. Ten slotte beschrijft dit onderdeel enkele moeilijkheden waar de technologie mee kampt.

Het idee

Het concept onderliggend aan artificiële intelligentie bestaat al veel langer dan de technologie zelf. De eerste duidelijke omschrijvingen van artificiële intelligentie doen zich voor in werken van fictie. Hierin wordt het vaak voorgesteld in de vorm van een robot die menselijk oogt en denkt, maar toch geen echt persoon is. De meest populaire voorbeelden

zijn Jules Verne in de negentiende eeuw en Isaac Asimov in de twintigste eeuw, maar ook al vroeger zijn er enkele voorbeelden te vinden. Zo schreef L. Frank Baum, de auteur van het boek 'The Wizard of Oz', ook enkele boeken over een mechanische man genaamd 'Tiktok'. Hij omschreef deze als een 'mechanische man met snelle reflexen, die denkt, praat, zich gedraagt en alles doet, op leven na'. Vermoedelijk was dit een belangrijke bron van inspiratie voor de eerste computerwetenschappers die onderzoek deden in dit vakgebied. Een belangrijk detail hier is dat men er altijd van uit ging dat intelligentie gecreëerd werd, en niet zelfstandig kon ontstaan, dit is namelijk in de lijn met het christelijke geloof. Pogingen tot creatie van artificiële intelligentie in deze periode volgden een drogere en meer mechanische aanpak, net als het ontwerpen van bij voorbeeld een pomp of een wiel. In de eerste plaats onderzoeken hoe iets werkt, en dan met deze kennis het proberen namaken.

Darwin veranderde deze filosofie in 1859 met zijn evolutietheorie. Als biologische individuen zelfstandig, hiermee bedoeld zonder 'maker', zich kunnen verbeteren en evolueren, kon dit ook op technologisch vlak mogelijk zijn (Buchanan, 2006).

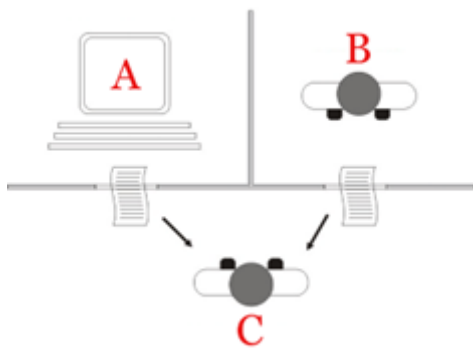
Eerste pogingen

Vooraleer er een eerste voorbeeld van concrete artificiële intelligentie kan gegeven worden, moet er bepaald worden waar de grens ligt, namelijk, wanneer is een computer intelligent. Alan Turing beschreef in 1950 een eerste definitie hiervan. Dit aan de hand van een test op die later naar hem vernoemd werd, namelijk de Turing Test. De test gaat, weliswaar erg vereenvoudigd, als volgt. Er zijn drie deelnemers, een ondervrager, een vrijwilliger en de computer wiens intelligentie getest wordt. De ondervrager stelt enkele vragen, zonder de andere deelnemers te kunnen waarnemen, en ontvangt de antwoorden van de beide deelnemers. Als de ondervrager geen onderscheid kan maken tussen de vrijwilliger en de computer, is de computer intelligent.

Deze test bleek uiteindelijk geen goede, volledige definitie te bieden (French, 1990). Als reactie hierop veranderde de definitie van artificiële intelligentie de daaropvolgende jaren sterk. Bij het ontstaan van de technologie als computerwetenschap, in het jaar 1956, had men niet enkel een andere definitie van de term, maar ook een andere aanpak en filosofie om deze technologie te verwezenlijken. In deze periode omschrijft de term eerder een technologie die menselijke taken vervangt, maar daarom niet menselijk handelt of denkt (Spector, 2006).

Evolutie tot heden

Alhoewel nog geen enkele computer de Turing Test heeft kunnen slagen, zijn er wel al vele grote vorderingen gemaakt sinds de eerste definities van artificiële intelligentie. Tegenwoordig wordt artificiële intelligentie niet enkel meer omschreven als een technologie die menselijke taken vervangt, maar eerder als een technologie die een echt persoon, menselijk contact simuleert. Ookal vallen deze computers nog altijd te onderscheiden van een echt persoon, toch blijkt dat dit steeds moeilijker wordt (Hill & Ford, 2015). Ook kunnen deze computers even behulpzaam zijn als een echt persoon (Luo & Tong, 2019). Als laatste nog een illustratie van hoe de ideeën van Darwin tot op de dag van vandaag artificiële intelligentie inspireren. Eén van de vele vaak voorkomende algoritmen die



Figuur 2.1: Turing test, gecreëerd door Alan Turing.

https://stringfixer.com/nl/Turing_Test

tegenwoordig gebruikt worden is het Genetisch Algoritme. Dit algoritme wordt gebruikt voor optimalisatie, en neemt principes zoals cross-overs en mutaties over uit de biologie (Holland, 1992).

Toepassingen artificiële intelligentie

Een belangrijke sector waar men meer en meer deze nieuwe technologieën probeert toe te passen is de medische sector. Het kan er ingezet worden voor het besturen van robotica, het bepalen van een diagnose, het opstellen en raadplegen van medische statistieken en zelfs ons begrip van de menselijke biologie verbeteren (Hamlet & Tremblay, 2017). Een andere toepassing is binnen de financiële sector. Tegenwoordig zijn vele financiële applicaties non-lineair en onvoorspelbaar. Artificiële intelligentie biedt hiervoor meer accurate oplossingen dan klassieke systemen (Bahrammizadeh, 2010). Ook binnen de transportsector speelt deze technologie een rol, vooral op vlak van optimalisatie. Zo wordt het gebruikt om bij voorbeeld de ideale route, ideale stapeling of ideale dozengroottes te berekenen. Hierdoor wordt de transport- en verpakkingskost geminimaliseerd, wat een groot voordeel biedt in een periode van grote fluctuaties in brandstofkosten (Abduljabbar e.a., 2019). Zowat in alle sectoren waar men veel data produceert of gebruikt, kan artificiële intelligentie hulp bieden.

De tot nu toe benoemde voorbeelden beperken zich tot de verouderde definitie van artificiële intelligentie. De techniek wordt gebruikt om menselijke taken over te nemen, maar niet om menselijk gedrag te simuleren. Dit laatste voorbeeld slaagt daar wel in. Een groep onderzoekers deed in 2019 onderzoek naar een autonome robot die ingezet werd als emotionele ondersteuning voor kinderen. De robot slaagde erin om na meerdere sessies met elk kind zich zelfstandig aan te passen van kind tot kind, en zo de hen in een positieve stemming te brengen en behouden (Gamborino e.a., 2020).

Moeilijkheden

Zoals bij elke andere revolutionaire technologie zijn er enkele twijfels of moeilijkheden. De belangrijkste twee binnen artificiële intelligentie zijn onzekerheid en onverklaarbaarheid. Onzekerheid houdt in dat wanneer een applicatie die gebruik maakt van deze technologie

een resultaat teruggeeft, het niet makkelijk is om in te schatten hoe accuraat dit is. De applicatie kan wel een schatting geven van hoe accuraat hij denkt te zijn, maar dit is geen concreet of exact oordeel.

De tweede moeilijkheid, onverklaarbaarheid, is hier sterk aan gelinkt. Aangezien het systeem zichzelf opbouwt en traint, is het zeer moeizaam of zelfs bijna onmogelijk te achterhalen hoe er tot een bepaald resultaat gekomen wordt. Vaak wordt dit probleem omschreven als een ‘black box’, de input en output is geweten, maar wat hen verbindt is onbekend. De twee hiervoor benoemde hindernissen zorgen voor twijfels bij implementatie van artificiële intelligentie, zeker binnen sectoren waar zekerheid en verklaarbaarheid een belangrijke rol spelen, zoals in een ziekenhuis (Holzinger e.a., 2019).

2.0.2 Machine learning

Introductie

Machine Learning is een onderdeel van artificiële intelligentie. Deze term omschrijft ook een technologie, of beter algoritme, die menselijke taken vervangt of zelfs menselijke handelingen simuleert. Dit zonder expliciet geprogrammeerd te worden. Het leert met andere woorden zelf hoe deze taak best te volbrengen. Het leerproces gebeurt wel niet in één enkele stap. Vaak is dit een iteratief proces, en wordt het algoritme stap voor stap verbeterd. Dit leren en verbeteren gebeurt aan de hand van data.

Deze vorm van artificiële intelligentie wordt tegenwoordig al breed toegepast. Zo wordt machine learning onder andere gebruikt om zoekresultaten en aanbevelingen te verbeteren.

Verskillende vormen van machine learning

Binnen machine learning bestaan er drie basisvormen, namelijk supervised learning, unsupervised learning en reinforcement learning. Dit zijn niet de enige vormen, er bestaan namelijk tussenvarianten zoals semi-supervised learning, maar deze worden hier niet besproken.

Supervised learning, unsupervised learning en reinforcement learning, alle drie onderdeel zijnde van machine learning, delen vele eigenschappen. In alle vormen is het de bedoeling om aan de hand van data een bepaald doel te bereiken. Dit doel verschilt sterk van vorm tot vorm, en kan bijvoorbeeld het groeperen van de data zijn, of het maximaliseren van een bepaald objectief, of het correct labelen op basis van kenmerken uit de data. Het selecteren van de correcte vorm is dus van groot belang, en is afhankelijk van wat het doel van het algoritme precies is.


Samengevat zijn er dus onderlinge gelijkenissen, maar heeft elke vorm zijn eigen kenmerken en functies. Om wat meer context te geven en de keuzes binnen deze paper te verduidelijken, worden alle drie de vormen hieronder besproken.

Supervised learning

Als eerste wordt supervised learning besproken. Supervised learning maakt gebruik van een gelabelde dataset. Hiermee wordt bedoeld dat de dataset een lijst van kenmerken

bevat, en bij deze lijst een bijhorend label. Zie eventueel onderstaande afbeelding ter verduidelijking. Het labelen van deze dataset vereist menselijke interactie, vandaar de naam supervised learning, of leren onder toezicht. Het algoritme bouwt een model op door relaties te leggen tussen de lijst van kenmerken en de bijhorende labels. Eenmaal getraind, is het model in staat om wanneer men het een niet-gelabelde set van kenmerken geeft, een vermoedelijk label terug te geven. Dit gebeurt aan de hand van de relaties die het algoritme ontdekte gedurende de trainingsfase. Dit is de techniek die in deze bachelorproef gebruikt en besproken wordt.

Ook binnen supervised learning, als onderdeel van artificiële intelligentie, blijven onzekerheid en onverklaarbaarheid een probleem. Wel is dit relatief beperkt in vergelijking met unsupervised learning, omdat er een concrete input en output is. Deze waarden zijn namelijk onder controle van de persoon die het algoritme overziet. De input is een eigen waarde die men zelf beslist, en de output beperkt zich tot de waarden die in de trainingsdata voorkwamen.



Position	Experience	Skill	Country	City	Salary (\$)
Developer	0	1	USA	New York	103100
Developer	1	1	USA	New York	104900
Developer	2	1	USA	New York	106800
Developer	3	1	USA	New York	108700
Developer	4	1	USA	New York	110400
Developer	5	1	USA	New York	112300
Developer	6	1	USA	New York	114200
Developer	7	1	USA	New York	116100
Developer	8	1	USA	New York	117800
Developer	9	1	USA	New York	119700
Developer	10	1	USA	New York	121600

Figuur 2.2: Voorbeeld van een gelabelde dataset, met kenmerken en label.

<https://d1zx6djv3kb1v7.cloudfront.net/wp-content/media/2019/09/Features-and-Labels-in-a-Dataset-i2tutorials.png>

Unsupervised learning

De tweede techniek, unsupervised learning, gebruikt een niet-gelabelde dataset. Deze niet-gelabelde dataset als inputwaarde is in tegenstelling tot supervised learning, waar er een gelabelde dataset gebruikt wordt. Dit is een belangrijk verschil tussen deze twee vormen van machine learning. Het doel en gebruik van deze techniek verschilt dan ook van de hiervoor besproken techniek. Aan de hand van unsupervised learning probeert het algoritme patronen en trends te herkennen in de niet-gelabelde dataset. Op basis van deze resultaten, kan de dataset onder andere gegroepeerd worden, geanalyseerd worden, of eenvoudiger voorgesteld worden. Deze vorm van Machine Learning vereist minder menselijke interactie. Wanneer men data groepeer aan de hand van unsupervised learning, bij voorbeeld, hoeft men enkel aan te geven hoeveel van deze groepen er gecreëerd worden. Er zijn andere parameters die ingesteld kunnen worden, maar deze zijn niet essentieel. Het is met andere woorden eenvoudiger in gebruik, maar niet geschikt voor dezelfde toepassingen als supervised learning.

