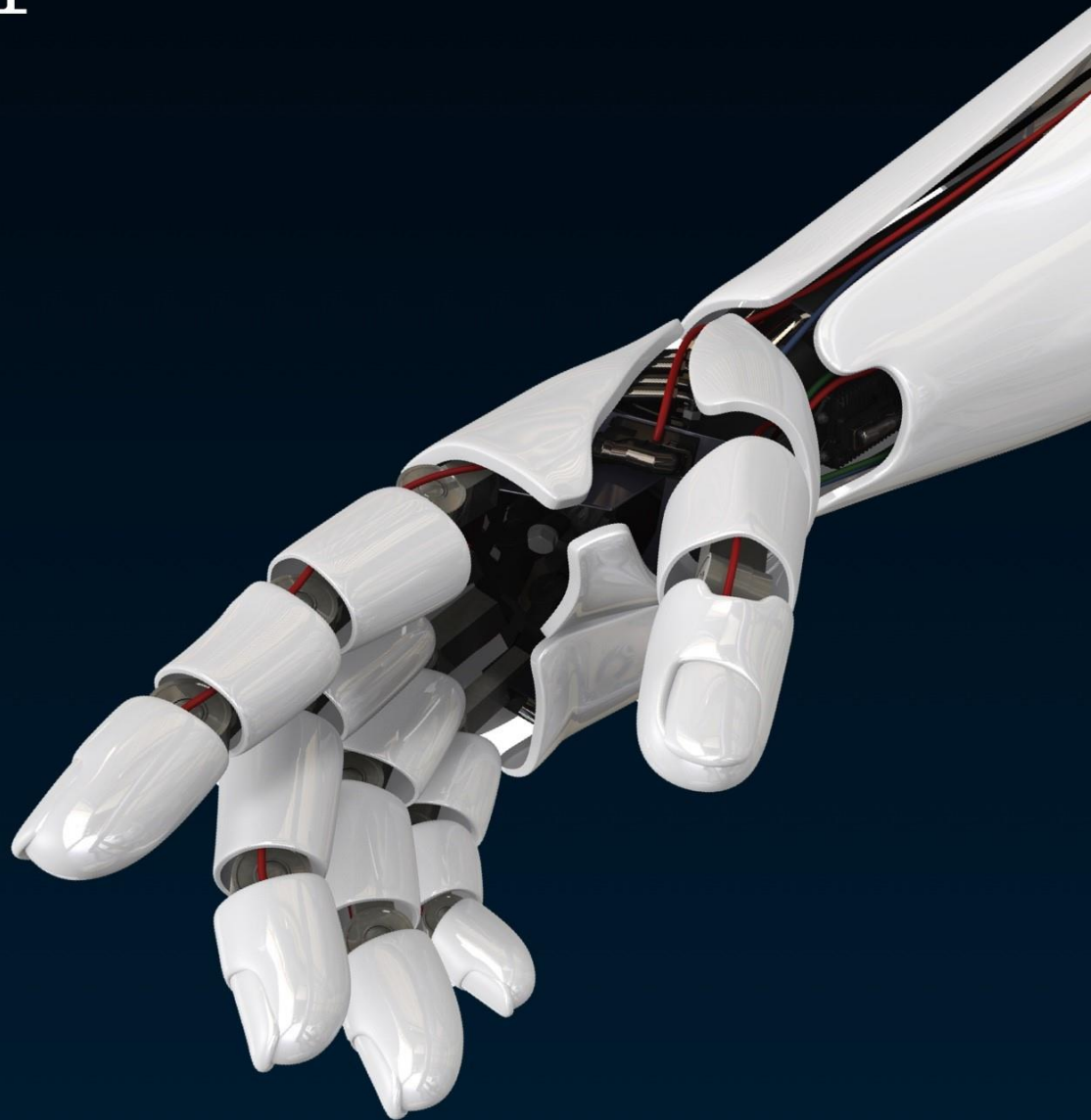


Smart Robotics

De spraak-herkenning van Pepper



Auteur: Ahmed Tyghri
Bedrijfsbegeleider: Leon Emmen
Stagebegeleider: Henk Middendorp

SMART ROBOTICS - DE SPRAAKHERKENNING VAN
PEPPER

AUTEUR(S) : Ahmed Tyghri
TELEFOONNUMMER : +31 6 134 85 294
DOCUMENTNUMMER : 2018-AT-HVA
VERSIE : 1.1
STATUS : Definitief
DOCUMENTDATUM : 10 augustus 2018
AANTAL PAGINA'S : 50

STAGEBEGELEIDER : Henk Middendorp
BEDRIJFSBEGELEIDER : Leon Emmen
TELEFOONNUMMER : +31 6 478 18 810
LOOPBAAN BEGELLEIDER : Merijn van der Laag

ONDERWIJSINSTELLING : Hogeschool van Amsterdam
STUDIERICHTING : Informatica, Software Engineering (SE)
STUDENTENNUMMER : 500702266

BEDRIJF : Atos Amstelveen
ADRES : Burgemeester Rijnderslaan 30
1185 MC, Amstelveen

STARTDATUM : 15-2-2018
EINDDATUM : 15-7-2018

Voorwoord

Voor u ligt de scriptie 'Smart Robotics – De spraakherkenning van Pepper.' Deze scriptie is geschreven in het kader van mijn afstuderen aan de opleiding Software Engineering aan de Hogeschool van Amsterdam en in opdracht van stagebedrijf Atos. Van februari 2018 tot en met juli 2018 ben ik bezig geweest met het onderzoek en het schrijven van de scriptie.

Samen met mijn bedrijfsbegeleider, Leon Emmen, heb ik de onderzoeksvraag voor deze scriptie bedacht. Het onderzoek dat ik heb uitgevoerd was complex. Na het uitvoeren van literatuur-, experimenteel- en exploratief onderzoek heb ik de onderzoeksvraag kunnen beantwoorden.

Tijdens dit onderzoek stonden mijn bedrijfsbegeleider, Leon Emmen, en mijn begeleider vanuit mijn opleiding, Henk Middendorp, altijd voor mij klaar. Zij hebben steeds mijn vragen beantwoord waardoor ik verder kon met mijn onderzoek. Bij dezen wil ik ze graag bedanken voor de fijne begeleiding en hun ondersteuning tijdens dit traject.

Tevens wil ik mijn collega's bij Atos graag bedanken voor de fijne samenwerking. Tom Verloop en Joost van Dalen in het speciaal voor de technische ondersteuning. Ik heb vaak met hen op effectieve wijze kunnen sparren over mijn onderzoek.

Ook van mijn vrienden en familie heb ik wijze raad mogen ontvangen. Bovendien hebben zij mij moreel ondersteund tijdens het schrijfproces.

Ik wens u veel leesplezier toe.

Ahmed Tyghri

Amsterdam, 29 juli 2018

Samenvatting

Robots spelen in de samenleving een steeds grotere rol. Veel bedrijven houden deze ontwikkeling in de gaten, waaronder Atos. Atos wil dat de door hen aangeschafte humanoïde robot Pepper zelfstandig een verscheidenheid aan vragen kan beantwoorden op een zo natuurlijk mogelijke manier. Momenteel kunnen er met Pepper alleen gescripte gesprekken gevoerd worden.

Het doel van deze afstudeeropdracht is onderzoeken hoe Pepper zelfstandig kan functioneren als assistent om vragen van bezoekers en medewerkers te beantwoorden. Met deze opdracht zullen de eerste stappen worden gezet om Pepper zelfstandig te laten converseren met een hoge mate van intelligentie. Hiervoor is de volgende onderzoeksvraag opgesteld: Hoe kan er op Pepper spraak worden omgezet naar tekst en daarvan de context worden bepaald?

Om een antwoord te kunnen geven op de onderzoeksvraag is er literatuur-, experimenteel- en exploratief onderzoek uitgevoerd.

Het onderzoek start met de werking van Pepper. De robot is voorzien van een aantal sensoren, waarvan de twee voorste richtingsgevoelige microfoons belangrijk zijn. De realtime audiodata van deze microfoons zijn noodzakelijk om spraak om te zetten naar tekst.

Pepper heeft zelf geen vrije spraakherkenning. Wat dit betekent is dat de woorden exact moeten matchen om gedetecteerd te worden. Hiervoor dient de oplossing zelf geprogrammeerd te worden door gebruik te maken van een externe spraakherkenning systeem.

De technische uitdagingen hiervan zijn dat de realtime audiodata niet kan worden verkregen, het servicemanagement niet goed werkt en de CPU niet de kracht heeft complexe services uit te voeren. Met Robot Operating System (ROS) worden de beperkingen grotendeels opgelost. ROS staat geïnstalleerd op een server die vervolgens weer verbonden is met Pepper. Via de server kan de realtime audiodata worden verkregen.

Er moet rekening worden gehouden met de omgeving waarin Pepper zich bevindt, omdat ruis de performance van de spraakherkenning kan belemmeren. Een noise gate zal gebruikt worden om geluiden onder bepaald geluidsniveau niet constant mee te nemen in de externe spraakherkenning.

Een systeem voor externe spraakherkenning zal aan de volgende eisen moeten voldoen: het moet een Continuous Speech Recognition systeem zijn, het moet een lage Word Error Rate hebben, spraak moet realtime worden verwerkt, het moet voorzien zijn van een grote woordenlijst, de Nederlandse en/of Engelse taalmodule moet aanwezig zijn en het moet worden geprogrammeerd in ROS. De spraakherkenningen die het beste naar voren zijn gekomen zijn Google Cloud en IBM Watson. Ze moeten beide als een socket worden geprogrammeerd voor een snelle responsetijd. Dit is op het moment alleen gelukt met IBM Watson.

Tot slot om de context te kunnen bepalen is er Natural Language Processing (NLP) nodig. Bij NLP gaat het om taal begrijpen, verwerken en genereren net als een persoon. Spraakherkenning is daar onderdeel van. Voor het begrijpen van de context is een dataset nodig, waar de verkregen tekst op gebaseerd kan worden. Hiervoor is IBM Assistant gekozen om het systeem te kunnen trainen met een dataset. IBM Assistant zal de bedoeling van de zin bepalen en de specifieke waardes uit de zin halen. Om een gebruiker door een proces te helpen kan de dialoog worden gemaakt in IBM Assistant. Het grote verschil met Pepper hiermee is dat woorden niet meer exact hoeven worden gezegd om een dialoog te kunnen voeren.

Op basis hiervan wordt aanbevolen om aan een basis platform te werken voor Pepper waar gemakkelijke andere services en diensten op aan kunnen worden gesloten. De intelligentie van Pepper wordt dan hiermee vergroot om in de toekomst in te kunnen zetten als zelfstandig vraagbak en assistent.

Inhoud

Voorwoord	3
Samenvatting	4
1 Inleiding	7
2 Opdracht.....	10
2.1 Opdrachtdefinitie.....	10
2.2 Producten.....	10
2.3 Uitdagingen	10
3 Onderzoeksmethode	11
4 Onderzoek.....	12
4.1 Pepper	12
4.2 Beperkingen	15
4.3 Robot Operating System (ROS).....	17
4.4 Spraakherkenning	20
4.5 Contextbepaling	27
5 Realisatie	29
5.1 Request	29
5.2 Socket	32
6 Resultaten.....	33
6.1 Prototype 1 - Request.....	33
6.2 Prototype 2 - Socket.....	33
7 Conclusie	34
8 Aanbeveling	35
Verklarende woordenlijst.....	36
Bronnenlijst.....	37
Figurenlijst	39
Bijlage A: Smart Robotics Project.....	40
Bijlage B: Interview.....	42
Bijlage C: Nodes & Topics	44
Bijlage D: Omgevingsfactoren	46
Bijlage E: Benchmarks.....	47
Bijlage F: Metingen	49

Wijzigingsbladen

Versie	Datum	Omschrijving
0.1	18-03-2018	Template
0.2	22-04-2018	Conceptversie
0.3	01-05-2018	Context Bijlage A: Smart Robotics Project
0.4	05-05-2018	Opdrachtdefinitie
0.5	06-06-2018	Onderzoeksmethode
0.6	12-06-2018	Voorblad 4.1 Pepper 4.2 Beperkingen
0.7	18-06-2018	4.3 Robot Operating System
0.8	02-7-2018	Feedback verwerken bedrijfs- en stagebegeleider Bijlage B: Interview Bijlage C: Nodes & Topics
0.9	20-7-2018	Bijlage D: Omgevingsfactoren Bijlage E: Benchmarks Bijlage F: Metingen 4.4 Spraakherkenning
1.0	29-7-2018	4.5 Contextbepaling Resultaten Conclusie Aanbeveling Samenvatting Voorwoord
1.1	10-8-2018	Feedback verwerken Velden bijwerken Spelling & grammatica

Tabel 1: Versiebeheer

1 Inleiding

In dit hoofdstuk wordt de context van de afstudeeropdracht omschreven. Er wordt ingegaan op rol van robots in de samenleving, het bedrijf waarvoor de afstudeeropdracht wordt uitgevoerd en de aanleiding van de opdracht.

Robots in de samenleving

In de afgelopen jaren zien we robots steeds meer in de samenleving verschijnen. Eenvoudige en eenzijdige taken kunnen vaak goedkoper uitgevoerd worden door robots dan door mensen, zoals in massaproducties. Robots kunnen op plaatsen komen die voor mensen moeilijk te bereiken zijn. Wanneer een taak te vies, gevaarlijk of fysiek onmogelijk is voor een mens, dan wordt een robot ingezet. Denk hierbij aan in de zorg of huishoudens of het opruimen van explosieven. Robots kunnen complexe taken vaak nauwkeuriger uitvoeren, met een beter resultaat als gevolg. De ontwikkeling van robotica gaat snel en wordt op steeds nieuwe gebieden ingezet (Balideli, 2014).

Het effect hiervan is dat robots intelligenter worden en fysiek meer op mensen gaan lijken. Een recent voorbeeld van deze ontwikkeling is Sophia. In Oktober 2017 werd de humanoïde robot Sophia een Saoedi-Arabisch burger. Het is de eerste robot die een burgerschap ontving in de wereld (Sophia, 2017). Alhoewel dit puur symbolisch was, is het een teken dat robots een steeds beter gedefinieerde plek gaan krijgen in onze samenleving.

De vraag is of dit een goede ontwikkeling is of dat men zich zorgen moet maken dat robots banen overnemen of zelfs de mensheid uitroeien. Stephen Hawking waarschuwde jaren terug al over de gevaren van robots die steeds intelligenter beginnen te worden (Cellan-Jones, 2014):

"The development of full artificial intelligence could spell the end of the human race....It would take off on its own, and re-design itself at an ever increasing rate. Humans, who are limited by slow biological evolution, couldn't compete, and would be superseded."

Deze gedachtegang ontstaat vaak door films, bijvoorbeeld in de film The Terminator, waarin een kwaadaardige robot de mensheid uit wil roeien. Het is zeer onwaarschijnlijk dat dit nog gaat gebeuren (Hawkins, 2015). Naast de snelle ontwikkeling kost het veel tijd om wetten en denkkaders te veranderen om robots daadwerkelijk intelligent te maken en uiteindelijk in te kunnen zetten (Balideli, 2014).

Iemand die zich al in de jaren veertig hiermee bezighield was Isaac Asimov. Hij schreef over robots in een serie van negen korte sciencefictionverhalen genaamd I, Robot. In deze verhalen werden voor het eerst de drie robotica wetten gepresenteerd die door Asimov zijn bedacht:

- ▶ "A robot may not injure a human being or, through inaction, allow a human being to come to harm."
- ▶ "A robot must obey the orders given it by human beings except where such orders would conflict with the First Law."
- ▶ "A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws."

Deze wetten hebben een grote invloed op het denken hoe robots zich in verhouding tot mensen moeten gedragen. Het is belangrijk om goed na te denken over wat robots en kunstmatige intelligentie voor ons mensen gaat betekenen. Deze ontwikkeling is namelijk niet te stoppen.

"The robots are coming, whether we like it or not, and will change our economy in dramatic ways." - (Anderson, n.d.)

Atos

Het is dus niet verassend dat veel organisaties de ontwikkeling van robotica in de gaten houden, waaronder Atos. Atos is een Frans IT-bedrijf dat zich bezighoudt met systeemontwikkeling en consultancy. Door een samenvoeging van twee Franse IT-bedrijven is Atos opgericht in 1997. In 2000 is Atos samengegaan met het Nederlandse bedrijf Origin B.V.

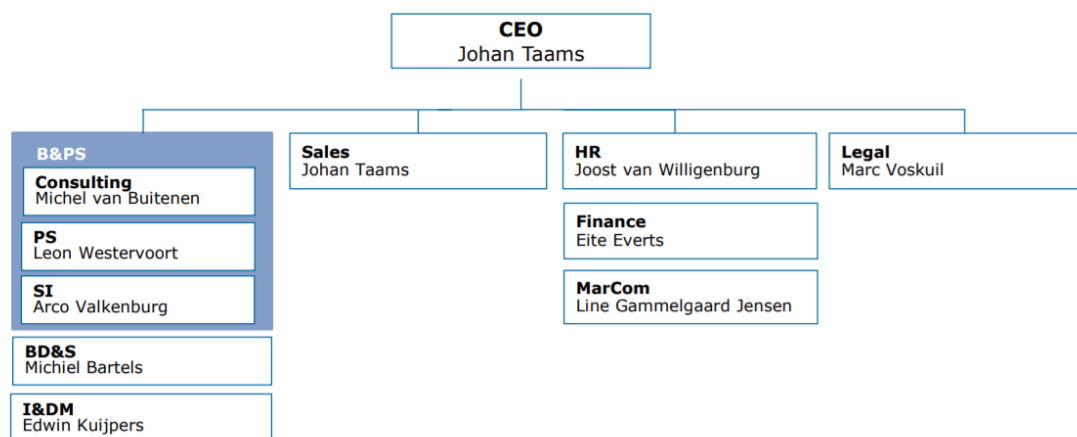
Atos heeft een jaaromzet van 13 miljard euro en telt 100.000 medewerkers in 73 landen. Het is de Europese leider in Big data, Cybersecurity, High Performance Computing en Digital Workplace Solutions. Atos biedt aan haar klanten Clouddiensten, Infrastructuur en datamanagement, Business & platform oplossingen, en transactiediensten via Worldline, de Europese leider in de markt voor betaaloplossingen en transactie diensten.

Met innovatieve technologieën, digitale expertise en kennis van industriële sectoren ondersteunt Atos de digitale transformatie van klanten in de volgende marktsectoren: Defensie, Financiële dienstverlening, Gezondheidszorg, Manufacturing, Media en Nutsvoorzieningen, Publieke Sector, Retail, Telecom en Transport.

Atos is de wereldwijde IT-partner voor de Olympische Spelen en Paralympische Spelen en opereert onder de namen Atos, Atos Consulting, Atos Worldgrid, Bull, Canopy, Unify en Worldline. Atos SE (Societas Europaea) staat genoteerd aan de CAC40 Paris stock index (Company Information, n.d.).

Organisatiestructuur

Atos heeft 10 Business Units wereldwijd. Nederland valt onder de businessunit Benelux & The Nordics (BTN) samen met België, Luxemburg, Denemarken, Zweden, Finland, Litouwen, Polen, Rusland en Estland. Atos wordt onderverdeeld in vier subdivisies: Infrastructuur & Data Management (IDM), Business & Platform Solutions (B&PS), Big Data & Cybersecurity(BDS) en Unified Communication & Collaboration (UCC). Naast de hoofddivisies zijn er ook de Support Functions: Sales, HR, Finance, Marketing & Communications en Legal. De CEO van Atos BTN is Peter 't Jong en de CEO van Atos Nederland is Johan Taams (Organization, n.d.).



Figuur 1: Organogram Atos Nederland

Technology Lab

In het Technology Lab van Atos wordt er gewerkt aan innovatieve oplossingen op basis van nieuwe technologieën. Er wordt geëxperimenteerd om te bepalen of deze innovatieve oplossingen kunnen uitgroeien tot potentiële businessmodellen. In totaal zijn er drie Technology Labs in Nederland. Het eerste Technology Lab is opgericht op 1 november 2015 in Amstelveen. In het Technology Lab in Amstelveen wordt er momenteel gewerkt met technologieën als Big Data Analytics, Internet of Things, Blockchain, Robotics en Virtual Reality/Extended Reality.

Aanleiding

Het Smart Robotics project binnen Atos werd gestart op 1 januari 2016. Atos heeft eerste 2 kleine Nao robots aangeschaft en later de humanoïde robot Pepper om bezoekers op een innovatieve manier te verrassen. De bedoeling is dat Pepper autonoom gesprekken kan voeren met medewerkers en bezoekers en taken van de service desk, zoals bijvoorbeeld het geven van locatie informatie, kan overnemen.

Pepper is out of the box voorzien van een aantal bijzondere features, zoals lichaam beweging controles, spraakherkenning, geluid lokalisatie, gezichtsherkenning, beeldherkenning en een emotion engine. Atos wil de functies van de robot flink uitbreiden, waarin met name de intelligentie van de robot centraal staat. Momenteel kunnen er alleen gescipte gesprekken met Pepper gevoerd worden. Om uiteindelijk autonoom te kunnen converseren, dienen nieuwe oplossingen geprogrammeerd te worden op basis van nieuwe technologieën¹.

Opdrachtoomschrijving

Met Pepper kunnen er momenteel alleen gescipte gesprekken gevoerd worden. Atos wil dat Pepper zelfstandig een verscheidenheid aan vragen kan beantwoorden op een zo natuurlijk mogelijke manier. Het doel van deze afstudeeropdracht is onderzoeken hoe Pepper zelfstandig kan functioneren als assistent om vragen van bezoekers en medewerkers te beantwoorden.

Met deze opdracht zullen de eerste stappen worden gezet om Pepper zelfstandig te laten converseren met een hoge mate van intelligentie. Om dit te bereiken, moet er aan de achterkant een systeem draaien wat kan analyseren wat er tegen Pepper wordt gezegd, om vervolgens de context te bepalen en een passend antwoord te geven.

Hiervoor moet Pepper eerst spraak kunnen omzetten naar tekst. In de scope van deze opdracht valt dan ook met name het onderwerp spraakherkenning. Dit is een technisch complex proces, omdat het omzetten van spraak naar tekst real time plaats moet vinden. Op basis van deze opdracht kunnen dan vervolgstappen gezet worden in het toevoegen van intelligentie aan de spraakmodule van Pepper.

Rapportstructuur

Voor het uitvoeren van de opdracht zijn er verschillende fases doorlopen. Deze fases zijn vertaald naar hoofdstukken in dit onderzoeksrapport.

De structuur van het onderzoeksrapport is als volgt: hoofdstuk twee geeft de opdracht van het onderzoek weer. De hoofd en deelvragen zijn hierin terug te vinden. In hoofdstuk drie wordt de onderzoeksmethode behandeld waaraan het onderzoek zal worden verantwoord.

Daarna in hoofdstuk vier begint de kern van het onderzoek. Hierin worden de onderwerpen Pepper, Beperkingen, Robot Operating System, Spraakherkenning en Contextbepaling behandeld.

Vervolgens zal in hoofdstuk vijf de realisatie van het onderzoek worden uitgelegd en in hoofdstuk zes de resultaten. In hoofdstuk zeven zal de conclusie van het onderzoek worden getrokken en op basis daarvan zal in hoofdstuk acht een aanbeveling worden gegeven.

Tot slot zijn de verklarende woordenlijst, bronnenlijst, figurenlijst, bronnen en bijlages achterin het onderzoeksrapport te vinden.

¹ In Bijlage A: Smart Robotics Project wordt meer inhoudelijk verteld over het project;

2 Opdracht

In dit hoofdstuk wordt de opdracht van het onderzoek besproken. De onderzoeksvraag en de bijhorende deelvragen worden uitgewerkt. Verder wordt de probleemstelling, producten en uitdagingen beschreven.