Onzekerheid en onverklaarbaarheid vormen hier een groter probleem. Alhoewel men controle heeft over de inputwaarden, zijn de outputwaarden onvoorspelbaar. Het is mogelijk om te achterhalen op basis van welke waarden de output gegenereerd is, maar dit verklaart niet alles.

Reinforcement learning

Deze laatste techniek verschilt sterk van de vorige twee en is dus een buitenstaander. Het doel hier is om de totale cumulatieve beloning te maximaliseren. Dit klinkt complex maar is eenvoudig te verduidelijken aan de hand van een voorbeeld. Reinforcement learning kan gebruikt worden om een algoritme te trainen een videospel, neem nu pacman, te spelen. Des te meer het algoritme erin slaagt om het spel correct te spelen, des te hoger de score die het spel aan die poging zal toereiken. Als het algoritme er dus in slaagt om veel te eten en niet dood te gaan, zal dit dus ook beloond worden aan de hand van een hogere score. De input data van dit soort algoritme is ook erg beperkt. Er moet enkel een beginpositie, omgeving en objectieffunctie voorzien worden, het algoritme doet verder alles zelfstandig. Reinforcement learning is gebaseerd op het 'trial and error'- principe, het algoritme leert op basis van vorige ervaringen. Deze ervaringen kunnen zowel positief als negatief zijn, en dus een positieve of negatieve invloed op de score hebben. Bij een volgende poging houdt het algoritme hier dus rekening mee, en is het dus beter in het uitvoeren van zijn taak.

De problemen rond onzekerheid en onverklaarbaarheid zijn hier beperkt. De objectief- of scorefunctie wordt opgesteld door de persoon die het algoritme traint. Wanneer deze functie correct opgesteld is, weerspiegelt deze score de resultaten van het algoritme. Een lage score verklaart een slechte poging van het algoritme, een hogere score een betere poging. Daarnaast is het proces iteratief. Met de juiste tools kan elke poging grafisch voorgesteld worden, en kan de eigenaar van het algoritme dus zien hoe de verschillende pogingen precies verlopen.

Overzicht

Kort samengevat verschillen deze drie basisvormen van machine learning sterk in zowel input- als outputwaarden, en bovendien ook in functie. Supervised learning gebruikt een gelabelde dataset om te trainen, en wordt na training ingezet om aan een niet-gelabelde dataset labels toe te kennen. Bij unsupervised learning wordt er gebruik gemaakt van een niet-gelabelde dataset, om diezelfde dataset te analyseren, groeperen of eenvoudiger voor te stellen. Als laatste werkt reinforcement learning niet met een echte dataset zoals de vorige twee. Slechts startpositie, omgevingsinformatie en een objectieffunctie moeten voorzien worden. Het doel van deze techniek is dan om die objectieffunctie te maximaliseren, aan de hand van trial and error.

Door de nood aan een gelabelde dataset vereist supervised learning algoritme de meeste manuren. Zowel unsupervised learning en reinforcement learning zijn eenvoudiger in gebruik. Ten slotte kampen alle drie de vormen van machine learning met een niveau van onzekerheid en onverklaarbaarheid. Binnen reinforcement learning vormt dit geen groot probleem, maar binnen supervised learning en vooral unsupervised learning wel (Mahesh, 2020).

Binnen deze paper wordt er gebruik gemaakt van supervised learning. Deze keuze is vooral

gemaakt op basis van de functies van deze techniek, aangezien het de bedoeling is om niet-gelabelde data te kunnen labelen.

2.0.3 Neurale netwerken

Neurale netwerken vormen de basis van deep learning, en zijn op hun beurt een onderdeel van machine learning, specifiek supervised learning. Geïnspireerd door de werking van de menselijke hersenen, bestaat ze uit een complex netwerk van nodes, analoog aan neuronen. Deze nodes zijn onderling verbonden via relaties. Wanneer een neuraal netwerk getraind wordt, worden de sterktes van de relaties tussen deze nodes geconfigureerd. Dit staat in contrast met een klassieke vorm van supervised learning, waarbij deze relaties rechtstreeks tussen de inputdata en het label liggen. Hier loopt die relatie dus via tussenelementen, de hiervoor benoemde nodes. De laatste van deze nodes is dan gelinkt aan de output layer, die de labels voorstelt.

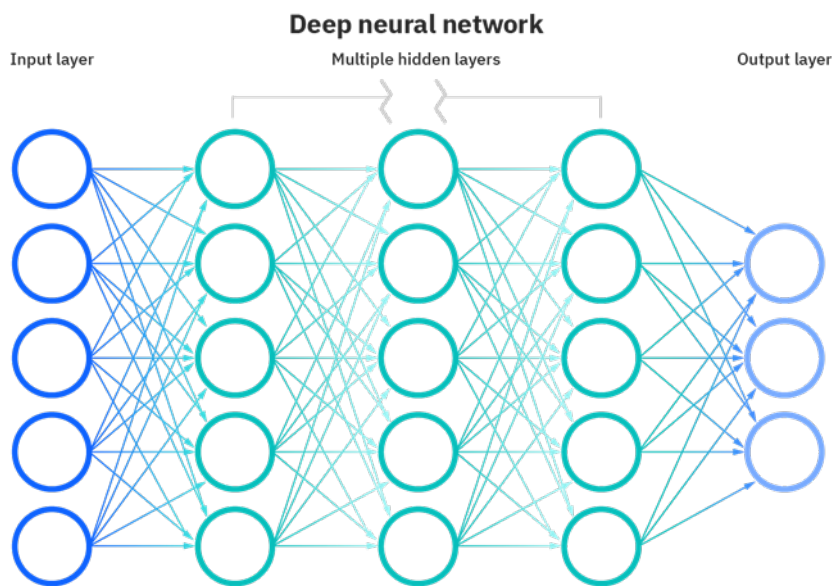
Bij training passeert input data door het netwerk van links naar rechts, maar ook van rechts naar links. Dit alles om de relatiesterktes optimaal in te stellen, op basis van de trainingsdata. Eenmaal getraind, spreken we over een model.

Wanneer er dan aan dit model inputwaarden gegeven worden, herleid het model deze waarden via de nodes en hun relaties tot een output. Hierbij wordt er rekening gehouden met welke relaties er bestaan tussen de nodes, maar ook hoe sterk deze zijn na de training. Het grote verschil tussen een neuraal netwerk en een meer eenvoudige vorm van supervised learning is dat een neuraal netwerk uit meerdere lagen bestaat, minstens drie. Minimaal is er één inputlaag, één hidden layer of tussenlaag, en één outputlaag. Het aantal input en outputlagen ligt vast, maar het aantal tussenlagen kan variëren. De inputwaarden of kenmerken worden dus niet direct gelinkt aan een bepaalde output, maar hun relatie met de outputwaarden loopt via een netwerk van nodes met elk hun eigen relaties, en relatiesterktes.

2.0.4 Deep learning

Deep learning is ook een subset van machine learning, en maakt gebruik van neurale netwerken. Wanneer er een neuraal netwerk bestaande uit meer dan drie lagen ingeschakeld wordt, spreekt men over deep learning. Gebruik makend van complexe neurale netwerken, bevat deep learning meerdere niveaus van abstractie. Hiermee wordt bedoeld dat het algoritme bestaat uit meerdere lagen, en zowel voorwaarts als achterwaarts kan optimaliseren en leren. Deze kenmerken worden overgeërfd van de neurale netwerken die er onderdeel van vormen. Hierdoor kan een deep learning algoritme een ingewikkelde structuur binnen een grote dataset herkennen. Enkele voorbeelden hiervan zijn spraakherkenning, voorwerpdetectie en het 'begrijpen' van taal en de connotaties ervan. Deze principes kunnen dan op hun beurt gebruikt worden voor bredere technologieën zoals zelfrijdende auto's (LeCun e.a., 2015).

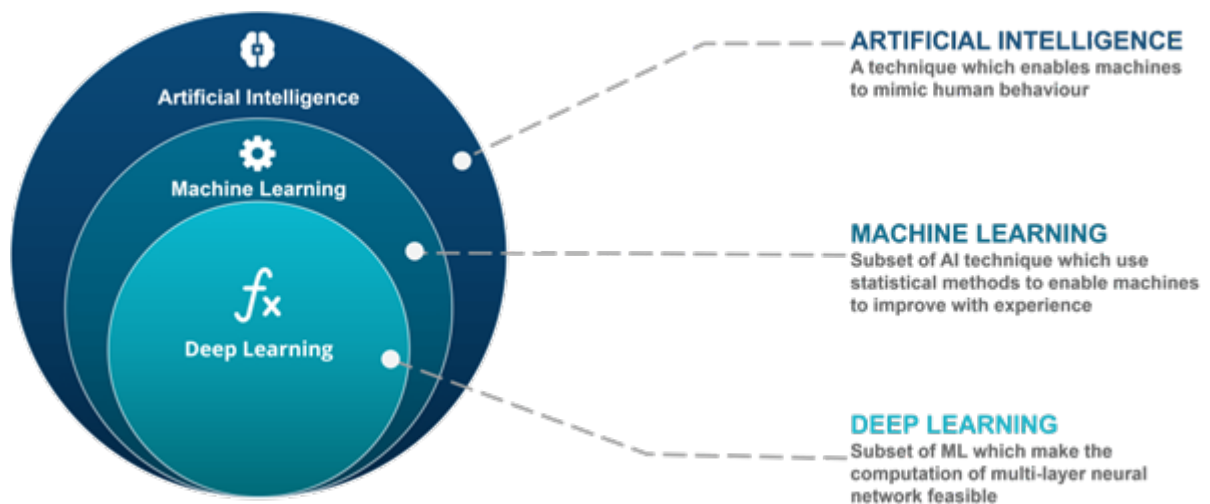
Een andere eigenschap die deep learning overneemt van neurale netwerken, maar eigenlijk van artificiële intelligentie als basis, is opnieuw onzekerheid en onverklaarbaarheid. Als meest complexe vorm van artificiële intelligentie besproken in deze paper, kampt het ook het sterkst met deze problemen. Een deep learning model is als het ware een 'black box' is.



Figuur 2.3: Voorstelling van een neurale netwerk.

https://1.cms.s81c.com/sites/default/files/2021-01-06/ICLH_Diagram_Batch_01_03-DeepNeuralNetwork-WHITEBG.png

De input en output van het algoritme is gekend en duidelijk, maar hoe deze intern gelinkt worden is uiterst moeilijk te achterhalen. Elke laag, elke node en elke relatie heeft een invloed op de output. De grootte van deze invloeden kan achterhaald worden, maar zelfs dit kan een output niet verklaren.



Figuur 2.4: Diagram ter illustratie van hoe de verschillende technologieën zich tegenover elkaar verhouden.

<https://www.edureka.co/blog/ai-vs-machine-learning-vs-deep-learning/>

2.0.5 Natural language processing

NLP, voluit natural language processing, is dan weer een onderdeel van deep learning. Het doel van deze technologie is het extraheren van informatie uit een tekst. Dit gaat niet enkel over inhoud, maar ook onder andere connotatie en geïmpliceerde informatie. De eerste versies van deze technologie werden opgesteld met als doel het vertalen van een Engelstalige tekst naar Russisch. Dit was niet succesvol, omdat het homografen, eenzelfde woord met meerdere betekenissen, niet kon verwerken (Nadkarni e.a., 2005). Het correct interpreteren van de vragen en antwoorden van een eindgebruiker aan de chatbot (zie verder) is van groot belang voor het slagen van deze proef (Chowdhary, 2020).

Als onderdeel van deep learning bestaat natural language processing ook uit een algoritme dat gebruikt wordt om een model op te bouwen. Het opbouwen van dit model bestaat uit twee grote stappen. Als eerste moet de inputdata opgeschoond worden tot een formaat dat eenvoudig te interpreteren en verwerken is door een algoritme. Dit houdt onder andere in maar is niet beperkt tot het opdelen van tekst in kleinere stukken, het verwijderen van stopwoorden, het reduceren van woorden tot hun grondvorm en het classificeren van woorden in hun bijhorende categorie. Enkele voorbeelden van deze categorieën zijn zelfstandig naamwoord, bijvoeglijk naamwoord en werkwoord.

Eenmaal de data verwerkt is, kan het model getraind worden. Zoals bij elke vorm van supervised learning wordt er gebruik gemaakt van een gelabelde dataset. Deze werd opgesteld in de vorige stap. Na uitvoerig trainen is het model compleet, en kan het gebruikt worden om voorspellingen te doen. Binnen een natural language processing model kunnen deze voorspellingen inhouden wat de bedoeling of ‘intent’ is van een inputzin, maar ook welke objecten of entiteiten erin voorkomen. Een populair voorbeeld van zo een getraind natural language processing model is LUIS, dit wordt verder besproken.