2.1 Opdrachtdefinitie

De titel van de opdracht luidt:

- Smart Robotics – De spraakherkenning van Pepper

Onderzoeksvraag

Om de opdracht af te bakenen en het resultaat gemakkelijk te kunnen beoordelen is een hoofdvraag opgesteld voor het onderzoek. Deze luidt:

- Hoe kan er op Pepper spraak worden omgezet naar tekst en daarvan de context worden bepaald?

De volgende deelvragen zijn opgesteld om de bovengenoemde hoofdvraag zo goed mogelijk te kunnen beantwoorden:

1. Hoe zit Pepper in elkaar?
2. Wat zijn de beperkingen van Pepper om zelfstandig als vraagbaak en assistent te kunnen functioneren?
3. Welke oplossingen zijn er voor deze beperkingen?
4. Hoe kan de audiodata worden verkregen van spraak?
5. Hoe kan de spraak worden omgezet naar tekst?
6. Hoe kan de context van een tekst worden bepaald?

2.2 Producten

Het product dat uiteindelijk wordt opgeleverd aan Atos is een eerste aanzet om spraak om te zetten naar tekst en dit in de juiste context te kunnen plaatsen. Dit systeem moet in de toekomst gekoppeld kunnen worden aan intelligente diensten om uiteindelijk uiteenlopende vragen van gebruikers te kunnen beantwoorden. Hier zal een advies over worden gegeven.

2.3 Uitdagingen

Onderzoekers en wetenschappers zijn al jaren bezig met het onderzoeken hoe robots slimmer kunnen worden gemaakt. De ontwikkeling van robotics begint nu ook in de wereld steeds meer op gang te komen, vooral met de opkomst van nieuwe technologieën zoals machine learning en natural language processing.

Atos wil daarom ook achterhalen wat de mogelijkheden zijn van deze nieuwe technologieën op het gebied van robotica. In het Technology lab van Atos is er op het moment een Pepper beschikbaar. Deze Pepper zal niet altijd beschikbaar zijn om erop te kunnen ontwikkelen, omdat het ook wordt ingezet voor demo's en evenementen.

Daarnaast zal gezien de tijd van de afstudeeropdracht het product niet helemaal gedetailleerd kunnen worden ontwikkeld. Waar de tijd vooral aan zal worden besteed is het onderzoeken welke toepassingen nodig zijn om Pepper te voorzien van een goed werkend spraakherkenning. Het uiteindelijk product zal dan ook een prototype zijn.

3 Onderzoeksmethode

In dit hoofdstuk wordt de onderzoeksmethode behandeld die wordt uitgevoerd gedurende de afstudeerperiode. Om antwoord te vinden op de onderzoeksvraag zullen de volgende onderzoeksmethodes gebruikt worden: literatuuronderzoek, experimenteel onderzoek en exploratief onderzoek. Vervolgens wordt het onderzoeksmateriaal besproken.

Literatuuronderzoek

In dit onderzoeksrapport zal er literatuuronderzoek worden gedaan. Door middel van het onderzoeken van officiële documentatie, theorieën, wetenschappelijke artikelen en beeldmateriaal kan een beter beeld worden geschetst wat er allemaal mogelijk en beschikbaar is dit onderzoek.

Experimenteel onderzoek

Naast literatuuronderzoek zal er ook experimenteel onderzoek worden uitgevoerd. In het experimenteel onderzoek wordt een bepaalde variabele gemanipuleerd om het effect ervan te bekijken. De omgeving waarin Pepper zich bevindt en de uitspraak en het spraakvolume van iemand spelen een rol in de kwaliteit van de spraakherkenning. Door deze variabele aan te passen en te vergelijken kan beter worden vastgesteld of Pepper wel goed bij de servicedesk kan worden neergezet.

Exploratief onderzoek

In het exploratief onderzoek gaat het om ideeën op doen en het onderzoeksgebied verkennen voor vervolgonderzoek. Alle mogelijke interessante gegevens worden verzameld. Het onderzoek is ook het begin om Pepper als zelfstandig vraagbak en assistent in te kunnen zetten. Door exploratief onderzoek te doen kan het onderzoeksprobleem beter worden begrepen en kunnen de belangrijke factoren omtrent spraakherkenning worden achterhaald, zodat daar in de toekomst vervolgonderzoek op gedaan kan worden.

Onderzoeksmateriaal

Om belangrijke keuzes te maken in het onderzoek zal er contact worden gelegd met specialisten. Samen met deze personen zal worden besproken welke technieken, achtergrondinformatie of technologieën van toepassing zijn om het systeem te verbeteren.

Hieronder is een lijst van personen waarvan hulp is ingeschakeld om aan het systeem te kunnen werken:

Databron	Naam	Functie/Expertise	Locatie
Personen	Leon Emmen	Automation Lead BTN Business & Platform Solutions	Atos Amstelveen
	Tom Verloop	Technische Informatica	Atos Amstelveen
	Joost van Dalen	Muziek Technologie & Software Engineer	Atos Amstelveen
	Arjen Tuinstra	Data Science Team Manager & Data Scientist	Atos Amstelveen/ Groningen

Tabel 2: Databronnen

4 Onderzoek

In dit hoofdstuk zal het onderzoek worden gestart. Hierin worden de volgende onderwerpen behandeld: Pepper, Beperkingen, Spraakherkenning en contextbepaling.

4.1 Pepper

In dit deel van het onderzoek wordt de eerste deelvraag beantwoord. Hierin wordt beschreven wat Pepper is, wat Pepper bijzonder maakt ten opzichte van andere robots, welke hardware en software Pepper heeft en hoe zelf geprogrammeerd kan worden voor Pepper. Hierbij ligt de focus op de onderdelen die belangrijk zijn voor spraakherkenning.

4.1.1 Over Pepper

Pepper is een humanoïde robot ontwikkeld door het bedrijf Softbank Robotics (voorheen Aldebaran Robotics). De bedoeling van Pepper is om 'mensen gelukkig te maken', het leven van mensen te verbeteren, relaties te vergemakkelijken, plezier te hebben met mensen en mensen te verbinden met de buitenwereld.

Pepper kan daarom ook de emoties en gezichten van mensen herkennen, praten, luisteren en rond bewegen. Wat Pepper dus onderscheid van andere robots is dat het een sociale robot is. Het is hoofdzakelijk bedoeld om te interacteren met mensen.

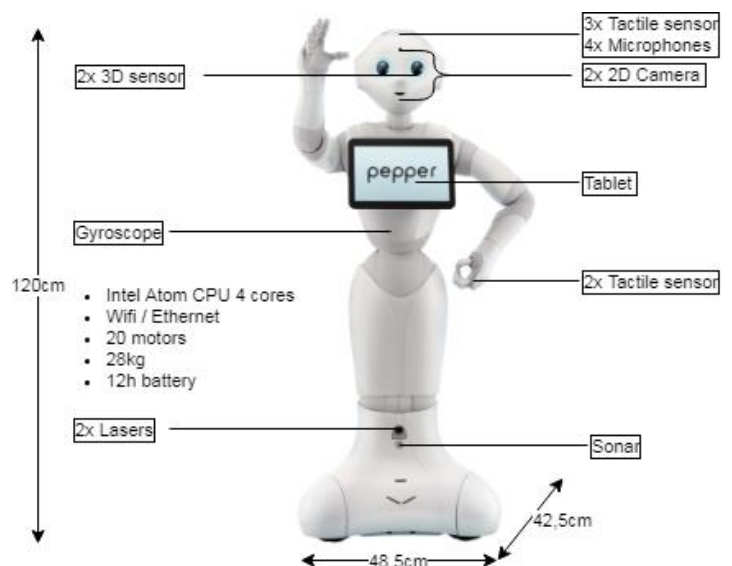
Inmiddels zijn er meer dan 140 Softbank mobiele winkels in Japan die Pepper gebruiken om klanten te verwelkomen, te informeren en te amuseren (Pepper, n.d.).

4.1.2 Hardware

Pepper is een kleine robot. De meeste volwassenen zullen dan vaak omlaag kijken naar Pepper om er tegenaan te praten. In Pepper zitten er allerlei sensoren en een tablet ingebouwd, zie figuur 2.

Sensoren

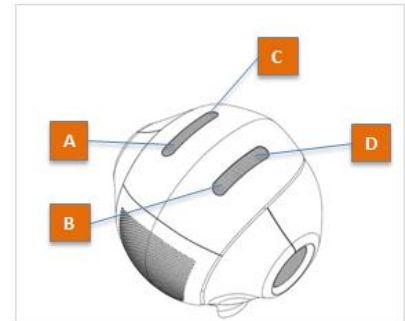
Pepper heeft vier richting gevoelige microfoons, drie aanraaksensoren en drie camera's (twee RGB-camera's en een 3D-camera) in het hoofd. Bij de benen zitten twee ultrasound transmitters en receivers, zes laser sensoren en drie obstakeldetectoren. Deze sensoren geven informatie over de stand van obstakels, zoals tafels en muren. Een ander sensor bij de benen is een temperatuursensor, deze zit naast de batterij. In de borst van Pepper zit een gyroscoop. Verder heeft Pepper ook aanraaksensoren in de handen (Pepper Technical overview, n.d.).



Figuur 2: hardware overzicht

Microfoons

Van al deze sensoren zijn de vier richting gevoelige microfoons belangrijk om spraakherkenning te kunnen gebruiken. In figuur 3 zijn de posities van de microfoons duidelijk te zien (Microphone, n.d.). Dit is goed om te weten, want aan de hand van waar de gebruiker zal staan zal door een van de microfoons de spraak van de gebruiker het beste worden opgenomen. Er wordt vanuit de officiële richtlijnen van Pepper uitgegaan dat een gebruiker voor Pepper zal staan (Soledad, Clément, Sandrine, 2017). Dit houdt in dat de microfoons C & D (zie figuur 3) de beste audio opnamen zullen hebben voor spraakherkenning. Deze audio opnamen of ook wel audiodata is dus belangrijk om te verkrijgen.



Figuur 3: Microfoons

Tablet

Pepper heeft overigens een tablet ingebouwd. Deze tablet wordt gebruikt om bepaalde informatie te laten zien en/of om een user input te verkrijgen. Vaak is te zien dat de functionaliteit van de tablet hierbij leidend is: het is een eenvoudige manier om iemand een keuze te laten maken op de robot. Atos vindt echter dat interactie met de robot zo menselijk mogelijk moet verlopen. De tablet kan hierbij een uitstekend hulpmiddel zijn, maar spraak moet leidend blijven. Zo lang de tablet noodzakelijk is voor input, kan Pepper nooit meer worden dan een fancy informatiezuil.

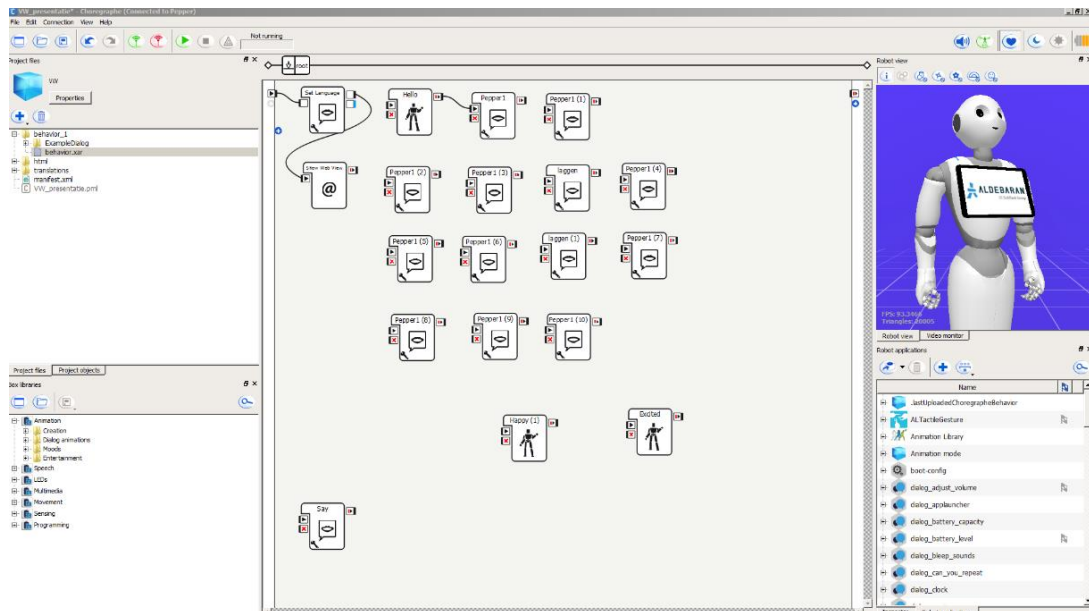
4.1.3 Software

Om gebruik te maken van de ingebouwde hardware is standaard op Pepper de softwareapplicatie Autonomous Life geïnstalleerd. Autonomous Life is verantwoordelijk voor de interactie van Pepper met mensen (Autonomous Life, n.d.). Autonomous life bestaat uit Basic Awareness en Basic Channel. Basic Awareness zorgt ervoor dat Pepper alert is voor de omgeving door met de camera's en microfoons een dichtstbijzijnde persoon te detecteren. Pepper zal dan vervolgens met het hoofd de persoon gaan volgen (Basic Awareness, n.d.). Met Basic Channel kunnen er eenvoudige gescipte dialogen voor Pepper geprogrammeerd worden, zoals bijvoorbeeld reageren op vragen als "hoe het gaat?" (Basic Channel, n.d.).