2.0.6 Chatbot

Wat is een chatbot

Een chatbot is een applicatie waarmee men op een slimme manier kan communiceren met een computer. Normaal communiceert een computer aan de hand van bits en bytes, klassen en objecten, maar hier schikt hij zich naar een menselijke vorm van communicatie, namelijk taal. Dit is niet evident aangezien computers hiervoor niet ontworpen zijn, en wordt dus onder andere mogelijk gemaakt door een chatbot. Eenvoudig voorgesteld verricht een chatbot twee taken. Als eerste is het in staat om een bericht naar de gebruiker te sturen, in vorm van een vraag of antwoord. Daarnaast begrijpt een chatbot de input van de gebruiker, ook dit kan een vraag of antwoord zijn.

De boodschap van de gebruiker correct interpreteren is niet evident, en hierbij wordt er gebruik gemaakt van natural language processing. Deze technologie stelt de chatbot in staat om te begrijpen wat de gebruiker wil of bedoelt, met een bepaalde boodschap of bericht. Naast de bedoeling van de gebruiker, kan deze techniek ook objecten of entiteiten uit het bericht halen. Deze worden dan teruggegeven aan de chatbot, die die informatie kan gebruiken om de intentie van de gebruiker te volbrengen. Als er nog informatie ontbreekt, stelt de chatbot verdere vragen.

Deze chatbots zijn populair in gebruik. Op websites zijn ze vaak terug te vinden als

assistent, en daarnaast zijn ze ook terug te vinden als persoonlijke assistent op vele smartphones.

Moeilijkheden

Zoals hierboven vermeld gebruikt een chatbot natural language processing om op een correcte manier de input van een gebruiker te verwerken. In praktijk wordt het gebruik van deze technologie beperkt. Een chatbot is namelijk ook niet ontworpen om een informele conversatie mee te voeren, maar eerder om te helpen met specifieke problemen. De training van het natural language processing model wordt dus beperkt tot het interpreteren van deze specifieke problemen. Wanneer de bot niet in staat is om zo een probleem te herkennen, wordt er een vraag van de bot uit voorzien om te herformuleren. Op deze manier wordt de functionaliteit beperkt, en worden er geen middelen onnodig verspild.

Het vinden van een juiste balans tussen de intelligentie van een chatbot, en het beperken van de complexiteit is met andere woorden van hoog belang (Buchanan, 2006).

2.0.7 Python

Trager maar goed ondersteund en betrouwbaar

Een mogelijke optie is het opbouwen van de bot is om deze zelf te schrijven in een taal als Python.

Dit is een van de meer eenvoudige programmeertalen, en ideaal voor ontwikkeling. Op vlak van efficiëntie is het beter om dit op lange termijn te vervangen door een taal als C, maar dat valt buiten de scope van deze paper. Naast eenvoud heeft Python ook enkele andere voordelen. Zo is dit de taal met het meeste open-source libraries, vooral op vlak van Machine Learning. Ten slotte zijn de frameworks (zie verder) die gebruikt worden ontworpen om met Python te werken.

Vermeldenswaardige alternatieven

Twee opties die uiteindelijk minder geschikt bleken zijn R en Julia. R is te eenvoudig, en moeilijker compatibel met de gebruikte frameworks.

Julia is een nieuwe, veelbelovende taal. Alhoewel deze taal sneller is dan Python, heeft het gebruik ervan toch enkele risico's. Het is namelijk recenter, en heeft het haar stabiliteit nog niet bewezen op een niveau zoals Python. Ook deze taal is niet standaard compatibel met de gebruikte frameworks (Müller & Guido, 2018).

2.0.8 Proof of concept

Het uiteindelijke doel van deze proef is het produceren van een 'proof of concept'. Dit houdt in dat er een werkende versie van de chatbot opgeleverd wordt, ter ondersteuning van de theorie. Het is als het ware een illustratie van hoe de technologie kan toegepast worden. Wel gaat het hier over een beperkte implementatie. Het is niet de bedoeling om

		
<ul style="list-style-type: none"> ✓ Data visualization ✓ Shiny & RStudio Products 	<ul style="list-style-type: none"> ✓ Natural Language Processing ✓ Image recognition ✓ Data visualization 	<ul style="list-style-type: none"> ✓ Fast execution ✓ Resource- and runtime-intensive apps

Figuur 2.5: Vergelijkende tabel rond R, Python en Julia.

<https://www.r-bloggers.com/2020/12/r-python-julia-in-data-science-a-comparison/>

deze al volledig uit te werken, het dient enkel ter bewijs van de mogelijkheden van de technologie.

2.0.9 Knowledge base

Een knowledge base is een online verzameling van informatie, beschikbaar voor gebruikers. In het algemeen bevat een bepaalde knowledge base informatie rond eenzelfde onderwerp. Deze verzameling van informatie kan zowel gestructureerd als ongestructureerd zijn. Een knowledge base kan bijvoorbeeld een lijst van problemen met bijhorende oplossingen bevatten. Een ander voorbeeld van implementatie is als opslag voor contactgegevens. In dit voorbeeld bevat de knowledge base dan een lijst van contactpersonen, met bijhorende contactgegevens.

2.0.10 Wolters Kluwer

Wolters Kluwer is het bedrijf waarvoor dit onderzoek gedaan wordt. Het is een uitgeverij op wereldschaal, niet enkel fysiek vlak, zoals boeken, maar ook op digitaal vlak. Zo produceert het bedrijf ook software voor professionelen en bedrijven. Enkele sectoren waarvoor ze deze producten uitbrengen zijn de medische sector, fiscaliteit en accounting, en de juridische sector. Uiteraard zijn er meerdere teams aanwezig om deze software te ontwerpen, op te bouwen en te onderhouden. Deze worden vanaf hier de ontwikkelingsteams genoemd. Indien een individu binnen een van de ontwikkelingsteams een technisch probleem heeft dat niet zelfstandig kan opgelost worden, contacteert deze persoon het ondersteuningsteam. Dit team staat in om hun vragen te beantwoorden en hun problemen op te lossen. Zij doen dit op basis van hun eigen kennis, maar maken ook gebruik van een ruime, bestaande knowledge base met oplossingen. Naast problemen oplossen vervult het ondersteuningsteam ook andere taken. Zo staan ze ook onder andere in voor het registreren en bestellen van nieuwe hardware.

2.0.11 Frameworks

Definitie

Een framework, of in het Nederlands raamwerk, is een verzameling van softwareonderdelen die gebruikt kan worden om andere software op te bouwen. Een framework kan geïmporteerd worden binnen de compatibele softwareontwikkelingsomgevingen, en daarbinnen dan aangesproken worden. Het is een manier om externe code te importeren en gebruiken binnen een eigen applicatie.

LUIS - introductie

De tool die Microsoft aanbiedt om op een eenvoudige en toegankelijke manier gebruikt te maken van natural language understanding heet Language Understanding (LUIS). Het is op voorhand al getraind met ruime datasets, en kan nog afgericht worden met eigen data.

LUIS - gebruik

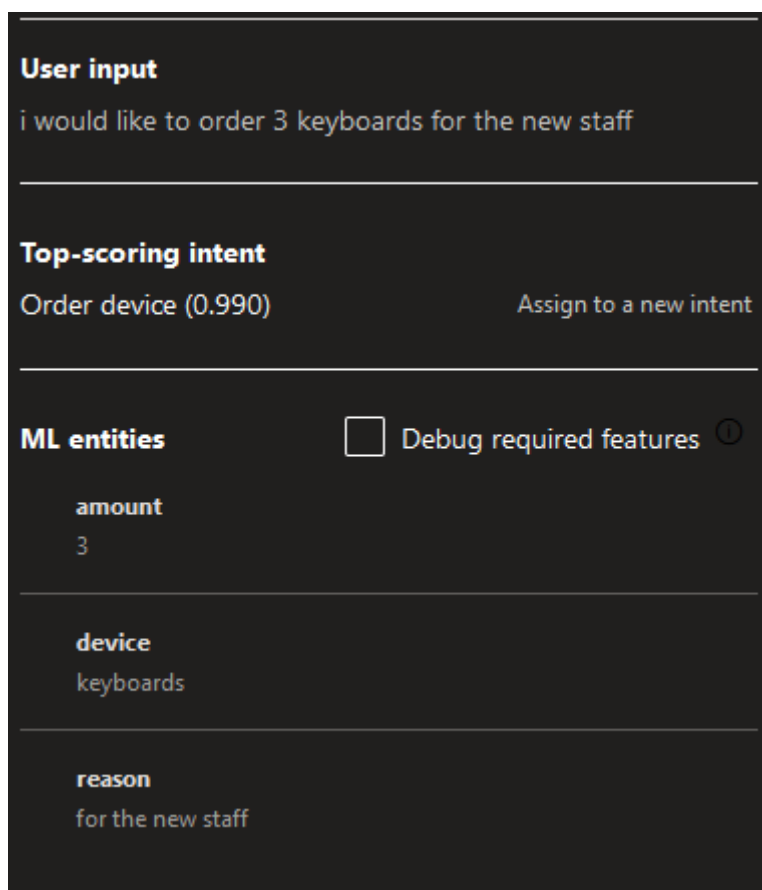
Communicatie met LUIS gebeurt aan de hand van JSON-objecten, dit is een gestandaardiseerd gegevensformaat in tekstvorm. Het antwoord van de gebruiker van de chatbot wordt ernaar omgezet en verstuurd naar LUIS, via bepaalde sleutels eigen aan de LUIS-applicatie. LUIS analyseert deze input, en geeft dan ook een JSON-object terug. Dit bevat twee belangrijke bronnen van informatie over het oorspronkelijke bericht, namelijk wat de vermoedelijke ‘intent’ is, en welke ‘entities’ er herkend zijn.

Een ‘intent’, of in het Nederlands ‘bedoeling’, omschrijft wat de gebruiker wou bereiken bij het versturen van het bericht. Een gebruiker kan zo bij voorbeeld een probleem willen registreren, of een bestelling willen plaatsen. Als tweede herkent LUIS ook ‘entities’ binnen het bericht. Deze entiteiten kunnen dan gebruikt worden om de ‘intent’ van de gebruiker te volbrengen.

LUIS - trainen

LUIS is op voorhand al getraind voor een ruim aantal scenario’s, maar moet ook bijgetraind worden voor specifieke toepassingen. De procedure is eenvoudig, ten eerste worden alle mogelijke intents opgelijst. Om even te herhalen, een intent is een mogelijk doel van een eindgebruiker. Een intent omschrijft een taak die de gebruiker wil voltooien aan de hand van de bot, en is dus een functie van die bot. Vervolgens worden er voor elke intent enkele voorbeeldzinnen verzameld. Op dit moment kan LUIS al getraind worden, en is het hierna in staat om al intents te herkennen.

Naast het herkennen van de bedoeling van de gebruiker, is het ook belangrijk dat er herkend wordt welke objecten er binnen de boodschap gebruikt worden. Een mogelijke voorbeeldzin van een gebruiker van de chatbot ter illustratie: ‘ik wil graag 3 laptops bestellen voor het nieuwe personeel’. LUIS is op dit punt al in staat om af te leiden dat de intent ‘bestel apparaat’ is, maar is nog niet in staat om te identificeren over welk apparaat het gaat, welk aantal, en waarom. ‘Apparaat’, ‘aantal’ en ‘waarom’ zijn dus voorbeelden



Figuur 2.6: Verduidelijking antwoord LUIS, eigen afbeelding

van entiteiten, elk met hun eigen waarden.

Om deze te trainen binnen LUIS, wordt er teruggegaan naar de voorbeeldzinnen die werden gebruikt om de intents te trainen. Als er binnen een zin een entiteit voorkomt, selecteert men deze en duidt men aan over welke entiteit het gaat. Terug naar de zin die daarnet als illustratie aangehaald werd, '3' wordt aangeduid als entiteit 'aantal', 'laptops' is een apparaat en 'voor het nieuwe personeel' verklaart 'waarom'. Na het ingeven van deze informatie wordt het model opnieuw getraind. Opnieuw wordt er gebruik gemaakt van natural language processing voor het trainen met en herkennen van entiteiten, en is LUIS in staat om voorbeelden van entiteiten die niet in de trainingszinnen voorkomen toch correct te herkennen, en aan de juiste intent toe te wijzen. Nogmaals een voorbeeld ter verduidelijking. Wanneer men in de voorbeeldzinnen 'toetsenbord', 'speaker', 'webcam', 'scherm', 'laptop' en 'kabel' aanduidt als entiteit 'apparaat', is LUIS in staat om zelf 'muis' als apparaat te herkennen. Ook voor elke entiteit wordt er een zekerheidsscore teruggegeven. Nu is LUIS niet enkel in staat om uit een boodschap de bedoeling te extraheren, maar ook om binnen diezelfde boodschap verschillende objecten aan te duiden.

Azure Bot Framework

Azure Bot Framework is een volgend framework, ook geproduceerd door Microsoft. Hiermee wordt de effectieve chatbot opgesteld en geconfigureerd. Aangezien het ook



Figuur 2.7: Voorbeeldzinnen gelabeld met entiteiten, eigen afbeelding

een product is van Microsoft, net zoals LUIS, zorgt dit voor een eenvoudige, en goed ondersteunde, communicatie tussen beide frameworks.

2.0.12 Gebruikte software binnen Wolters Kluwer

Binnen Wolters Kluwer wordt er een set van software gebruikt ter ondersteuning van hun personeel. Vele van deze software is eigendom van Microsoft, waardoor de keuze van software voor dit project ook naar producten van Microsoft neigde. Dit om implementatie binnen en communicatie met bestaande software te vereenvoudigen.

Jira

Oorspronkelijk ontworpen door softwaregigant Atlassian als bug- en probleemopvolgingssoftware, is de functionaliteit van deze applicatie al sterk uitgebreid. Vandaag omvat Jira nog altijd zijn oorspronkelijke taak, maar is de software ook inzetbaar als werkbeheertool van vereisten- en testcasebeheer tot tool voor agile softwareontwikkeling.

Binnen deze proef is vooral de oorspronkelijke functionaliteit als bug- en probleemopvolgingssoftware belangrijk. Met behulp van de geschiedenis van oude problemen vormt het een belangrijke knowledge base voor het vinden van oplossingen. Het is de bedoeling om de informatie uit deze knowledge base te migreren naar een centrale knowledge base. Vervolgens kan de chatbot deze centrale knowledge base aanspreken om gebruikers een oplossing te bieden voor eenvoudige problemen. Daarnaast is het ook de bedoeling dat, indien de bot niet in staat is om een oplossing te vinden, er een ticket wordt aangemaakt. Dit ticket bevat alle informatie rond het probleem, van gebruiker bij wie het voorkwam, tot apparaat waarop het probleem zich voordoet, tot probleemomschrijving. Dit alles kan op Jira geregistreerd worden door de chatbot aan de hand van een API.

Confluence

Confluence is een wiki-applicatie die ook ontwikkeld werd door Atlassian, en heeft een sterke integratie met Jira. Deze software, samen met Sharepoint (zie hierna) vormen ze de twee grootste knowledge bases binnen Wolters Kluwer, en bevatten dus informatie over vorige problemen en hoe deze op te lossen. Nogmaals is het de bedoeling dat al deze

informatie gemigreerd wordt naar een centrale knowledge base, waarna de chatbot deze kan aanspreken, opdat het op die manier oplossingen voor problemen kan vinden.

Microsoft Teams

Microsoft Teams is communicatiesoftware. Binnen de applicatie is het mogelijk om afspraken te plannen, te chatten en te bellen met anderen. De doelstelling is om de chatbot binnen deze software te implementeren, aangezien Microsoft Teams al in sterk gebruik is binnen Wolters Kluwer. Op deze manier wordt het gebruik van de bot een stuk eenvoudiger. Een gebruiker moet niet naar een apart platform navigeren, zo wordt het gebruik van de bot ook gemotiveerd.

Sharepoint

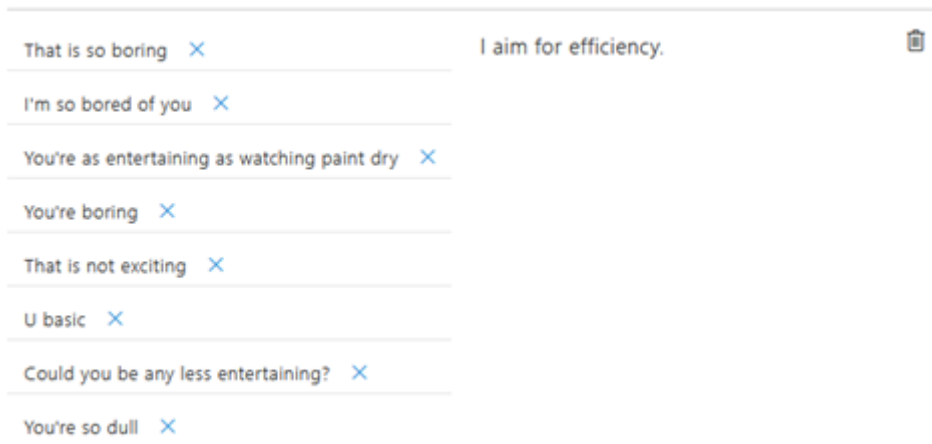
Microsoft biedt ook software aan voor het opbouwen van websites, genaamd Sharepoint. Deze applicatie vormt een goede basis waarmee er een centrale en veilige plaats kan opgebouwd worden voor het opslaan, ordenen en openen van gegevens. Ook deze software vormt een knowledge base binnen Wolters Kluwer. Het is dus de bedoeling dat de chatbot, aan de hand van een centrale knowledge base, wordt getraind op en gebruik maakt van de informatie uit Sharepoint, Confluence en ten slotte ook Jira.

2.0.13 Bot Framework Emulator

Bot Framework Emulator is een desktopapplicatie van Microsoft die gebruikt kan worden om chatbots te testen en te debuggen. Aan de hand van dit programma kan een chatbot aangesproken en gebruikt worden, nog voor de echte implementatie in bijvoorbeeld Microsoft Teams. Deze applicatie kan zowel lokaal op een computer gebruikt worden, als op afstand via een tunnel. Het is beschikbaar voor zowel Windows, Mac als Linux.

2.0.14 QnA Maker

Een volgend product van Microsoft dat binnen deze proef gebruikt wordt is QnA Maker. Dit is een applicatie die kan geïntegreerd worden binnen een chatbot, en functionaliteit voorziet voor eenvoudige vragen en antwoorden. Een voorbeeld van implementatie is om de FAQ, of meest voorkomende vragen, hierbinnen voor te programmeren. Dit alles wordt opgezet aan de hand van een knowledge base. Hierin worden de meest voorkomende vragen opgelijst, met het bijhorende antwoord. QnA Maker is slechts beperkt in staat om taal te begrijpen, dus het is belangrijk om voor elk antwoord de vraag op zoveel mogelijk verschillende manieren te formuleren. QnA Maker kan gebruikt worden als centrale knowledge base, waarin alle informatie van Wolters Kluwer over vorige problemen opgeslagen wordt. Naast het programmeren van eigen vragen en antwoorden, voorziet QnA Maker ook al een grote 'standaard' knowledge base. Hierin zijn vragen en antwoorden opgelijst een chatbot gesteld worden. Deze past zich aan op basis van een instelling, voor bedrijven is dit formeel, voor persoonlijk gebruik niet.



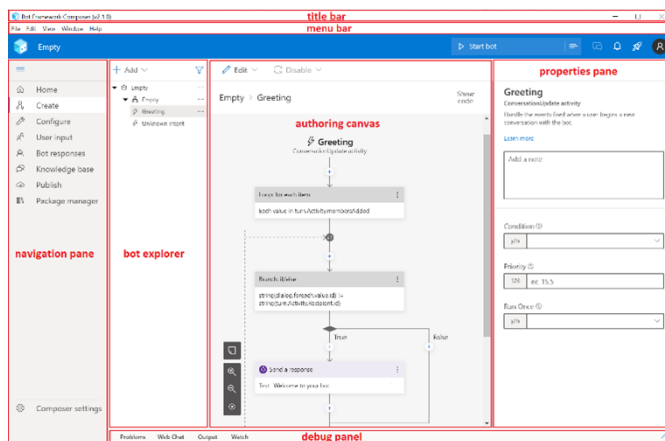
Figuur 2.8: Fragment uit voorgeprogrammeerde knowledge base van QnA maker. Links de vraag, rechts het antwoord. Eigen afbeelding

2.0.15 Verschillen tussen QnA Maker en LUIS

Het verschil tussen deze applicatie en LUIS is dat QnA Maker enkel geschikt is om eenvoudige vragen te beantwoorden. De natural language understanding geïmplementeerd binnen QnA Maker is erg beperkt, in tegenstelling tot LUIS. Zoals hiervoor beschreven is LUIS in staat om effectieve betekenissen en objecten uit berichten te halen. Wel kan LUIS zelf geen antwoord op een vraag bieden, maar geeft wel de informatie terug die een chatbot kan gebruiken om een antwoord te geven of een taak uit te voeren.

2.0.16 Bot Framework Composer

Ook Bot Framework Composer is een programma aangeboden door Microsoft en beschikbaar via Azure. Deze software biedt een eenvoudige oplossing om een chatbot op te bouwen, zonder zelf code te hoeven schrijven. Het is dus een alternatief voor het zelf opstellen van een chatbot aan de hand van Python, zoals hiervoor al aangehaald werd. Een grafische interface vervangt de code binnen Bot Framework Composer. Daarnaast is het mogelijk om binnen deze software de verschillende services die ook aangeboden worden op Microsoft Azure, waaronder LUIS en QnA maker, te integreren in de chatbot.



Figuur 2.9: Grafische interface van Bot Framework Composer.
<https://docs.microsoft.com/en-us/composer/media/introduction/composer-overview-image.png#lightbox>

3. Methodologie

3.0.1 Onderzoek naar optimale manier van opbouwen van de chatbot

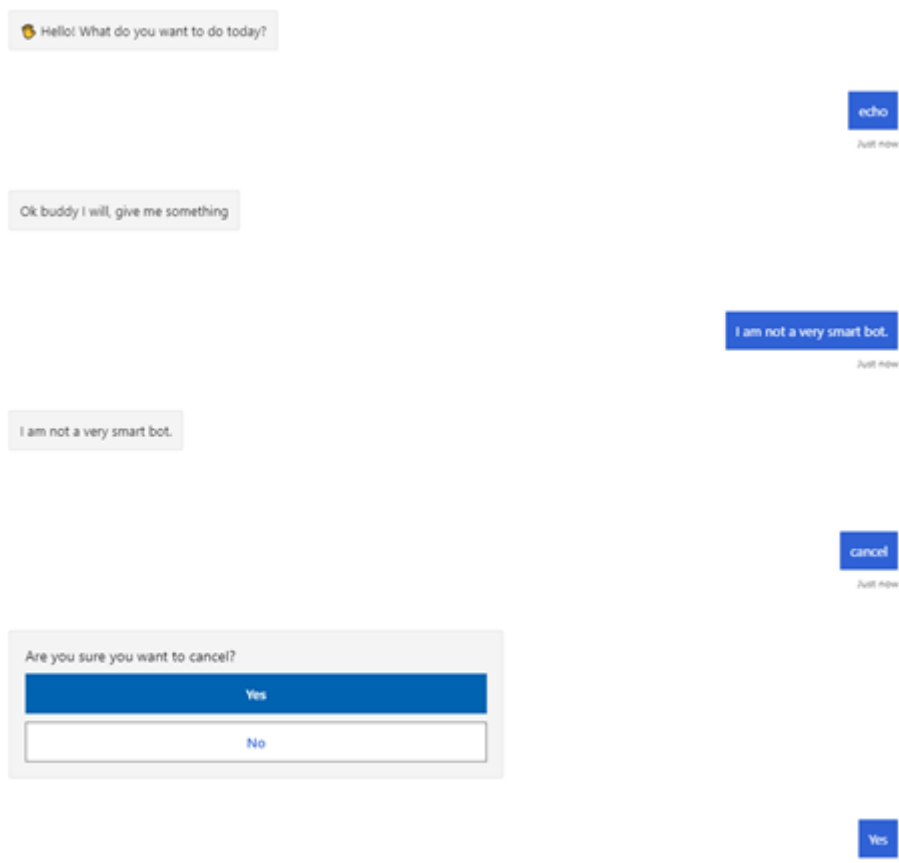
Er zijn vele verschillende bedrijven die platformen voor het opbouwen van een chatbot aanbieden, elk met andere functies en beperkingen. Het selecteren van het correcte platform is dus van uiterst groot belang. Na evaluatie van de mogelijkheden en overleg met Wolters Kluwer, werd er voor Azure Bot Framework gekozen. Dit is een product van Microsoft, en integreert makkelijk met andere software van diezelfde producent. Binnen Wolters Kluwer is Teams namelijk uitvoerig in gebruik, en vlotte implementatie van de bot binnen deze applicatie is een prioriteit. Naast makkelijke integratie biedt Azure Bot Framework nog voordelen. Zo is de documentatie uitgebreid, en biedt Microsoft enkele services ter ondersteuning en uitbreiding van de bot, waaronder LUIS en QnA Maker.