Ondanks dat Pepper een indrukwekkende robot is door Autonomous Life, heeft het helaas geen positronisch brein of een andere vorm van Artificial Intelligence. Voorlopig zal alle interacties dus gewoon geprogrammeerd moeten worden door ontwikkelaars. Softbank Robotics heeft hiervoor Choregraphe en de Software Development Kit beschikbaar gesteld.

Choregraphe

Choregraphe is een ontwikkeling omgeving specifiek gemaakt voor de NAO-robots. Door een Graphical User Interface (GUI) kunnen functieblokken neer gezet worden en aan elkaar worden verbonden: Een blok is een actie die Pepper kan uitvoeren en door deze blokken aan elkaar te verbinden ontstaat er tijdlijn van opeenvolgende acties die worden uitgevoerd. De gebruiker heeft dus geen kennis nodig van programmeren om de robot iets gemakkelijk te laten uitvoeren, zoals de robot iets laten uitspreken, iets laten zien op de tablet of de robot een animatie laten afspelen (Choregraphe, n.d.). Atos gebruikt Choregraphe om de inzet van Pepper op evenementen voor te bereiden. Zo zorgen ze ervoor dat Pepper bijvoorbeeld een keynote speech op een event kan doen of nieuwe sprekers kan aankondigen.



Figuur 4: Choregraphe

Software Development Kit (SDK)

In plaats van constant afhankelijk te zijn van Choregraphe kunnen scripts zelf worden geprogrammeerd. Het programmeren van eigen scripts zorgt ervoor dat er meer complexe taken kunnen worden uitgevoerd. Hiervoor dient de Software Development Kit (SDK) te worden gebruikt van Softbank Robotics.

NAOqi

Door de SDK kan de NAOqi framework gebruikt worden om te programmeren. Pepper is afhankelijk van NAOqi om überhaupt te kunnen functioneren. NAOqi maakt gebruik van API-calls om de services aan te kunnen roepen. De programmeertalen die hiervoor nodig zijn, zijn Python en C++.

Met Python is het mogelijk om eigen code te laten uitvoeren vanaf een computer naar de robot toe. Python wordt gebruikt om de functionele features van Pepper te ontwikkelen. C++ is een gecompileerde taal. De C++ code moet eerst worden gecompileerd voor het computer operating system (Windows, Linux, Mac) en vervolgens moet deze cross-gecompileerd worden voor het NAOqi operating system, NaoOS) om te kunnen draaien op Pepper. C++ is aan te raden als iets meer low level geprogrammeerd moet worden, bijvoorbeeld het programmeren van functies van de hardware van de robot.

Service

Door het gebruiken van NAOqi worden er services gemaakt. De term service verwijst naar een softwarefunctionaliteit of een set van software functionaliteiten (zoals het ophalen van informatie of het uitvoeren van een reeks bewerkingen) met als doel dat verschillende clients het kunnen hergebruiken voor verschillende doeleinden. Pepper heeft vanuit zichzelf al een heleboel services. Een voorbeeld is de say service waarmee Pepper een tekst kan uitspreken.

Door het gebruiken van zulke services kunnen dus nieuwe functionaliteiten worden gemaakt of zelfs een eigen service worden aangemaakt. De service moet geregistreerd worden aan Pepper om de functionaliteiten ervan te kunnen uitvoeren. De functieblokken in Choregraphe representeren een service, waarbij een variabele zelf gewijzigd kan worden.

4.2 Beperkingen

Omdat Pepper er uitnodigend uit ziet, gaan mensen al snel voor de robot staan en er tegen praten. Er wordt dan niet altijd een antwoord gegeven. Dit komt omdat het geen vrije spraakherkenning heeft. De woorden of zinnen die worden gezegd tegen Pepper moeten exact matchen. Een voorbeeld hiervan is om aan Pepper te vragen zich te introduceren. De robot hoort het woord introduceren en zal vervolgens de bijbehorende dialoog en animatie afspelen. De woordenschat en regels die Pepper kent is daarom ook beperkt.

Hiervoor dient de oplossing zelf geprogrammeerd te worden door gebruik te maken van een bestaande vrije spraakherkenning systeem. In dit hoofdstuk wordt besproken wat hiervan de technische uitdagingen zijn en hoe deze opgelost kunnen worden.

4.2.1 Sensordata

Een van de belangrijkste databronnen voor het implementeren van een spraakherkenningssysteem, is de beschikbare audiodata. Softbank Robotics wil dat het NAOqi framework wordt gebruikt om voor NAO-robots te kunnen ontwikkelen. Het NAOqi framework is niet open source en kan dus niet door andere ontwikkelaars buiten Softbank Robotics worden verbeterd.

In de (beperkte) bijgeleverde documentatie over Pepper staat een lijst van API-calls die kunnen worden gebruikt. Wat hierin mist, is hoe deze inhoudelijk in elkaar zitten en hoe specifieke sensordata uit deze API-calls kan worden verkregen om eigen implementaties te maken. Een van deze API-calls stelt de gebruiker in staat om een recording te maken van wat er gezegd wordt, welke vervolgens opgeslagen wordt op de robot zelf. Voor vrije spraakherkenning is echter realtime audiodata nodig, welke niet beschikbaar is.

Hierdoor kan dus niet de audiodata worden verkregen die nodig is voor de implementatie van vrije spraakherkenning.

4.2.2 Service Management

Een andere beperking is dat Pepper in de praktijk niet goed in staat blijkt om meerdere services naast elkaar te draaien. Het opstarten of willen aanroepen van meerdere services die tegelijkertijd moeten draaien, leidt vaak tot het vastlopen van (een onderdeel) van de robot of het niet correct uitvoeren van een bepaalde taak.

In de praktijk komt het ook voor dat bepaalde services binnen Pepper na uitschakeling weer automatisch worden opgestart. Een voorbeeld hiervan is de eigen dialoog en speech service van Pepper. Deze moeten worden uitgeschakeld om een externe spraakherkenning service te gebruiken zodat niet beide services met elkaar in conflict komen. Echter, het handmatig uitschakelen van deze service leidt tot een reset van de service, waardoor deze zichzelf steeds weer op blijft starten.

Wat tevens ontbreekt is een overzicht van de status van de verschillende services. Er is geen duidelijkheid te krijgen over welke services op welk moment in gebruik zijn, vrij beschikbaar zijn of een afhankelijkheidsrelatie met elkaar hebben.

4.2.3 CPU

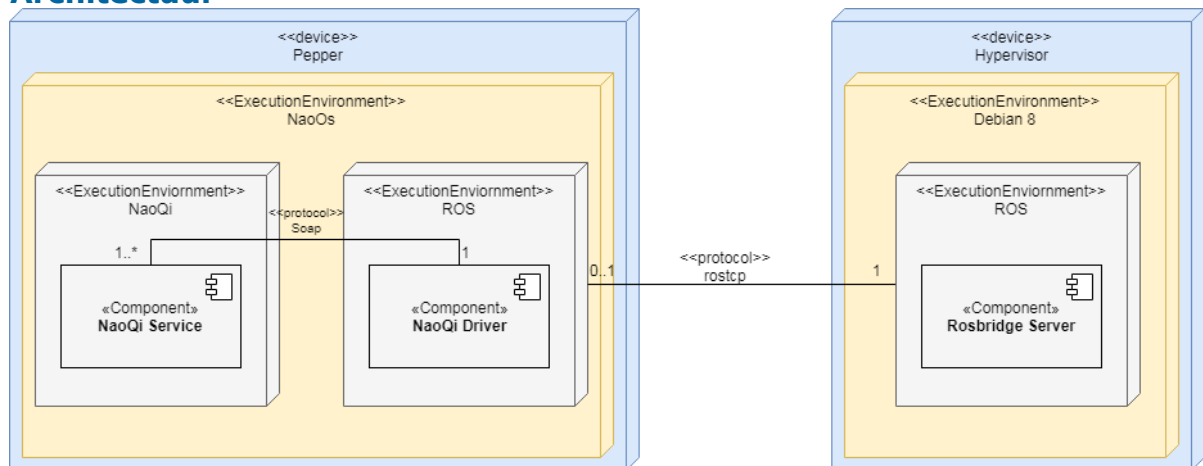
De laatste beperking is de CPU-capaciteit van Pepper. Pepper is niet in staat om zelf al te veel complexe services uit te voeren, omdat de CPU van Pepper niet de kracht heeft om alle berekeningen uit te voeren. De CPU van Pepper is geoptimaliseerd om eigen standaardsoftware en hardware te kunnen aansturen. Voor spraakherkenning moet de audio constant gestreamd worden en realtime worden verwerkt, dit is een bottleneck voor de processing power van Pepper. Het is dus niet verstandig om de service op Pepper zelf te hosten.

4.2.4 Oplossingen

Atos wil een platform hebben wat gebruikt kan worden voor verschillende typen robots. Vooral omdat in de toekomst steeds nieuwe en verbeterde robots uitkomen en het wenselijk is om deze aan te kunnen sluiten op een bestaand platform, zodat reeds ontwikkelde functionaliteit meteen beschikbaar is voor eventuele nieuwe robots. Met dit in gedachten heeft Atos besloten de backend van Pepper en de server helemaal opnieuw in te richten met het Robot Operating System (ROS). Door de structuur van ROS, zoals verder uitgelegd in het volgende hoofdstuk, is het eenvoudiger mogelijk om modulaire services voor robots te bouwen².

Echter, met ROS worden ook de bovenstaande beperkingen grotendeels opgelost. ROS is vanwege zijn structuur in staat om gedetailleerde sensordata uit te lezen, met ROS kan het servicemanagement van Pepper aanzienlijk verbeterd worden en de server waarop ROS draait is voorzien van krachtige CPU's die instaat zijn om complexe software berekeningen uit te voeren voor Pepper.

4.2.5 Architectuur



Figuur 5: Architectuur

Pepper

In het figuur hierboven is te zien hoe de architectuur is ingericht. Op Pepper is ROS geïnstalleerd naast de standaard NaoQi. De NaoQi Driver van ROS handelt de communicatie af met de NaoQi Service. De NaoQi Driver publiceert de beschikbare data van sensoren en actuatoren. Om specifieke sensor data te verkrijgen van Pepper kan de NaoQi Driver worden uitgebreid.

Server

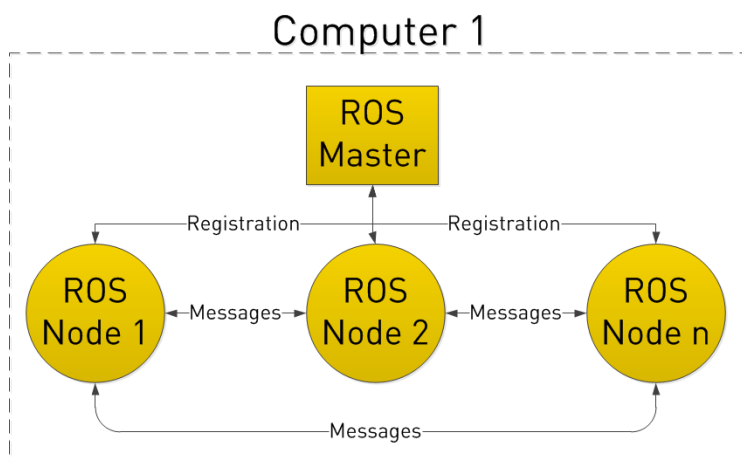
Op de Server is ook ROS geïnstalleerd, waardoor het mogelijk is om te communiceren met Pepper. De Rosbridge Server haalt alle beschikbare data op van de NaoQi driver. Door een computer met ROS geïnstalleerd te verbinden met deze server kan er aan de externe spraakherkenning voor Pepper worden ontwikkeld.

² In Bijlage B: Interview is te lezen hoe deze besluiten tot stand zijn gekomen;

4.3 Robot Operating System (ROS)

In dit deel van het onderzoek zal worden ingegaan op hoe ROS werkt. Deze kennis is nodig om uiteindelijk zelf software te kunnen programmeren om Pepper als zelfstandige vraagbak en assistent te laten functioneren. De eerste stap is dan ook om met ROS de audiodata op te halen, zodat dit als input voor een spraakherkenningssysteem kan fungeren.

4.3.1 Over ROS



Figuur 6: ROS

ROS is een systeem voor het besturen van robot componenten vanaf een pc. Het ROS-systeem bestaat uit onafhankelijke nodes, die elk communiceren met andere nodes met behulp van een model voor het publiceren en abonneren van berichten (zogenoemde 'topics'), zie figuur 6. De ROS-master is verantwoordelijk voor het draaien van het systeem en registreren van de nodes. In dit geval is de ROS-master de server. Pepper staat hiermee in verbinding en door een pc te verbinden met de ROS-master kan er eenvoudig ontwikkeld worden aan de spraakherkenning voor Pepper.

4.3.2 Nodes

Een node is een proces dat een berekening uitvoert. Nodes worden met elkaar gecombineerd en communiceren met elkaar met behulp van streamingonderwerpen, RPC-services en de parameterserver. Een node is bedoeld om op zo klein mogelijke schaal te werken. Een robotbesturingssysteem zal dan gewoonlijk veel nodes omvatten (Conley, 2012).

Het stuurprogramma van een bepaalde sensor kan bijvoorbeeld worden geïmplementeerd als een node, dat sensorgegevens in een stream of berichten publiceert. Voor de microfoons van Pepper kan een node worden geschreven die de audiodata uitzend in een bericht. Deze berichten kunnen worden gebruikt door een aantal andere nodes, waaronder filters, loggers, en systemen van een hoger niveau, zoals een spraakherkenningssysteem (Robotics, 2014).

Node types

Nodes hebben ook een node type. Deze node type hebben de naam van het pakket van het node en de naam van het node-uitvoerbaar bestand. Het is hierdoor makkelijk om een node te vinden. Om een node type op te lossen, zoekt ROS naar alle uitvoerbare bestanden in het pakket met de opgegeven naam en kiest de eerste die wordt gevonden. Daarom is het belangrijk om voorzichtig te zijn en geen verschillende uitvoerbare bestanden met dezelfde naam in hetzelfde pakket te produceren. Een ROS-node wordt geschreven met behulp van een ROS-clientbibliotheek, zoals rospy of rospy (Conley, 2012).

4.3.3 Topics

Een node kan berichten uitwisselen. Het bericht wordt topic genoemd. Topics hebben een anonieme publicatie/abonneer-semantiek, die de productie van informatie van het verbruik ontkoppelt. Over het algemeen weten de nodes niet met wie ze communiceren. In plaats daarvan abonneren nodes die geïnteresseerd zijn in gegevens zich voor het betreffende topic; nodes die gegevens publiceren naar het relevante onderwerp. Er kunnen meerdere uitgevers en abonnees zijn voor een topic.

Topics zijn bedoeld voor één richting, streamingcommunicatie. Nodes die externe procedureaanvragen moeten uitvoeren, d.w.z. een antwoord op een verzoek ontvangen, zouden in plaats daarvan diensten moeten gebruiken. Er is ook de parameterserver voor het onderhouden van de kleine hoeveelheden data van een topic (Foureh, 2014).

Topic types

Elke topic wordt sterk getypeerd door het ROS-berichttype dat wordt gebruikt om het te publiceren en nodes kunnen alleen berichten met een passende type ontvangen. De ROS-Master dwingt type-consistentie tussen uitgevers niet af, maar abonnees zullen geen berichttransport instellen tenzij de typen overeenkomen. Verder controleren alle ROS-clients of de bericht bestanden overeenkomen. Deze controle zorgt ervoor dat de ROS-nodes zijn samengesteld op basis van consistente codebases (Foureh, 2014).

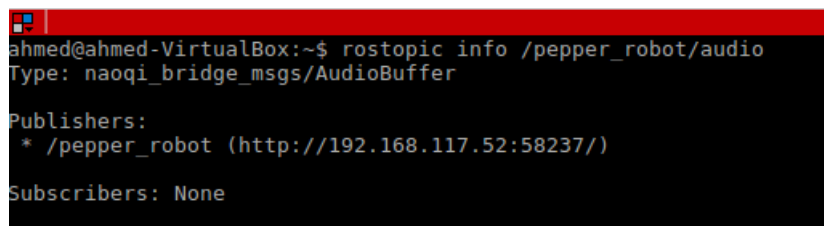
4.3.4 Voordelen

Het gebruik van ROS biedt verschillende voordelen voor het gehele systeem. Er is een extra fouttolerantie omdat crashes worden geïsoleerd naar afzonderlijke nodes. De codecomplexiteit is kleiner in vergelijking met monolithische systemen. Implementatiedetails zijn goed verborgen omdat de nodes een minimale API aan de rest van de graaf blootstellen en alternatieve implementaties, zelfs in andere programmeertalen, eenvoudig kunnen worden vervangen.

4.3.5 Audio van Pepper

Nu omschreven is hoe ROS in elkaar zit, volgt de uitleg hoe met ROS de ruwe audiodata van Pepper verkregen kan worden. Zoals eerder uitgelegd draait de ROS-master op de server en staat Pepper hiermee in verbinding. Door een computer te verbinden aan de ROS-master kan via een ROS-commando een lijst worden opgevraagd van beschikbare nodes en topics³.

In deze lijst is te zien dat Pepper een audio topic heeft. Dit audio topic zal de input vormen voor een spraakherkenningssysteem. Hiervoor moet eerst het bericht van het audio topic worden begrepen, omdat daarin de audiodata zich bevindt. In ROS kan via een andere commando informatie worden opgevraagd over de audio topic, zie figuur 7.



```
ahmed@ahmed-VirtualBox:~$ rostopic info /pepper_robot/audio
Type: naoqi_bridge_msgs/AudioBuffer

Publishers:
 * /pepper_robot (http://192.168.117.52:58237/)

Subscribers: None
```

Figuur 7: Audio topic

³ In Bijlage C: Nodes & Topics is de lijst van beschikbare nodes en topics te zien;

Audio Topic

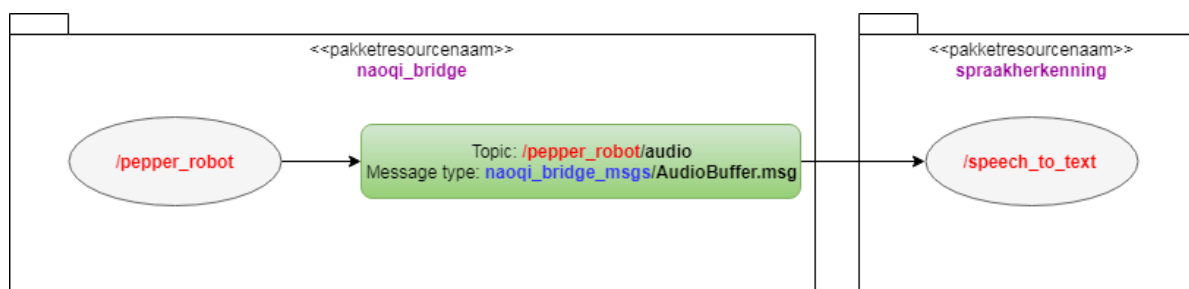
Uit de opgevraagde informatie over de audio topic is te zien dat het wordt gepubliceerd door de /pepper_robot node. Er zijn verder geen andere nodes die zich op het audio topic geabonneerd hebben. Het bericht dat wordt uitgezonden is van het type naoqi_bridge_msgs/AudioBuffer (AudioBuffer, 2017). In de tabel hieronder worden de ruwe data van het bericht uitgelegd.

Naam	Omschrijving
Header	Hierin wordt de timestamp opgeslagen van het moment dat het audiobuffer bericht is verstuurd.
Frequentie	De waarde geeft het aantal trillingen per seconde aan.
Channel Order	Elke microfoon heeft zijn eigen audiokanaal. De volgorde van de audiokanalen die worden opgehaald zijn: linksboven, rechtsboven, rechtsonder, linksonder.
Channel Map	Hierin wordt de volgorde van de audiokanalen opgeslagen in een lijst.
Data	Alle audiodata worden in een lijst van zestien getallen opgeslagen. Elke vier getallen behoren tot een audiokanaal. Dat betekent dat er vier audiokanalen zijn. Per audiokanaal kan de audiodata worden verkregen van een specifieke microfoon.

Tabel 3: AudioBuffer

ROS-softwarepakket

Om dit audio topic te kunnen gebruiken voor een spraakherkenningssysteem moet er een ROS-softwarepakket gemaakt worden. Dit pakket kan bijvoorbeeld een node /speech_to_text bevatten die zich abonneert op het audio topic en vervolgens de audio kan omzetten naar tekst. Zie de figuur hieronder.



Figuur 8: Pepper Robot

4.4 Spraakherkenning

In dit subhoofdstuk wordt vervolgens gekeken hoe de met ROS verkregen audiodata kan worden omgezet naar tekst. Het omzetten van de audio naar tekst is nodig om deze vervolgens te kunnen analyseren, zodat Pepper op basis van de tekst met behulp van Natural Language Processing software kan begrijpen wat er gezegd wordt. Voor het omzetten van deze audiodata van Pepper zal daarom een externe spraakherkenning gekozen worden die onder andere kan omgaan met de realtime audioverwerking. De focus in dit subhoofdstuk ligt in hoe spraakherkenning in elkaar zit en het selecteren van een spraakherkenningssysteem.

4.4.1 Over spraakherkenning

De termen spraakherkenning en stemherkenning worden soms door elkaar gebruikt, maar ze betekenen niet hetzelfde.

Spraakherkenning

Spraakherkenning is het proces van woorden en zinsdelen identificeren als iemand praat en het omzetten naar een digitaal leesbaar formaat. Spraakherkenning software heeft een gelimiteerde woordenlijst en zinnen en het kan alleen geïdentificeerd worden als deze woorden en zinnen duidelijk worden uitgesproken (Rouse, 2016). Spraakherkenning staat ook bekend als automatic speech recognition (ASR) en speech to text (STT).

Stemherkenning

Stemherkenning is het identificeren van de persoon die aan het praten is. Iedereen heeft een uniek spraakpatroon uit hun anatomie (de grootte en vorm van de mond en keel) en gedragspatroon (toonhoogte, spreekstijl, accent, enzovoort). Stemherkenning wordt in combinatie met spraakherkenning gebruikt om apparaten te besturen, opdrachten uit te voeren of te schrijven zonder een toetsenbord, muis of te klikken op een knop (Hope, 2017).

4.4.2 Voor- en nadelen

Het voordeel van spraakherkenning is dat het makkelijk te gebruiken is. Het wordt vaak al op computers en mobiele apparaten geïnstalleerd, zodat het toegankelijk is. Het nadeel is dat het niet staat is om alle woorden vast te leggen vanwege variaties in uitspraak. Er is een gebrek aan ondersteuning voor de meeste talen buiten Engels (Rouse, 2016).