Eenmaal het platform geselecteerd is, is het tijd om na te denken over hoe de verschillende beoogde functionaliteiten kunnen geïmplementeerd worden, en welke services hier hulp bij kunnen bieden. Oorspronkelijk was het de bedoeling om de chatbot vanaf nul op te bouwen aan de hand van Python en Azure Bot Framework, om dan later functionaliteit uit te breiden aan de hand van LUIS en QnA Maker. Dit bleek niet haalbaar, desondanks de goede documentatie zorgde het gebrek aan voorbeelden van implementatie van zo een bot in Python voor te veel moeilijkheden. Python is niet geschikt voor applicaties met een grote complexiteit. Hierna werd er overgeschakeld naar Azure Bot Framework Composer. Deze applicatie maakt gebruik van dezelfde Azure Bot Framework softwareontwikkelingskit, maar biedt een eenvoudigere manier, zonder code, om de bot op te stellen. Alles gebeurt namelijk aan de hand van een grafische interface. Aangezien het gebruik maakt van dezelfde softwareontwikkelingskit, blijft het mogelijk om na het opzetten van de bot de functionaliteit uit te breiden en te verbeteren aan de hand van LUIS en QnA Maker.

3.0.2 Opbouwen van de chatbot deel 1: echoBot

Wat is een echobot?

Als eerste werd er een zeer eenvoudige chatbot opgesteld, namelijk een echoBot. Het doel van het opstellen van deze bot, en van de bot zelf, is om een eerste kennismaking met de ontwikkelingsomgeving en de chatbot te vormen. Op deze manier kan er ingeschat worden wat de mogelijkheden zijn van het platform en hoe complex het is om een chatbot in praktijk op te zetten. De functionaliteit van de bot zelf is erg beperkt. Eerst groet de bot de gebruiker. Hierna kan deze gebruiker het 'echo'-commando oproepen via een bericht naar de bot dat het woord 'echo' bevat. Vervolgens kan men de bot een boodschap sturen, waarop de bot antwoordt met diezelfde boodschap. De bot blijft de boodschap van de gebruiker herhalen totdat het een stopwoord als antwoord ontvangt. Enkele voorbeelden van deze stopwoorden zijn 'cancel', 'stop' en 'quit'.

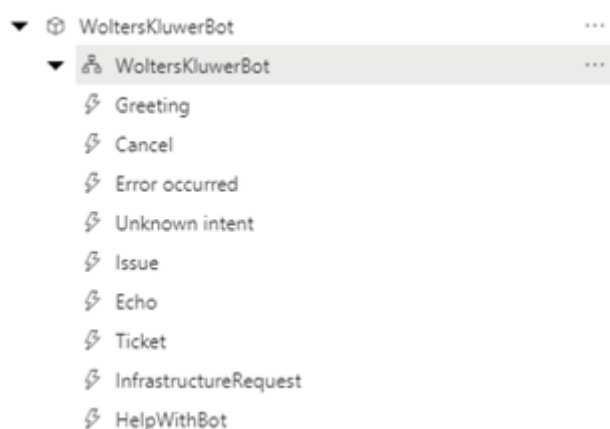


Figuur 3.1: Voorbeeld van een echogesprek met de bot, eigen afbeelding

Hoe wordt zoiets geïmplementeerd binnen Bot Framework Composer?

Zoals hiervoor vermeld dient deze echoBot als kennismaking met de Bot Framework Composer. Het opzetten van de bot loopt als volgt. De bot heeft een lijst aan triggers, deze komen overeen met de intents binnen LUIS. Wanneer een van deze triggers opgeroepen

wordt, wordt er een bijhorende dialoog gestart. Deze dialoog helpt dan de gebruiker verder, en is in staat om zelf ook subdialogen te starten.



Figuur 3.2: Lijst van triggers binnen de WoltersKluwerBot, eigen afbeelding

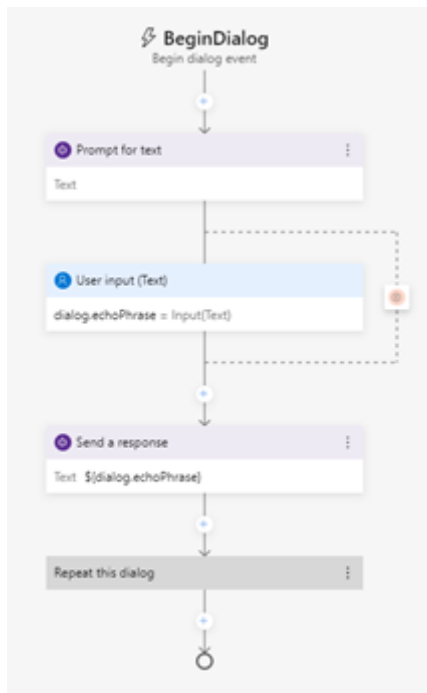
Even een korte illustratie aan de hand van de echoBot. Als eerste wordt er een trigger voorzien. Deze trigger wordt getraind met LUIS aan de hand van een aantal voorbeeldzinnen. Wanneer de bot, met behulp van LUIS een van deze zinnen of een soortgelijke zin herkent, wordt de echodialoog gestart. Deze dialoog is een chronologisch verloop. Eerst wordt er gevraagd naar een input, waarna deze herhaald wordt. Op het einde van het verloop wordt de dialoog herhaald. De dialoog kan slechts op een manier gestopt worden, namelijk door een ‘interrupt’ of onderbreking. Deze onderbreking wordt gestart wanneer er een andere trigger, waarvoor er toestemming voor onderbreking geconfigureerd is, herkend wordt. Hier is er ingesteld dat enkel en alleen de ‘cancel’ trigger een onderbreking kan veroorzaken. Wanneer deze trigger herkend wordt, wordt de canceldialoog gestart. Deze dialoog is op zijn beurt ook een chronologisch verloop die het afsluiten van de echoBot helpt afhandelen.

3.0.3 Opbouwen van de chatbot deel 2: opzet en implementatie LUIS

Een volgende stap voor de chatbot is de implementatie van LUIS. De bot ‘slim’ maken is wat hiermee bereikt wordt. Zo begrijpt de chatbot niet enkel voorgeprogrammeerde zinnen, maar is het ook in staat om aan de hand van een aantal voorbeelden nieuwe, gelijkaardige zinnen ook te herkennen.

Opzet LUIS

Vooraleer LUIS kan geïntegreerd worden, moet er eerst een model getraind worden. LUIS is al voorgetraind voor een groot aantal scenario’s, maar het bijtrainen voor specifieke doelstellingen is essentieel. De procedure gaat als volgt. Als eerste wordt er een geschikte omgeving opgezet binnen Azure. Als volgt worden alle mogelijke ‘intents’ opgelijst. Elke intent, of bedoeling, vertegenwoordigt een functionaliteit binnen de bot. Een voorbeeld hiervan is ‘registreer probleem’ of ‘bestel apparaat’. Voor elk van deze worden er enkele



Figuur 3.3: Verloop van de echodialoog, eigen afbeelding

voorbeeldzinnen ingelezen. Vervolgens wordt het model getraind, en kan het getest worden. Voor elke zin die men nu test binnen het LUIS-model, geeft LUIS de vermoedelijke intent terug, en hierbij een zekerheidsscore. Deze testzinnen moeten geen letterlijke kopieën te zijn van de voorbeeldzinnen, LUIS is intelligent genoeg om aan de hand van machine learning, specifiek natural language processing, aan soortgelijke zinnen toch de correcte intent toe te wijzen. Wanneer er geen getrainde intent herkend wordt, wordt er 'None' teruggegeven.

Implementatie LUIS

Eenmaal getraind, kan er een link gelegd worden tussen LUIS en de chatbot. Dit is eenvoudig, er is ondersteuning voor LUIS binnen het Azure Bot Framework, en gebeurt aan de hand van een aantal sleutels. Communicatie van en naar LUIS vanuit de chatbot gebeurt aan de hand van JSON-objecten. Vanuit de bot naar LUIS is dit een eenvoudig object met daarin de boodschap die de bot binnenkreeg van de gebruiker. LUIS geeft dan op zijn beurt een JSON-object terug, met hierin alle mogelijke herkende intents, en per intent alle mogelijke herkende entities. Wanneer de bot opgebouwd wordt via eigen code moet de verwerking van deze objecten zelf voorzien worden. Binnen Bot Framework Composer wordt dit allemaal automatisch afgehandeld.

Zoals aangetoond bij de echoBot kunnen de trainingszinnen voor LUIS binnen Bot Framework Composer ingevuld worden. Deze worden dan gebruikt om triggers op te roepen. Via die triggers start er een dialoog, die dan de gebruiker verder helpt. Een voorbeeld van zo een dialoog is het aanmaken van een ticket voor een probleem, of het verzamelen van informatie voor het bestellen van hardware.


3.0.4 Opbouwen van de chatbot deel 3: aanvullen met QnA Maker

Waarom QnA Maker implementeren bovenop LUIS?

LUIS is intelligent en kan in vele situaties gebruikt worden, maar is complex om te configureren. Voor vragen die vaak voorkomen, bijvoorbeeld een vraag naar contactinformatie of een link naar een bepaalde informatiebron, is het makkelijker om een ander systeem te gebruiken. Azure voorziet hiervoor QnA Maker. Deze applicatie is eenvoudig in gebruik, en er kan een zeer grote hoeveelheid vragen en antwoorden toegevoegd worden in een korte tijd.

Trainen QnA Maker

Het trainen van QnA Maker gaat als volgt. Op basis van een vaak voorkomende vraag, met hierbij bijhorend antwoordt, wordt er een entry aangemaakt in de knowledge base. Vervolgens worden er zoveel mogelijk alternatieve verwoordingen voor diezelfde vraag voorzien, opdat de bot de vraag in meer scenario's zou kunnen herkennen. Vervolgens wordt de vooruitgang opgeslagen en de bot getraind met de nieuwe data, waarna de knowledge base gepubliceerd wordt. Eenmaal gepubliceerd, is het mogelijk om de getrainde QnA Maker knowledge base te implementeren binnen een andere applicatie, bijvoorbeeld een chatbot binnen Bot Framework Composer.

Question	Answer
^ Editorial	
How do I contact Jan Delamper? X	You can send him an email at jan.delamper@wolterskluwer.com . 
Contact Jan Delamper X	
I would like to speak with Jan Delamper. X	
What is Jan Delamper's email address? X	
+ Add alternative phrasing	

Figuur 3.4: Voorbeeld van een zelf getrainde vraag en antwoord, eigen afbeelding

Implementatie QnA Maker

Opnieuw is het erg eenvoudig om een Azure applicatie te integreren binnen Bot Framework Composer. Slechts enkele sleutels worden ingevuld in de instellingen om de link te leggen met het getrainde QnA Maker knowledge base.

Een voorbeeld van implementatie is om deze getrainde knowledge base op te roepen elke keer wanneer er geen intent herkend wordt. De knowledge base voorziet naast de zelf getrainde vragen en antwoorden ook een grote hoeveelheid standaardzinnen, die elke situatie op een propere manier kunnen afronden, ookal is de bot hier eigenlijk niet voor voorzien.

3.0.5 Opbouwen van de chatbot deel 4: integratie Microsoft Teams

Waarom Microsoft Teams?

Binnen Wolters Kluwer, het bedrijf waarvoor deze paper geschreven wordt, is Microsoft Teams al in uitgebreid gebruik. Het staat standaard op elke computer geïnstalleerd, en werknemers gebruiken deze software intern om onderling te communiceren.

Het is de bedoeling om deze chatbot zo toegankelijk mogelijk te maken, opdat men eerder de bot zou gebruiken dan kostbare tijd van een collega. Wanneer de bot in Teams geïntegreerd wordt, is er geen nood om andere applicaties te downloaden. Daarnaast is dit ook eenvoudiger, zowel voor de persoon die de bot opstelt als voor de gebruiker, dan het voorzien van een webapp.

Implementatie

Als onderdeel van de Microsoft Azure suite, is de integratie van Microsoft Teams binnen Bot Framework Composer eenvoudig. Eenmaal de bot zelf afgewerkt is, is het mogelijk om deze te publiceren. Op die manier is het niet meer nodig om lokaal altijd een instantie van de bot te laten lopen, maar loopt deze instantie online in de Azure cloud. Vervolgens kan de connectie met Microsoft Teams aangeschakeld worden in het online configuratiepaneel van Azure. Hierdoor wordt er een link beschikbaar gesteld. Wanneer een gebruiker op de link klikt, wordt er een nieuwe privé Microsoft Teams-conversatie gestart. Binnen deze conversatie kan de gebruiker de bot altijd aanspreken. Er is geen nood om nogmaals de link te gebruiken.