4.4.3 Soorten spraakherkenningssystemen

Voor spraakherkenning zijn er verschillende soorten systemen ontwikkeld die spraak omzetten naar tekst (Hope, 2017). Hieronder in de tabel is een overzicht van bekende spraakherkenningssystemen:

Type systeem	Toelichting
Speaker dependent system	Het systeem moet getraind worden voordat het gebruikt kan worden. Het is dan nodig om een aantal series van woorden en zinnen te lezen.
Speaker independent system	Het systeem herkent de meeste gebruikers stemmen zonder training.
Discrete speech Recognition	De gebruiker moet pauzes nemen tussen elk woord, zodat het woord kan worden geïdentificeerd.
Continuous speech recognition	Het systeem kan een normale snelheid van spreken begrijpen.

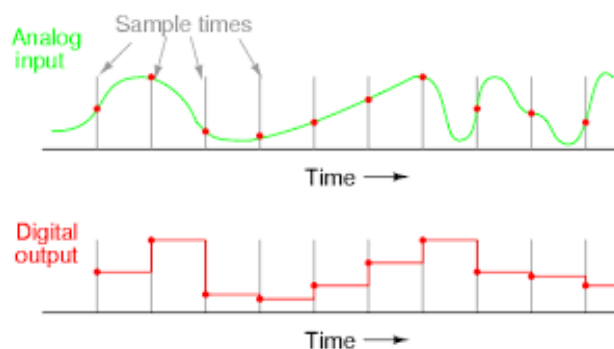
Tabel 4: Soorten spraakherkenningssystemen

4.4.4 Werking

Spraakherkenningssoftware maakt gebruik van verschillende processen om te begrijpen wat iemand zegt.

Voorbewerking

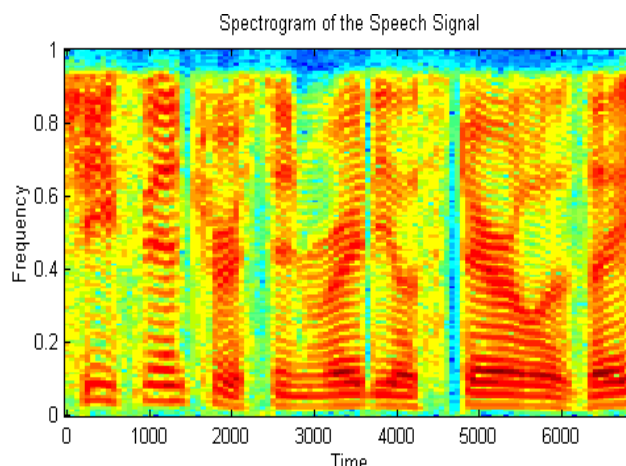
De stem wordt opgenomen door een microfoon als geluidsgolven. Een geluidsgolf is een analoog signaal en deze moet eerst worden omgezet naar een digitaal signaal, zodat een computer er gebruik van kan maken. Hiervoor wordt de hoogte van de geluidsgolf op gelijkmatige verspreide punten gemeten; een techniek die sampling wordt genoemd.



Figuur 9: Sampling

Spectrogram

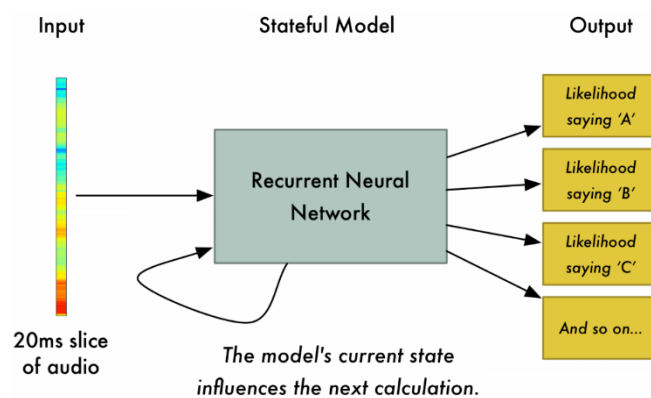
Uiteindelijk zijn er een reeks getallen over die de amplitudes van het geluidssignaal op bepaalde tijdstippen voorstellen. Het signaal zal worden opgesplitst in een tijdsframe. In deze tijdsframe kunnen de verschillende frequenties terug gevonden worden die samengevoegd het complexe geluid van menselijke spraak vormen. Bij spraakherkenningstechnologie wordt het complexe geheel van frequenties opgesplitst in frequentiebanden. In elke frequentieband kan worden nagegaan hoeveel energie het bevat om als het ware de vingerafdruk van het geluid te verkrijgen. Deze vingerafdruk wordt spectrogram genoemd. Door de spectrogrammen in de tijd naast elkaar te plaatsen wordt een visuele weergave van het uitgesproken woord getoond.



Figuur 10: Spectrogram

Fonemen

Fonemen zijn de verschillende basisklanken die samen onze taal vormen. Het audiosignaal is omgevormd tot een formaat dat eenvoudig te begrijpen is. Het kan nu worden verstuurd naar een artificieel neurale netwerk. Voor de input wordt gebruik gemaakt van de spectrogrammen. De output die wordt gegenereerd geeft de kansen weer dat het signaal een bepaald foneem voorstelt. Iedereen zal bijvoorbeeld een 'p' net anders uitspreken, maar er wordt wel de klank 'p' verstaan. Gemiddeld genomen zijn er 35 fonemen per taal; bij het Nederlands bestaan er 40 verschillende klanken (Vervoort, 2017).



Figuur 11: Neuraal netwerk

Trainen

Het artificieel neurale netwerk moet de juiste klanken kunnen raden. Hiervoor moet het getraind worden met een grote dataset. Iedereen spreekt fonemen net iets anders uit. Een slecht getraind neurale netwerk zal door variaties in uitspraak in de war kunnen worden gebracht. Door een grote hoeveelheid trainingsdata te gebruiken, zal het netwerk leren om beter de fonemen te onderscheiden van elkaar. Het leerproces gaat zelfs verder als de software al in gebruik is (Vervoort, 2017).

Woorden

Ondanks een goed getraind artificieel neurale netwerk blijft het een grote opdracht om klanken om te zetten in letters en woorden. Het neurale netwerk zal daarom een geheugen hebben van voorgaande bevindingen die invloed heeft op toekomstige schattingen. Door dergelijke overwegingen te koppelen aan de herkenning van een klank zal het neurale netwerk veel accurater reageren.

Er blijven uiteindelijk enkele woorden over, waarvan het netwerk met wisselende zekerheid denkt dat het juiste woord is. De software wordt gekoppeld aan een database waarin wordt aangegeven hoe waarschijnlijk het is dat een bepaald woord voorkomt. Door het neurale netwerk erg veel geschreven teksten te laten lezen kan het beter het woord kiezen dat het meest voorkomt (Vervoort, 2017).

4.4.5 Performance

De performance wordt gemeten aan de hand van nauwkeurigheid en snelheid. De nauwkeurigheid wordt gemeten met de Word Error Rate (WER). WER werkt op woordniveau en identificeert onnauwkeurigheden in transcriptie. Het kan niet identificeren hoe een fout is opgetreden. Een fout kan een vervanging (substitution), een verwijdering (deletion) of een toevoeging (addition) zijn. Hieronder in de tabel wordt een voorbeeld gegeven.

Expected output	This sentence will be recognized
Substitution	This sentence will bee recognized
Deletion	This sentence will recognized
Addition	This sentence will not be recognized

Tabel 5: Word Error Rate

De snelheid is gemeten in de tijd die nodig is om een woord om te zetten in tekst. Een variatie van factoren heeft effect op de performance, zoals uitspraak, accent, pitch, volume en achtergrondgeluid (Rouse, 2016).

Audiokwaliteit

Om specifiek te achterhalen welke factoren mogelijk de performance belemmeren is er naar de audio opnamen van Pepper geluisterd. Uit deze opnamen is naar voren gekomen dat de stem zacht wordt opgenomen. Dit zal waarschijnlijk komen, doordat de microfoons inwendig en afgesloten ingebouwd zijn in Pepper.

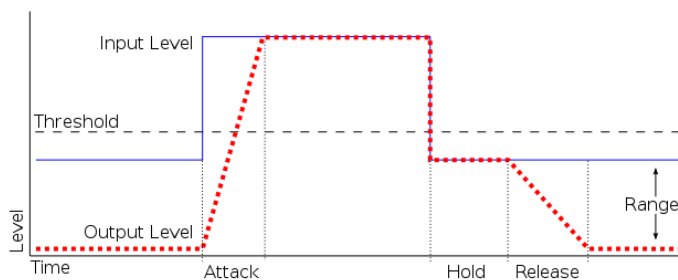
Het andere nadeel van deze ingebouwde microfoons is dat alle geluiden van de motoren worden opgenomen. Verder wordt de stem van Pepper ook opgenomen door de microfoons. De verwachting is dat de audiokwaliteit niet al te goed is voor spraakherkenning. Het beste zou een externe microfoon of een audiokaart kunnen worden toegevoegd, maar de nauwe micro usb port ligging maakt dit erg lastig.

4.4.6 Noise Gate

Een deel van de factoren zoals uitspraak en accent ligt bij de gebruiker zelf, maar het achtergrondgeluid ligt aan de omgeving waarin Pepper zich bevindt. Hier zal rekening mee moeten worden gehouden. Het is niet slim om de spraakherkenning te gebruiken in een drukke omgeving zoals bij evenementen, omdat dit behoorlijk de performance belemmert. De omgeving waarin Pepper zich zal bevinden is bij de servicedesk. Daar zijn achtergrondgeluiden zoals mensen die in de achtergrond praten, lopen en bezig zijn met apparatuur. Pepper moet dus zo min mogelijk van deze geluiden meenemen in de spraakherkenning. Ook moet er een signaal zijn wanneer de gebruiker actief praat met een bepaald volume.

De noise gate zal gebruikt worden om zo min mogelijk achtergrondgeluiden op te nemen voor de spraakherkenning, omdat geluiden zonder spraak niet relevant zijn. Ook zal de noise gate als trigger worden gebruikt om het moment van audio streamen te starten als een persoon praat met een bepaald volume. Hierdoor worden mensen die in de achtergrond praten niet meegenomen in de spraakherkenning, mits ze niet al te hard praten.

Werking



Figuur 12: Noise Gate

De noise gate is een luik waar alleen geluid doorheen mag als het over een bepaald geluidsniveau komt, ook wel de threshold genoemd. Het geluidsniveau wordt berekend aan de hand van Root Mean Square (RMS). RMS heeft dezelfde fysieke eenheid volt (eenheid voor spanning) als Pascal (eenheid voor druk).

De RMS-waarde helpt om feedback terug te geven over wat het geluidsniveau is. Deze waarde kan dan worden vergeleken in verschillende omgevingsomstandigheden. Het laat zien hoe luid iemand moet praten tegen Pepper om überhaupt verstaan te worden in een omgeving.

Wanneer het geluid de threshold overschrijdt blijft het luik open tot dat het geluidsniveau onder de threshold komt. Vervolgens blijft het luik voor een bepaalde tijd nog open om te voorkomen dat iemand nog zijn zin moet afmaken. Uiteindelijk sluit het luik en herhaalt het proces zich. Een typisch geluidsniveau voor een stille ruimte zal tussen 0 en 100 liggen en een typisch geluidsniveau voor als iemand praat zal tussen 150 en 3500 liggen (Izhaki, 2011).

Omgevingsfactoren

Om een beeld te krijgen wat de performance in verschillende omgevingen kan zijn, is een test uitgevoerd. In deze test is per omgeving elke minuut het geluidsniveau gemeten. Dit werd tien minuten lang gedaan, zodat daarvan het gemiddelde geluidsniveau kon worden berekend. De resultaten zijn in de bijlage terug te vinden⁴.

⁴ In Bijlage D: Omgevingsfactoren zijn de resultaten te vinden;

Er zijn in dit onderzoek gekozen voor drie omgevingsfactoren:

- ▶ Een Atos vergaderruimte, dit is een afgesloten ruimte waar weinig tot geen geluid is;
- ▶ Het Technology Lab waar ontwikkeld wordt aan Pepper;
- ▶ De servicedesk waar Pepper mogelijk in de toekomst wordt geplaatst.

In de vergaderruimte zal de spraakherkenning optimaal moeten werken, omdat het stil is. Alleen de motorgeluiden van Pepper beïnvloeden het geluidsniveau.

In de Technology Lab was er een groepje van drie mannen aanwezig die op een normale toon aan het praten waren achter Pepper. Uit de resultaten blijkt dat het geluidsniveau een beetje hoger is dan de vergaderruimte, met daarin soms af en toe ruis die Pepper heeft opgenomen van de drie mannen.

Tot slot is er de servicedesk. Hier was te zien dat het lawaaiër kon zijn. Het geluidsniveau moet niet al te hoog zijn en zo constant mogelijk blijven, zodat de spraakherkenning beter in staat is de spraak om te zetten. In de servicedesk zal soms het lawaai de spraakherkenning kunnen belemmeren. Het is verstandig om te wachten tot het rustig is om iets te vragen aan Pepper

4.4.7 Eisen

Uit de voorgaande paragrafen over spraakherkenning en op basis van de eerdere omschrijving dat Pepper zelfstandig bij de servicedesk moet kunnen functioneren, kan een eisenlijst samengesteld worden waaraan het spraakherkenningssysteem voor Pepper moet voldoen. Dit zijn:

- ▶ Het moet geprogrammeerd worden in Python als een node in ROS;
- ▶ Het moet van het type Continuous Speech Recognition zijn, zodat het een normaal tempo van spreken kan bijhouden;
- ▶ Het moet een lage WER hebben, zodat spraak zo nauwkeurig mogelijk wordt omgezet naar tekst;
- ▶ Het moet spraak zo snel mogelijk real time omzetten naar tekst, zodat er sneller antwoord op vragen gegeven kan worden aan de spreker;
- ▶ Het moet voorzien zijn en getraind op basis van een grote dataset (woordenlijst) om woorden beter te kunnen kiezen;
- ▶ De Nederlandse en/of Engelse taalmodule moet aanwezig zijn en hoeft niet volledig zelf gemaakt en getraind te worden.

4.4.8 Kaldi-NL

Op basis van de bovenstaande eisenlijst, kan een selectie gemaakt gaan worden voor een geschikt spraakherkenningssysteem voor Pepper.

Binnen Atos draait reeds een spraakherkenningssysteem, Kaldi-NL. Kaldi-NL werd gebruikt om klantengesprekken om te zetten naar tekst met externe microfoons. Er zal eerst gekeken worden of dit makkelijk te koppelen is met Pepper en aan de eisen voldoet.

Universiteit Twente en Nederlands Instituut voor Beeld en Geluid hebben samen gewerkt aan de open source Nederlandse spraakherkenning Kaldi-NL (Opensource spraakherkenning, n.d.). Een klein audio fragment van Pepper is handmatig verstuurd naar de Kaldi-Server om het om te zetten naar tekst. Daaruit kwam naar voren dat de procestijd nogal lang en de Word Error Rate nogal groot was. Dit heeft onder andere te maken met de audiokwaliteit van Pepper en het feit dat het systeem voor andere doeleinden getraind is. Het systeem moet nog verder uitgebouwd worden om real time de audio te kunnen omzetten naar tekst. Dit ontbreekt en maakt het sowieso onmogelijk voor Pepper om te gebruiken. Het kost veel tijd om dit te maken en daarom is ervoor gekozen om dit niet te gaan gebruiken binnen dit onderzoek. Nu Kaldi-NL niet bleek te voldoen, moet er een keuze gemaakt worden tussen andere spraakherkenningssystemen.

4.4.9 Selectie

De audiodata van Pepper moet verstuurd worden naar een spraakherkenningssysteem service. Om te voldoen aan de eis dat het systeem als node in ROS met Python kan worden geschreven, is het raadzaam om een reeds bestaande Python library als uitgangspunt hiervoor te nemen. Een library is een softwarepakket met programmeer tools om software of applicaties te ontwikkelen. Voor spraakherkenning is de Python library SpeechRecognition open source beschikbaar. De services die door de library worden ondersteund zijn: CMU sphinx, Google Cloud, Wit.ai, Microsoft Bing, Houndify, IBM Watson en Snowboy Hotword Detection (Zhang, n.d.), wat voor dit project betekent dat de keuze voor een spraakherkenningssysteem een van de bovenstaande services moet zijn. Dit aangezien de beschikbare tijd een beperkende factor is om de software om de audiodata te versturen vanaf scratch te ontwikkelen voor alle spraakherkenningssystemen.

CMU sphinx en Snowboy Hotword Detection komen niet in aanmerking, omdat deze alleen zelf opgegeven woorden detecteren. Dit is niks anders dan hoe Pepper nu zelf functioneert. Houndify komt ook niet in aanmerking omdat het slechts een beperkt audioformaat accepteert (Biran, 2017) wat vanuit Pepper niet goed te verkrijgen is met ROS.

Omdat betrouwbaarheid van het systeem het belangrijkste is, moet van de overige beschikbare systemen gekeken worden naar de benchmark gegevens betreffende de performance op het gebied van Word Error Rate en snelheid. Uit de recente benchmarks van een expert op dit gebied, Franck Dernoncourt, is te zien dat Google Cloud en IBM Watson goed presteren (Dernoncourt, 2018), waardoor besloten is om een van deze twee systemen te kiezen. De door hem ontworpen benchmark tool kan later gebruikt worden om zelf met de audio van Pepper een actueel benchmark te verkrijgen.

Google Cloud

Google Cloud Speech-to-Tekst heeft een krachtig neuraal netwerk om spraak zo nauwkeurig mogelijk om te zetten naar tekst. De spraakherkenning software is terug te zien op Android telefoons waarop Google Voice standaard wordt meegeleverd. Het fijne van Google is dat het zelf al over meer dan 120 taalmodules beschikt, inclusief Nederlands. Een ander pluspuntje is dat Google een Noise Robustness feature heeft die met ruis om kan gaan. De prijs voor het omzetten van spraak naar tekst is \$0.006 USD per 15 seconden (Cloud Speech-To-Text, n.d.).

IBM Watson

IBM ondersteunt op het moment geen Nederlandse taalmodule, maar het is wel aangegeven dat er in de toekomst meer taalmodules uit zullen komen. De prijs van het standaardpakket van IBM Watson speech to text is \$0.020 USD per minuut met de eerste 1000 minuten gratis (Watson Speech-To-Text, n.d.). IBM heeft in 2016 aangekondigd dat ze zullen werken aan een Watson versie voor Softbank Robotics, de leverancier van Pepper (Crowe, 2016). Dit houdt in dat er meer ondersteuning komt in de toekomst voor Pepper.

4.4.10 Databeheer

Een belangrijk punt om stil te staan bij Google en IBM is het beheer van data. Doordat deze diensten op hun Cloud worden gebruikt worden de data ook bij Google en IBM beheert. Atos wil natuurlijk de privacy van haar medewerkers en klanten garanderen. Het moet niet zo zijn dat gevoelige informatie door de spraakherkenning wordt opgenomen en daar negatieve gevolgen aan zijn verbonden.

Atos is partners met Google en IBM en zouden eventueel in overleg kunnen gaan met wat er precies met data gebeurt en of ze er zelf inzage op hebben. Het is ook mogelijk om de audio zelf en de tekst die wordt verkregen van Google of IBM op te slaan. Voor dit onderzoek zal bij het testen van de spraakherkenning hiermee rekening worden gehouden. De tester zal op de hoogte worden gebracht en om goedkeuring worden gevraagd. Het verzamelen van deze data geeft inzicht op de performance. Dit helpt om het prototype te verbeteren.

4.4.11 Besluit

Om Google en IBM met elkaar te vergelijken is er een benchmark gemaakt. Hiervoor is de taal Engels gekozen, omdat ze beiden deze taalmodule hebben.

In de eerste benchmark zijn er vier audiobestanden gekozen. In deze audiobestanden worden door vier personen een zin uitgesproken. De WER van IBM presteert beter dan Google, maar het valt op dat google een snellere responsetijd heeft.

In de tweede benchmark is het audiobestand van Pepper gebruikt. Hierin hebben drie personen dezelfde stuk tekst uitgesproken. De omgeving die hiervoor is gekozen is een afgesloten stille ruimte, omdat het geluidsniveau laag en constant is. Het is gebleken dat in deze omstandigheden Google beter presteert dan IBM. Dit kan komen door de Noise Robustness feature van Google, die slimmer weet om te gaan met de motorgeluiden van Pepper. Verder is de responsetijd van IBM nog steeds langzamer dan die van Google.

Beide hebben ze een lage WER, maar waar het probleem in zit is de responsetijd. Ondanks dat Google beter presteert dan IBM met de audio van Pepper is de responsetijd nog steeds van beiden te traag⁵.

Na kritisch te kijken naar het prototype is gebleken dat de responsetijd ligt in hoe de Speech Recognition Library werkt van Python. Er wordt een request gedaan naar de server, waarbij de connectie steeds wordt geopend en afgesloten. De connectie moet openblijven en real time alle audiodata verwerken via een stream. Dit kan door middel van een socket. Een socket houdt de connectie open en luistert naar de ontvangen data en geeft meteen een respons terug als die beschikbaar is.

Om de issues met de responstijd op te lossen, moest er dus zelf een socket geprogrammeerd worden voor Google Cloud en IBM Watson om te bepalen welk spraakherkenningssysteem het meest geschikt is voor Pepper. Voor beiden was officiële documentatie beschikbaar, maar op basis van een andere Python library om de audio van de microfoons te verkrijgen. In de praktijk kon deze code niet worden uitgevoerd, omdat Pepper niet de permissie geeft aan de microfoons om de audiodata te verkrijgen. Dit was ook niet nodig, omdat de audiodata al wordt verkregen als topic met behulp van ROS. De vraag was hoe dit te combineren met de API van Google en IBM. Uiteindelijk is het gelukt om dit te maken voor IBM Watson. Mede omdat er meer informatie te vinden was betreffende de koppeling tussen ROS, IBM Watson en Pepper.

Uit de uiteindelijke testresultaten is gebleken dat de responstijd van de socket implementatie inderdaad sneller is dan van de request implementatie. De gemiddelde responstijd is inmiddels 0.2 seconde en dit is perfect om real time tegen Pepper te kunnen praten⁶.

De keuze is dus gevallen op het gebruik van het spraakherkenningssysteem van IBM Watson.

⁵ In Bijlage E: Benchmarks zijn de benchmarks terug te vinden;

⁶ In Bijlage F: Metingen zijn de metingen terug te vinden;

4.5 Contextbepaling

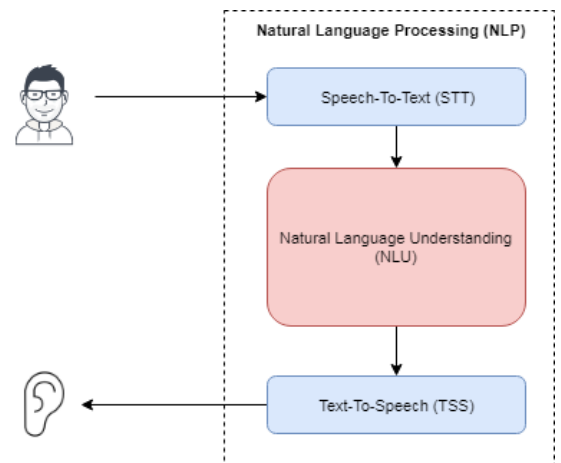
Zoals in hoofdstuk 4.4 al aangegeven, dient de verkregen tekst van de spraakherkenning geanalyseerd te worden. Er moet worden vastgesteld wat de context is van de zin om een vervolg besluit te kunnen nemen. De context van een zin is afhankelijk van de omgeving van Pepper. Er zullen namelijk andere vragen worden gesteld aan Pepper bij de servicedesk dan bij bijvoorbeeld een gemeentehuis. Deze vragen zijn belangrijk om de context van de zin te kunnen bepalen, want zonder voorbeeldvragen is dit lastig. Hiervoor wordt gebruik gemaakt van Natural Language Processing.