4. Proof of concept

4.0.1 Context

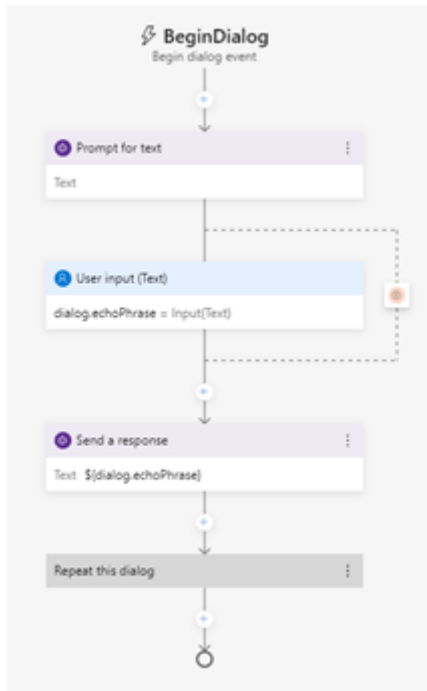
Zoals vermeld in de methodologie is deze bot opgebouwd met behulp van Azure Bot Framework Composer. Dit vereenvoudigt het opstellen van de bot maar beperkt de functionaliteiten. Het is met andere woorden mogelijk om een chatbot op te bouwen die slimmer en efficiënter werkt, maar dit is niet het doel van deze proof of concept. Deze proof of concept dient ter illustratie van hoe er op een eenvoudige manier een bot kan opgebouwd worden, en wat er allemaal mogelijk is binnen het framework. De functionaliteiten die hieronder opgelijst worden zijn met andere woorden maar een fractie van wat er mogelijk is, maar bewijzen toch dat de bot toch in staat is om enkele menselijke taken te vervangen of vereenvoudigen.

Vele van de functionaliteiten worden opgeroepen aan de hand van een commando. Dit commando is flexibel, aangezien er gebruik wordt gemaakt van natural language processing met behulp van LUIS. Met andere woorden, er is geen nood om het commando exact te gebruiken. Een variatie in verwoording vormt geen probleem, en de bot herkent het commando even goed. Deze bot is beschikbaar via Microsoft Teams aan de hand van onderstaande link. <https://teams.microsoft.com/l/chat/0/0?users=28:8897d915-6e6f-4923-8921-9af2a4ad07de>

4.0.2 Functionaliteiten

Groet

Een eerste functionaliteit van de bot is de groet. Wanneer een gebruiker voor het eerst een chatgesprek start met de bot, wordt deze gebruiker gegroet. Momenteel is dit een eenvoudige, uitnodigende groet.



Figuur 4.1: Groet van de bot, eigen afbeelding

Echo

De echoBot functionaliteit besproken binnen het onderdeel methodologie is nog altijd onderdeel van de finale versie van de bot. Dit omdat het een belangrijk onderdeel van het leerproces vormde, en ook belangrijk is voor wie deze paper wil reproduceren.

De functie en werking van dit onderdeel werd al uitvoerig besproken, maar hierbij een korte samenvatting. De gebruiker activeert de echodialoog aan de hand van het 'echo' commando. Hierna herhaalt de bot alles wat de gebruiker stuurt. De dialoog stoppen gebeurt aan de hand van het 'cancel' commando.

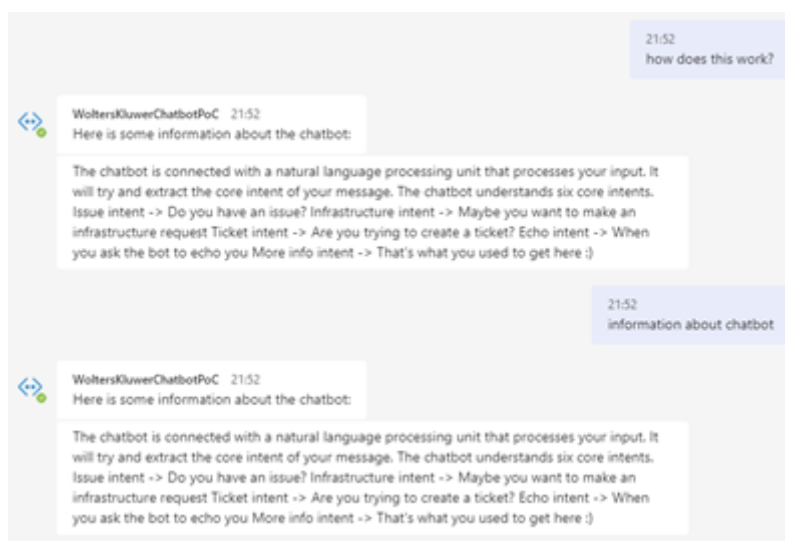
Informatie opvragen over de bot

Wanneer de gebruiker niet goed snapt hoe te werken met de chatbot, kan er naar extra informatie gevraagd worden. Het oproepen van deze extra informatie gebeurt aan de hand van ruime lijst aan commando's die allemaal dezelfde boodschap teruggeven. Dit opdat een gebruiker zonder kennis van de bot hier ook toe in staat is.

Eenmaal de bot herkent dat de gebruiker nood heeft aan extra informatie, wordt er geantwoord met een lijst aan commando's en een beschrijving van de werking van elk van deze commando's.

Ticket aanmaken

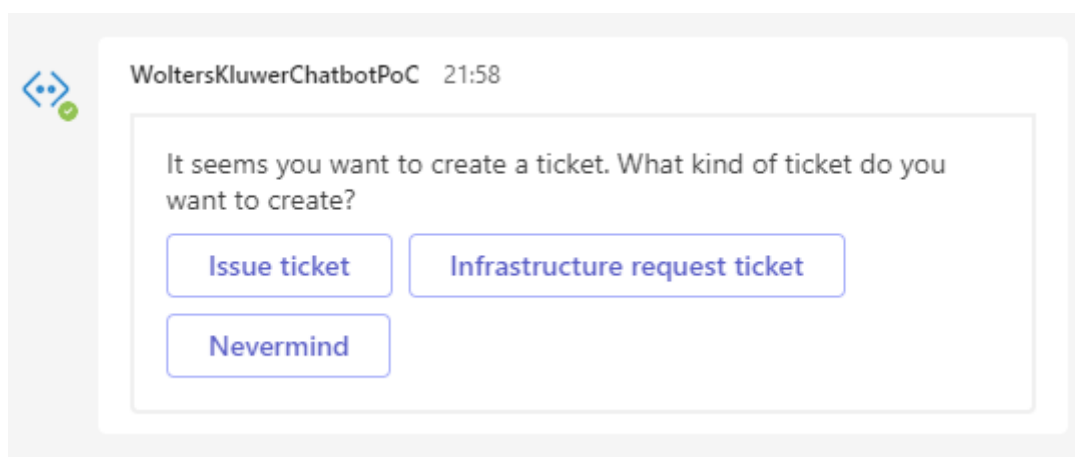
Via het 'ticket'-commando is de bot in staat om twee verschillende vormen van ticket aan te maken. 'Ticket' beschrijft hier een formulier waarbinnen er informatie verzameld wordt, om dan later afgehandeld te worden door een lid van het ondersteuningsteam. Aan de



Figuur 4.2: Opvragen extra informatie aan de hand van twee verschillende commando's, eigen afbeelding

hand van een vragenlijst wordt er verzekerd dat de bot alle benodigde informatie voor het probleem volledig en correct verzamelt. Zelfs al wordt er uiteindelijk toch een werknemer ingeschakeld om dit probleem te helpen oplossen, toch wordt de taak van deze werknemer vereenvoudigd.

Binnen het aanmaken van de twee tickets worden dezelfde commando's gebruikt. Wanneer men binnen het formulier het 'cancel'-commando gebruikt, krijgt de gebruiker de optie om, na bevestiging, het formulier vroegtijdig af te sluiten. Bij gebruik van het 'restart'-commando, krijgt de gebruiker de optie om, na bevestiging, het formulier vanaf het begin te herstarten.



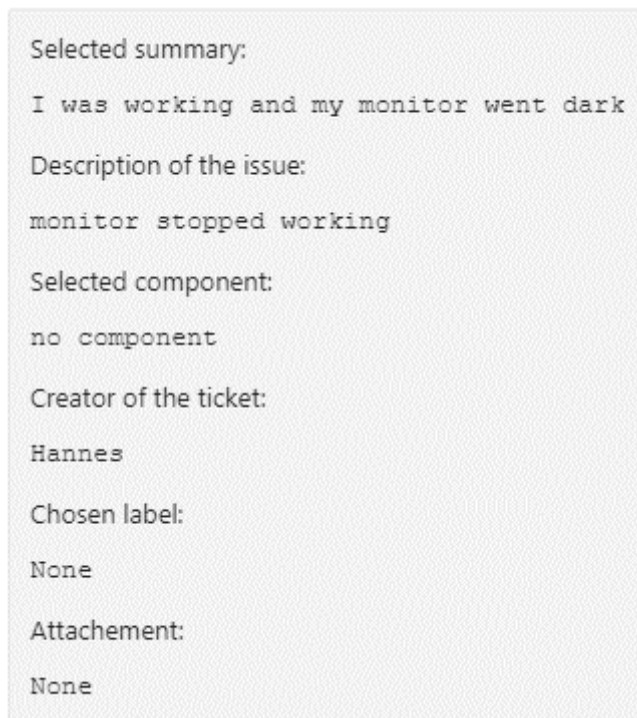
Figuur 4.3: Dialoog na herkennen 'ticket' commando, eigen afbeelding

Probleem registreren

Probleem registreren, of 'issue ticket' is een van de hiervoor benoemde formulieren. Het doel van dit formulier is om alle benodigde informatie te verzamelen rond een probleem,

opdat een lid van het ondersteuningsteam vlot dit probleem kan afhandelen. Op deze manier bespaart dit teamlid tijd aangezien er geen informatie meer moet verzameld worden. 'Issue' is het commando dat hier gebruikt wordt. Wanneer de bot dit herkent, wordt de dialoog gestart. Als eerste wordt er gevraagd of de gebruiker weldegelijk een probleem wil registreren. Eenmaal bevestigd, wordt er in volgorde een lijst van vragen gesteld aan de gebruiker. Volgende informatie wordt verzameld rond het probleem: een korte samenvatting, een beschrijving, een component, de naam van de gebruiker, en optioneel ook een label en attachment. Component wordt opgevraagd aan de hand van een selectlist, of een lijst met een aantal vaste opties. De gebruiker kan enkel een van de gelijste opties als antwoord selecteren. Ook dit dient ter illustratie van wat er allemaal mogelijk is binnen Bot Framework Composer. Als laatste vraagt de bot of de gebruiker nog een overzicht van het ticket zou wensen, en print deze indien nodig.

Deze structuur binnen het ticket is gemodelleerd naar de huidige structuur van een ticket binnen Wolters Kluwer.



```
Selected summary:
I was working and my monitor went dark

Description of the issue:
monitor stopped working

Selected component:
no component

Creator of the ticket:
Hannes

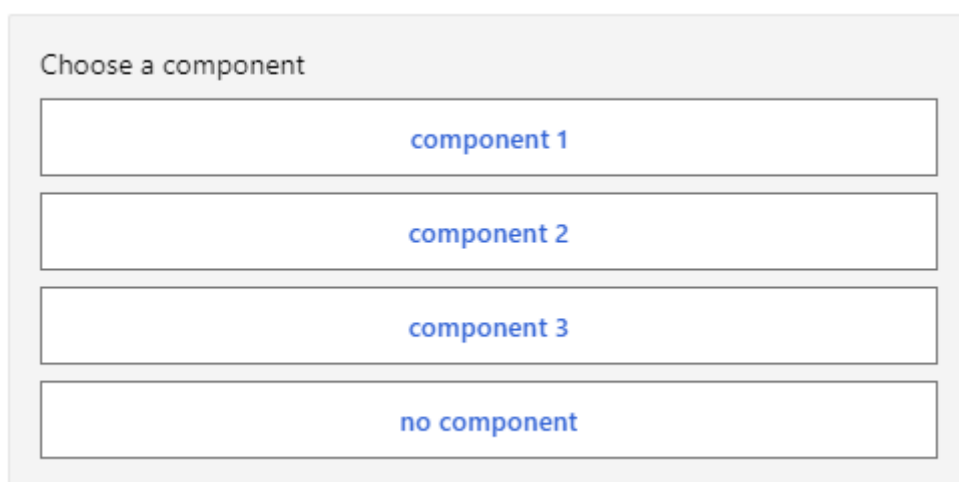
Chosen label:
None

Attachement:
None
```

Figuur 4.4: Overzicht van een geregistreerd probleem, eigen afbeelding

Infrastructuur aanvraag

Een tweede formulier is voorzien om informatie te verzamelen rond een infrastructuur aanvraag, of 'infrastructure request'. Dit wordt gebruikt wanneer een lid van een van de ontwikkelingsteams nood heeft aan nieuwe hardware of infrastructuur. In het huidige systeem contacteert hij of zij hiervoor rechtstreeks een lid van het ondersteuningsteam, maar ook hier kan er tijd bespaard worden door een bot de benodigde informatie te laten verzamelen. Hierna kan een lid van het ondersteuningsteam het ticket vlot afhandelen. Binnen dit ticket worden volgende vragen gesteld, in volgorde: het scrumteam, de SPOC



Choose a component

component 1

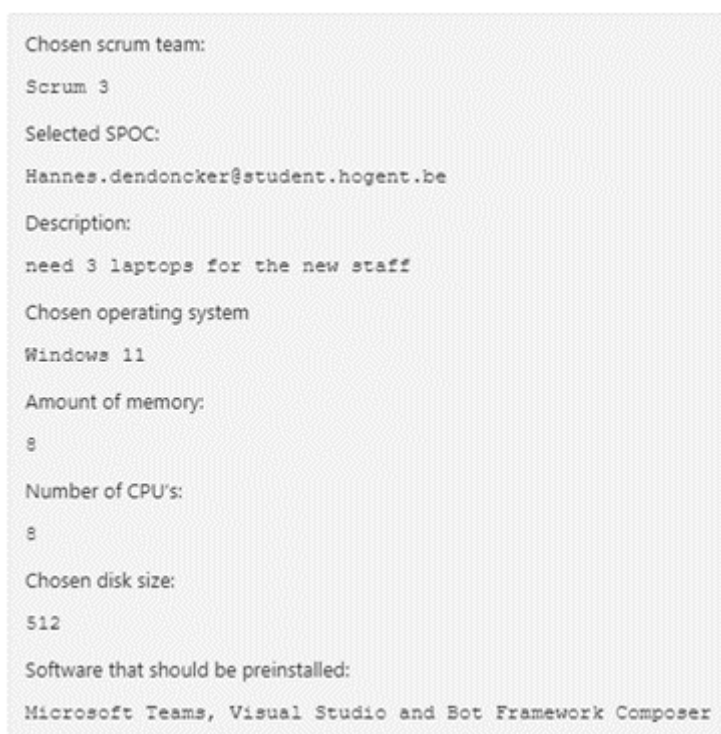
component 2

component 3

no component

Figuur 4.5: Opvragen component aan de hand van een selectlist, eigen afbeelding

of contactpersoon, een beschrijving, het besturingssysteem aan de hand van een selectlist, de hoeveelheid geheugen, het aantal processorkernen, de opslagcapaciteit en als laatste welke software er voorzien moet worden. Ook hier krijgt de gebruiker een optie om een overzicht aan te vragen.



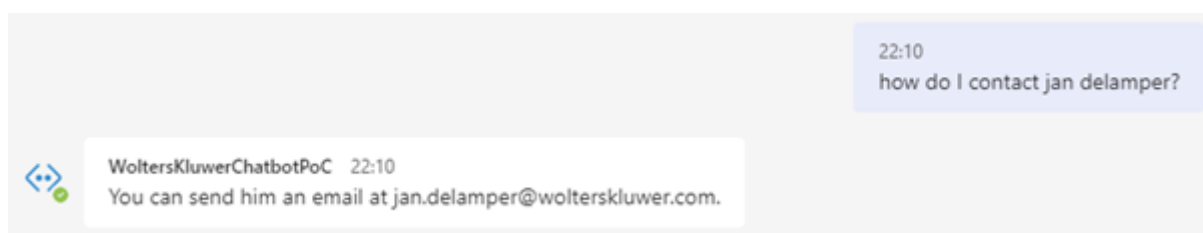
```
Chosen scrum team:  
Scrum 3  
Selected SPOC:  
Hannes.dendoncker@student.hogent.be  
Description:  
need 3 laptops for the new staff  
Chosen operating system  
Windows 11  
Amount of memory:  
8  
Number of CPU's:  
8  
Chosen disk size:  
512  
Software that should be preinstalled:  
Microsoft Teams, Visual Studio and Bot Framework Composer
```

Figuur 4.6: Overzicht van een infrastructuur aanvraag, eigen afbeelding

Contactgegevens

Dit onderdeel maakt gebruik van QnA Maker, en dient ten voorbeeld van wat er mogelijk is met deze software. QnA Maker wordt gebruikt om eenvoudige antwoorden te bieden op eenvoudige vragen. Met eenvoudig wordt bedoeld dat zowel de vragen als antwoorden vaststaan, en dus niet interactief zijn. Een voorbeeld hiervan is dus contactinformatie, dit ligt normaalgezien vast en verandert niet.

Binnen QnA Maker wordt dit geïmplementeerd aan de hand van een knowledge base, deze bevat vragen en antwoorden. Voor elk antwoord wordt deze vraag op meerdere manieren geformuleerd, opdat QnA Maker deze beter zou herkennen. Er wordt namelijk beperkt gebruik gemaakt van een natural language processing model dat getraind wordt op basis van de verschillende voorbeeldzinnen.



Figuur 4.7: Opvragen van contactgegevens met behulp van QnA Maker, eigen afbeelding

4.0.3 Mogelijke uitbreidingen

Zoals al meermaals vermeld is de chatbot die hier geproduceerd werd geen afgewerkt product. Het is een proof of concept, en dient ter illustratie van wat er allemaal mogelijk is binnen de gebruikte software, maar ook wat de beperkingen zijn.

Binnen de proof of concept worden vele functionaliteiten kort besproken. In dit onderdeel wordt er beschreven hoe deze verder kunnen uitgebreid worden. In het geval dat der binnen Wolters Kluwer beslist wordt om een bot op te bouwen op basis van deze proof of concept, vormt dit onderdeel een voorstel van wat er nog geïmplementeerd kan worden.

Naast het uitbreiden van bestaande functionaliteiten worden er ook enkele nieuwe functionaliteiten besproken. Deze zijn niet mogelijk binnen Bot Framework Composer, en vereisen dus een andere aanpak voor de opbouw van de bot.

Uitbreiden functionaliteiten

Binnen de proof of concept gebruikt de bot verschillende manieren om informatie van de gebruiker te verkrijgen. Selectlists, een vraag naar tekst, een vraag naar een integer of een vraag naar een bijlage zijn hier voorbeelden van. Momenteel worden deze enkel gebruikt voor het opstellen van twee tickets, maar aan de hand van deze vragen kan er ook veel andere informatie verzameld worden. De bot kan bijvoorbeeld ingezet worden om te polsen naar tevredenheid bij de werknemers aan de hand van een enquête. Niet enkel tevredenheid over de bot, maar ook tevredenheid op de werkvloer kan onderzocht worden. Een ander voorstel is het aanvragen van verlof aan de hand van de bot. Verbonden met het internet, kan de bot controleren of een verlofperiode niet voor conflicten zorgt met

andere werknemers, en kan dit alles op een automatische en efficiënte manier geregistreerd worden. Voor deze functies wordt er dan een extra intent voorzien.

Een volgende uitbreiding is mogelijk aan de hand van QnA Maker. De knowledge base die aangesproken wordt door deze software kan op een zeer eenvoudige manier uitgebreid worden. Momenteel wordt QnA Maker enkel gebruikt voor het opvragen van contactinformatie. Het is de bedoeling dat de volledige historiek van problemen en oplossingen binnen Wolters Kluwer ook naar de QnA Maker knowledge base gemigreerd wordt. Deze informatie bevindt zich momenteel op Jira, Sharepoint, en Confluence. Aangezien dit veel vertrouwelijke gegevens bevat, stelde Wolters Kluwer deze niet beschikbaar en werd er voor de proof of concept gewerkt met een eenvoudige knowledge base als voorbeeld.

Los daarvan kan er ook informatie rond de beschikbaarheid van de verschillende teamleden toegevoegd worden. Ten slotte kan een lid van het ondersteuningsteam, wanneer hij of zij merkt dat een vraag erg vaak herhaald wordt, deze toevoegen als ‘frequently asked question’ in de QnA Maker knowledge base.

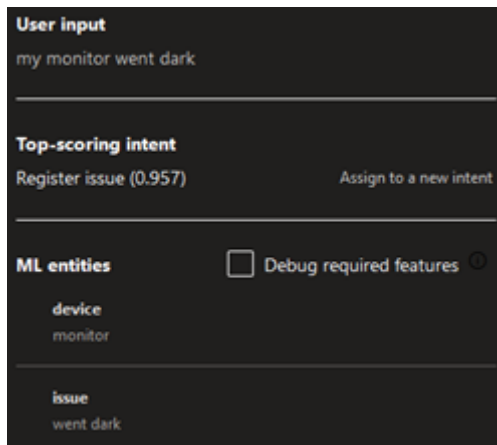
Extra functionaliteiten - LUIS

Aan de hand van Bot Framework Composer is het eenvoudig om een chatbot op te stellen. Deze eenvoud resulteert wel in enkele nadelen. Bot Framework Composer als hulpmiddel voor gebruik van het onderliggend Azure Bot Framework beperkt de functionaliteiten van de bot. Niet alles wat mogelijk is in het onderliggend framework is beschikbaar in Bot Framework Composer. Alle voorstellen die hieronder opgelijst worden vallen onder deze categorie.

Om toch gebruik te kunnen maken van deze functies moet de bot op een andere manier opgebouwd worden, namelijk door het zelf schrijven van de code voor de bot. Hierbij wordt er natuurlijk niet helemaal vanaf nul gestart, er kan nog gebruikt gemaakt worden van het framework dat ook gebruikt wordt door Bot Framework Composer, namelijk Azure Bot Framework.

Ten eerste kan de bot veel intelligenter gemaakt worden, aan de hand van betere natural language processing. Zoals aangehaald in de literatuurstudie, is LUIS in staat om zowel intents als entiteiten te extraheren uit een bericht. Helaas wordt de werking van LUIS beperkt binnen Bot Framework Composer, en is het binnen de software enkel evident om de intent te achterhalen. In beperkte mate is het ook mogelijk om met entiteiten te werken, maar de implementatie hiervan is erg houterig en niet bruikbaar binnen een realistische setting. Wanneer men zelf de bot opstelt aan de hand van code, is dit wel mogelijk. Om dit te verduidelijken, en een concreet gebruik van deze functionaliteiten voor te stellen, wordt er een voorbeeld met bijhorende afbeelding gebruikt.

Binnen het voorbeeld stuurt de gebruiker volgende zin naar de bot: “my monitor went dark”. De intent wordt herkend, dus de chatbot kan de ‘registreer probleem’-dialoog starten. Naast intent zijn er hier ook enkele entiteiten herkend. Er is geweten over welk apparaat het gaat, en daarnaast is er ook een beschrijving van het probleem geïdentificeerd. Deze kunnen al opgeslagen worden als antwoorden binnen het formulier, en er is geen nood meer om de bijhorende vragen te stellen.



Figuur 4.8: Voorbeeld mogelijkheden LUIS, eigen afbeelding

Extra functionaliteiten - tickets

Wanneer er in de proof of concept een ticket voltooid wordt, gebeurt er verder niets mee. In een echte implementatie is dit niet de bedoeling. Hiervoor worden twee oplossingen voorgesteld, beide niet mogelijk wanneer de bot opgesteld wordt aan de hand van Bot Framework Composer, maar wel mogelijk wanneer men de bot zelf opbouwt aan de hand van code. Een eerste voorstel is om een overzicht van het ticket als bericht te sturen op Microsoft Teams naar een van de leden van het ondersteuningsteam. Dit is de meest eenvoudige manier om de informatie naar hen te communiceren, maar is geen goede oplossing op lange termijn. Microsoft Teams is niet voorzien als ticketsysteem. Een complexere maar betere oplossing op lange termijn om aan de hand van de verzamelde informatie een ticket aan te maken binnen Jira. Deze applicatie is ontwikkeld als probleemopvolgingssoftware, en is dus de ideale plaats om deze informatie op te slaan. Communicatie tussen de bot en Jira kan voorzien worden aan de hand van een API voorzien binnen Jira.

5. Conclusie

De doelstelling van deze paper is het verlichten van de last op het ondersteuningsteam aan de hand van een chatbot. Deze chatbot moet enkele eenvoudige taken van het ondersteuningsteam overnemen, en meer complexe taken vereenvoudigen. Dat een chatbot in staat is om die taken te volbrengen wordt aangetoond aan de hand van een proof of concept. Ten slotte moet eenvoudige toegang tot de bot het gebruik ervan motiveren. De proof of concept die binnen deze proef beschreven wordt bewijst dat een chatbot hiertoe in staat is. Eenvoudige problemen kunnen opgelost worden door deze op te lijsten in de vorm van vragen en antwoorden binnen QnA Maker. Voor meer complexe problemen verzamelt de proof of concept alle benodigde informatie, opdat een lid van het ondersteuningsteam dit vlot kan afhandelen. Implementatie van de bot binnen een gekend platform zoals Microsoft Teams vergemakkelijkt de toegang tot de bot, en motiveert zo het gebruik ervan. Ten slotte zorgt een correct getraind natural language processing model ervoor dat de proof of concept in staat is om de bedoeling van de gebruiker vlot en correct te interpreteren, wat opnieuw het gebruiksgemak verhoogt.

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

Deze bachelorproef probeert op basis van een chatbot, ondersteund met artificiële intelligentie een oplossing te bieden voor een probleemcasus binnen het bedrijf Wolters Kluwer. Er bestaat namelijk een zeer grote knowledge base met informatie omtrent veelvoorkomende problemen, en oplossingen hiervoor, maar deze is niet makkelijk in gebruik. Hierdoor wordt het ondersteuningsteam vaak opgeroepen door het ontwikkelingsteam, maar deze heeft de capaciteit niet om al deze aanvragen correct en tijdig te verwerken. De bedoeling is om dit ondersteuningsteam te vervangen door een chatbot als link tussen het ontwikkelingsteam en de knowledge base. Hierdoor is het ondersteuningsteam vrij om echte problemen op te lossen. Naast het bieden van een oplossing wordt er ook onderzocht hoe deze kan geïmplementeerd worden, en of dit alternatief weldegelijk voor een efficiënter verloop zorgt.

A.2 State-of-the-art

Voorbeelden van concrete chatbots zijn moeilijk te vinden, aangezien de implementatie erg verschilt van toepassing tot toepassing, en deze ook vaak achter een patent zitten. Tegenwoordig zijn er al heel wat chatbots in gebruik, al dan niet gebaseerd op artificiële intelligentie. Het gebruik hiervan wordt ook naar steeds bredere sectoren uitgebreid. Deze

proef beperkt zich tot een sector die erg gelinkt is aan technologie, namelijk een software-ontwikkelingsbedrijf, maar tegenwoordig is er ook al onderzoek om dit bij voorbeeld ook in de medische sector toe te passen (Divya S., 2018), en worden deze principes vandaag al toegepast om nieuwe werknemers te verwerven in human resources (Nishad Nawaz, 2020). Deze ‘bots’ slagen er meer en meer in om menselijk over te komen, hiermee wordt bedoeld dat het onderscheid tussen een gesprek met een chatbot en een echte persoon minder merkbaar wordt (Hill & Ford, 2015). Ook is er ondertussen al bewezen dat, indien er niet specifiek vermeld wordt dat de persoon met een robot aan de praat is, deze even efficiënt kunnen zijn als een gesprek met een getrainde persoon (Luo & Tong, 2019). Ten slotte wordt er niet enkel onderzoek gedaan naar waar deze chatbots kunnen toegepast worden, en hun efficiëntie, maar ook hoe ze werken en op welke manier we dit nog kunnen verbeteren. Zo blijken dit de drie belangrijkste pijlers te zijn in de meest recente. Ten eerste is het belangrijk dat een chatbot conclusies kan trekken over een gebruiker op vlak van meerdere datapunten. Ten tweede moet een chatbot de kennis van de gebruiker correct toepassen om zo een meer gepersonaliseerde oplossing te bieden. Ten derde blijkt er een sterk verband te bestaan tussen het analytisch vermogen van de bot en de capaciteit om op het juiste moment te interageren met de gebruiker (Eleonora Pantano, 2020).

Zoals hiervoor vermeld is de informatie omtrent chatbots voor support binnen een bedrijf erg beperkt. Toch werden er al enkele user studies gedaan (Dario Fiore, 2019). Uit dit beperkt onderzoek blijkt dat dit zeer nuttige toepassing is van een chatbot. De eenvoudigheid, proactieve aanpak en vlotheid van zo een bot zijn de grootste troeven. Op vlak van chatbots buiten een bedrijf, zoals voor ondersteuning van klanten, is er wel al meer onderzoek gedaan (Nguyen, 2019). In grote lijn zijn deze bots ook erg nuttig, wel wordt er ondervonden dat het potentieel hiervan niet overschat mag worden. Er kruipt namelijk erg veel tijd in, niet enkel in het opzetten maar ook in het onderhoud. Daarnaast kunnen bots enkel nog maar de eenvoudigste taken aan, er is nog altijd nood aan een achterliggend, menselijk team voor ondersteuning voor moeilijkere taken.

A.3 Methodologie

Ten eerste zal er een uitgestrekte analyse worden gemaakt van het systeem dat nu in plaats is. Dit opdat de verschillende taken van het ondersteuningsteam kunnen opgesplitst worden in twee groepen, namelijk een groep met eenvoudige vragen die de chatbot kan beantwoorden, en een groep met moeilijkere vragen die dan doorverwezen kunnen worden naar het menselijke ondersteuningsteam. Hiernaast zal deze analyse ook als een baseline functioneren, waartegen we dan onze latere resultaten kunnen pijlen.

Als tweede wordt de chatbot opgezet. Dit gebeurt via het Google Dialogflow platform. Google Dialogflow ondersteunt namelijk Nederlands als taal, biedt een gratis proefperiode aan en wordt geadverteerd als een chatbot voor ondersteuning.

Vervolgens wordt deze chatbot dan getraind. Hiervoor is er al voldoende informatie aanwezig. Binnen het bedrijf wordt er namelijk gebruikt gemaakt van het ontwikkelingssysteem ‘Jira’. In deze software bestaat er al een historiek van vorige problemen en een beschrijving van hoe deze opgelost zijn. Ook is deze software bereikbaar via een API, wat de integratie erg kan vereenvoudigen. Daarnaast zal de knowledge base ook een belangrijke bron van trainingsdata vormen. Deze bestaat uit Jira, Confluence en Sharepoint. Deze knowledge

base is erg gefragmenteerd, dus er wordt ook gekeken naar een oplossing om deze te verenigen en vereenvoudigen. Naast het leren van hoe de chatbot een probleem oplost, moet de bot ook leren wat het moet doen indien het probleem doorverwezen wordt naar een techniker. Zo moet de bot onder andere een ticket aanmaken binnen Jira, met alle verzamelde informatie uit het chatgesprek.

Hierna wordt de chatbot getest. Indien mogelijk is het ook de bedoeling om deze te integreren in Microsoft Teams. Dit is namelijk een platform dat al sterk gebruikt wordt binnen het bedrijf, en is erg toegankelijk. Het gaat hier over een proof of concept, en geen volledige implementatie. De tests vinden plaats binnen een gecontroleerde omgeving, en slechts met een beperkt aantal proefpersonen.

Ten slotte worden deze tests vergeleken met de baseline die opgesteld werd in de eerste stap. Hieruit wordt er dan afgeleid of deze chatbot al dan niet een meerwaarde heeft voor het bedrijf.

A.4 Verwachte resultaten

Als resultaat wordt er een chatbot opgeleverd, op basis van Google Dialogflow en getraind aan de hand van data die al aanwezig is binnen het bedrijf. Er wordt verwacht dat deze chatbot grotendeels zelfstandig kan aanleren welke ondersteuning aangeboden moet worden. Daarnaast moet de bot ook kunnen inschatten wanneer een probleem zelf op te lossen, en wanneer door te verwijzen naar een menselijk lid van het ondersteuningsteam. Ook wordt er verwacht dat er, aan de hand van de baselines, bewezen wordt dat de chatbot voor een efficiënter verloop van ondersteuning zorgt. Dit aan de hand van een proof of concept, die een eerste concrete implementatie van de chatbot vormt, en kan bewijzen dat het weldegelijk een meerwaarde voor het bedrijf betekent.

A.5 Verwachte conclusies

De verwachte conclusie is dat deze bot dus weldegelijk een goed alternatief biedt voor het ondersteuningsteam. De last op hen wordt verlicht en belangrijkere taken worden sneller voltooid. Daarnaast wordt er verwacht dat deze chatbot ook een stuk efficiënter is, dit niet enkel op vlak van onderhoudskosten, maar ook op vlak van operatiekosten. Zo zou deze chatbot niet langer enkel een proof of concept zijn, maar zou het ook echt kunnen geïmplementeerd worden. Wolters Kluwer zou dan het abonnement binnen Google Dialogflow verlengen, en de reeds getrainde bot verder implementeren en integreren in hun systemen.

Bibliografie

- Abduljabbar, R., Dia, H., Liyanage, S., & Bagloee, S. A. (2019). Applications of Artificial Intelligence in Transport: An Overview. *MDPI*. Verkregen 26 mei 2022, van <https://ieeexplore.ieee.org/abstract/document/8956460>
- Bahrammizadeh, A. (2010). A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems. *Springer*, 1165–1195. Verkregen 11 mei 2022, van <https://link.springer.com/article/10.1007/s00521-010-0362-z>
- Buchanan, B. G. (2006, november 4). *A (Very)Brief History of Artificial Intelligence*. International Journal of Computer Sciences en Engineering. Verkregen 30 maart 2022, van https://www.researchgate.net/profile/Menal-Dahiya/publication/321864990_A_Tool_of_Conversation_Chatbot/links/5a360b02aca27247eddea031/A-Tool-of-Conversation-Chatbot.pdf
- Chowdhary, K. R. (2020). Natural Language Processing. *Springer*, 603–649. Verkregen 30 maart 2022, van https://link.springer.com/chapter/10.1007/978-81-322-3972-7_19
- Dario Fiore, M. B. (2019). "Forgot your password again?": acceptance and user experience of a chatbot for in-company IT support. *ACM Digital Library*. Verkregen 14 februari 2022, van <https://dl.acm.org/doi/abs/10.1145/3365610.3365617>
- Divya S., I. V. (2018). A Self-Diagnosis Medical Chatbot Using Artificial Intelligence. *Journal of Web Development and Web Designing*, 3. Verkregen 14 december 2020, van https://core.ac.uk/display/230494941?utm_source=pdf&utm_medium=banner&utm_campaign=pdf-decoration-v1
- Eleonora Pantano, G. P. (2020). Forecasting artificial intelligence on online customer assistance: Evidence from chatbot patents analysis. *Science Direct*, 55. Verkregen 14 december 2020, van <https://www.sciencedirect.com/science/article/abs/pii/S0969698919311865>

- French, R. M. (1990). Subcognition and the Limits of the Turing Test. *JSTOR*, 99(393), 53–65. Verkregen 28 maart 2022, van <https://www.jstor.org/stable/2254890>
- Gamborino, E., Yueh, H.-P., Lin, W., Yueh, S.-L., & Fu, L.-C. (2020). Mood Estimation as a Social Profile Predictor in an Autonomous, Multi-Session, Emotional Support Robot for Children. *IEEE*, 11(1). Verkregen 11 mei 2022, van <https://www.mdpi.com/2071-1050/11/1/189>
- Hamlet, P., & Tremblay, J. (2017). Artificial intelligence in medicine. *Science Direct*, 69, S36–S40. Verkregen 11 mei 2022, van <https://www.sciencedirect.com/science/article/abs/pii/S002604951730015X>
- Hill, J., & Ford, W. R. (2015). Real conversations with artificial intelligence: A comparison between human–human online conversations and human–chatbot conversations. *Science Direct*, 49, 245–250. Verkregen 14 december 2020, van <https://www.sciencedirect.com/science/article/abs/pii/S0747563215001247>
- Holland, J. H. (1992). Genetic Algorithms. *JSTOR*, 167(1), 66–73. Verkregen 30 maart 2022, van <https://www.jstor.org/stable/24939139>
- Holzinger, A., Langs, G., Denk, H., Zatloukal, K., & Müller, H. (2019). Causability and explainability of artificial intelligence in medicine. *Wiley*. Verkregen 11 mei 2022, van <https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1312>
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, 170, 436–444. Verkregen 30 maart 2022, van <https://www.nature.com/articles/nature14539>
- Luo, X., & Tong, S. (2019). Frontiers: Machines vs. Humans: The Impact of Artificial Intelligence Chatbot Disclosure on Customer Purchases. *PubsOnLine - Informs*, 38(6). Verkregen 14 december 2021, van <https://pubsonline.informs.org/doi/abs/10.1287/mksc.2019.1192>
- Mahesh, B. (2020). Machine Learning Algorithms - A Review. *International Journal of Science and Research*. Verkregen 30 maart 2022, van https://www.researchgate.net/profile/Batta-Mahesh/publication/344717762_Machine_Learning_Algorithms_-_A_Review/links/5f8b2365299bf1b53e2d243a/Machine-Learning-Algorithms-A-Review.pdf
- Müller, A. C., & Guido, S. (2018, oktober 19). *Introduction to machine learning with Python*.
- Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2005). Natural language processing: an introduction. *Jamia*, 18, 544–551. Verkregen 28 maart 2022, van <https://www.jstor.org/stable/4319091>
- Nguyen, T. (2019). Potential effects of chatbot technology on customer support: A case study. *Aalto University Learning Centre*. Verkregen 14 februari 2022, van <https://aaltodoc.aalto.fi/handle/123456789/38921>
- Nishad Nawaz, A. M. G. (2020). Artificial Intelligence Chatbots are New Recruiters. *SSRN*, 10(9). Verkregen 14 december 2020, van https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3521915
- Spector, L. (2006). Artificial Intelligence. *Science Direct*, 170, 1251–1253. Verkregen 28 maart 2022, van <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.482.1549&rep=rep1&type=pdf>