4.5.1 Natural Language Processing

Natural Language Processing (NLP) is een deel van Artificial Intelligence (AI) die zich bezighoudt met taal begrijpen, verwerken en genereren net als een persoon. NLP maakt gebruik van Machine Learning (ML). ML slaat woorden op en de manieren waarop ze samenkomen. Uitdrukkingen, zinnen en hele boeken worden in een ML-engine ingevoerd, waarop ze worden verwerkt op basis van grammaticale regels, de taalkundige gewoonten van mensen of beide. De computer gebruikt deze gegevens vervolgens om patronen te vinden en te voorspellen wat erna komt (Bell, 2018).

Spraakherkenning is onderdeel van NLP, zie figuur 13. De tekst die wordt verkregen zal begrepen moeten worden met Natural Language Understanding (NLU). NLU is verantwoordelijk voor het achterhalen van de bedoeling van de tekst, de acties die daarbij moeten worden uitgevoerd om tot een antwoord te komen en het teruggeven van een menselijk leesbaar tekstformaat. Hiervoor worden verschillende services en systemen met elkaar gebruikt (Bobriakov, 2018).

Het laatste onderdeel van NLP is de Text-To-Speech (TTS), waarbij de tekst door de computer op zo natuurlijk mogelijke manier wordt uitgesproken. Pepper beschikt al over TSS.



Figuur 13: Natural Language Processing

4.5.2 IBM Assistant

Een belangrijk onderdeel van Natural Language Understand is de bedoeling achterhalen van de tekst, zodat daarop gereageerd kan worden. Het systeem dat daarvoor verantwoordelijk is vereist een dataset om getraind te worden. Als Pepper in de toekomst in de servicedesk ingezet gaat worden kunnen de vragen van klanten en medewerkers als trainingsdata worden gebruikt. De scope van Pepper wordt dan ook specifiek gedefinieerd waardoor Pepper meer gericht getraind kan worden. Het is dan mogelijk voor Pepper om klanten en medewerkers door processen te helpen net zoals een baliemedewerker.

Om hieraan een begin te maken is in dit onderzoek gekozen voor IBM Assistant (voorheen IBM Conversation), omdat binnen het tijdsbestek en beschikbaarheid het meest toepasselijke was. Het wordt het meeste gekozen door bedrijven (Monsters, 2017), het sluit goed aan op de IBM Watson speech-to-text, de documentatie is duidelijk en kan via een interface snel een dataset importeren, getraind en getest worden. IBM Assistant bestaat uit intents, entities en dialogs (Bluemix, 2018).

Intents

De bedoeling van een zin wordt een intent genoemd. Als een gebruiker aan Pepper vraagt of een vergaderruimte vrij is, dan is de intent om te controleren welke vergaderruimtes vrij zijn. In deze voorbeeld zou Check_Meetingsrooms de intent naam kunnen zijn.

Hoe meer voorbeeldzinnen worden gegeven bij de Check_Meetingsrooms intent, des te makkelijker Pepper in staat is de bedoeling ervan te begrijpen. Dit is een groot verschil met hoe de spraakherkenning van Pepper werkt. Daarin wordt context bepaald aan de hand van de exacte woorden, terwijl in werkelijkheid een zin op verschillende manier gezegd kan worden.

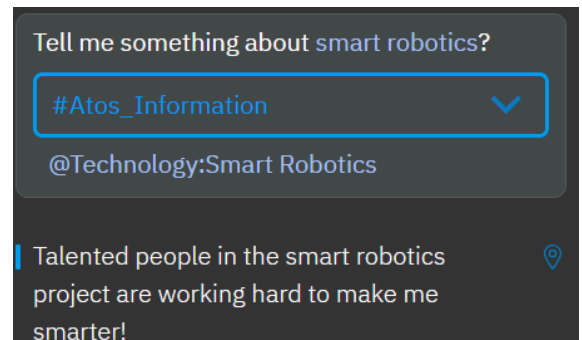
```
[WARN] [1532593999.002727]: User: are you a robot
[WARN] [1532593999.409614]: intent: General_Human_or_Bot
[WARN] [1532593999.410542]: Pepper: Yes ofcourse. I am your robot assistant
```

Figuur 14: Intent

Entities

Een entity is een term of object in de zin van de gebruiker die verduidelijking of specifiek context biedt voor de intent. Als intents werkwoorden voorstellen (iets wat een gebruiker wil doen), vertegenwoordigen entities zelfstandige naamwoorden (zoals het object van, of de context voor, een actie).

In andere woorden, entities zijn woorden die de bedoeling van een zin net wat specifieker maken. Om een voorbeeld te geven kan een klant informatie over de projecten in het Technology Lab vragen, zie figuur 15. Door Smart Robotics in de zin te zeggen, wordt specifiek informatie gevraagd over het Smart Robotics project. Deze variabele wordt dan opgeslagen als entity.



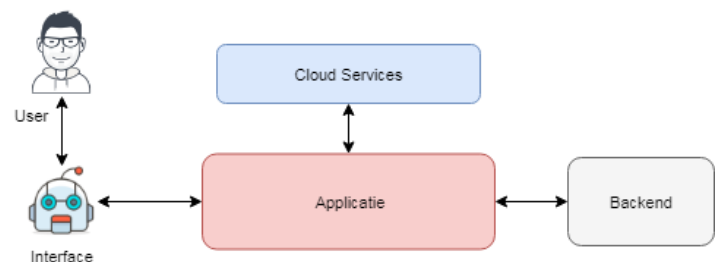
Figuur 15: Entities

Dialogs

Dialogs geven de flow aan van een proces. Door intents is Pepper instaat om een bepaalde bedoelingen te detecteren. Als bijvoorbeeld de Check_Meetingsrooms intent wordt gedetecteerd, dan kan Pepper daarop antwoorden. Zo kan Pepper antwoorden met: laat me dat voor jou checken. Zo krijgt de gebruiker feedback dat Pepper bezig is met opzoeken van beschikbare vergaderruimtes. Vervolgens als de beschikbare vergaderruimtes zijn opgehaald, kan Pepper dat aan de gebruiker vertellen. Pepper kan ook doorvragen op specifieke entities als die ontbreken. Zo is er een controle dat een gebruiker niet bepaalde informatie vergeet.

4.5.3 Platform

Waar het op neer komt is dat er platform nodig is waarbij allemaal services en systemen met elkaar gekoppeld kunnen worden, zie de figuur rechts. Pepper is een interface dat interacties heeft met de gebruiker. Het platform voorziet Pepper dan van de benodigde intelligentie. Het platform bestaat uit de applicatie, die de gebruikers input ophaalt en de robot aanstuurt, in dit onderzoek is ROS de applicatie. Verder Cloud services, waar Natural Language processing plaats vindt, en een backend waar data opgeslagen en opgehaald kan worden. Dit is een oplossing om Pepper in de toekomst zo goed mogelijk te laten functioneren als zelfstandig vraagbak en assistent.



Figuur 16: Platform

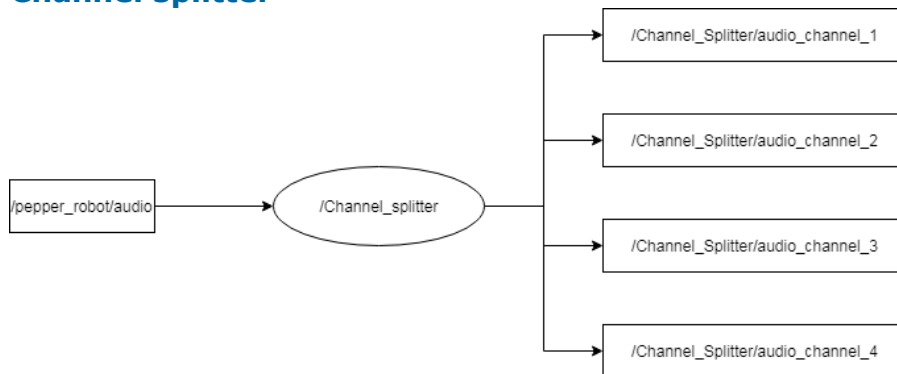
5 Realisatie

In dit hoofdstuk worden eerst de request realisatie uitgewerkt. Hierin wordt uitgelegd hoe de nodes en topics de audiodata versturen naar een spraakherkenning service. Daarna wordt de socket realisatie uitgewerkt. Wat een verbeterde versie is van de eerste realisatie.

5.1 Request

In de request realisatie wordt de audiodata verstuurd naar de spraakherkenning service van Google Cloud of IBM Watson. De nodes en topics die hiervoor zijn gemaakt worden uitgelegd.

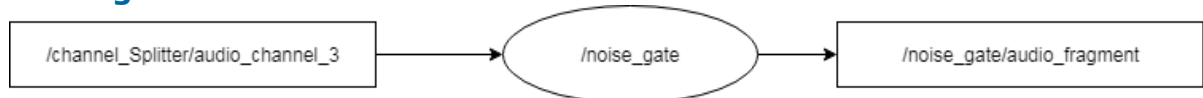
5.1.1 Channel splitter



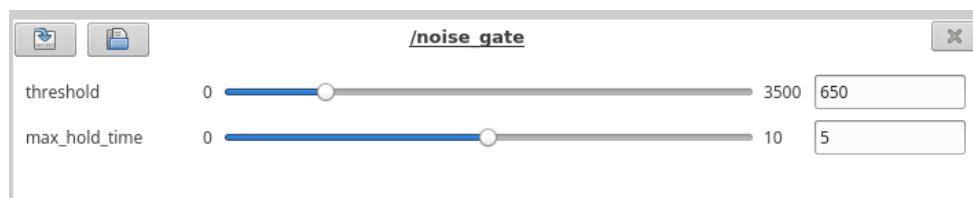
Figuur 17: Channel Splitter Node

De Channel Splitter node abonneert zich op het audio topic van de /Pepper_robot node. De audio wordt opgesplitst in aparte audiokanalen voor elke microfoon. Deze worden door de channel splitter weer gepubliceerd. Het belang van deze node is dat de audiodata kan worden gebruikt van de voorste microfoons.

5.1.2 Noise gate



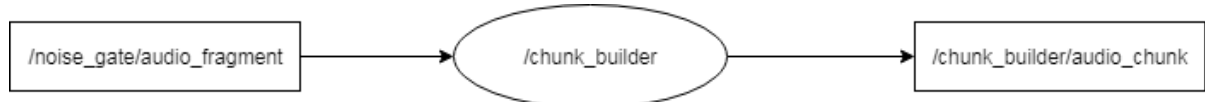
Figuur 18: Noise gate Node



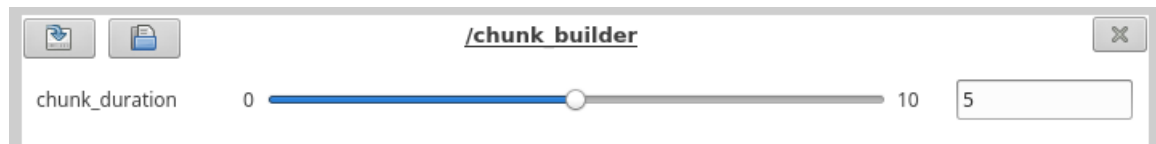
Figuur 19: Noise gate configuratie

De noise gate kan zich op een van de audiokanalen topics abonneren van de Channel splitter node. Audiokanaal drie is de links bovenste microfoon en audiokanaal vier is rechts bovenste microfoon. Deze microfoons hebben de beste kwaliteit voor spraak. De noise gate controleert of de spraak voldoet aan de threshold en publiceert vervolgens het audio fragment. Hierdoor wordt het aantal ruis verminderd. De waarde van de threshold en de opnametijd kan dynamische worden aangepast.

5.1.3 Chunk builder



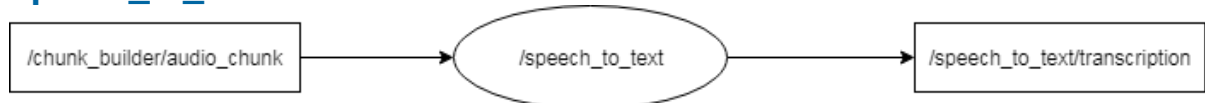
Figuur 20: Chunk builder node



Figuur 21: Chunk builder configuratie

De chunk builder node abonneert zich op het audio fragment topic van de noise gate node. Er wordt van deze audio fragment een groter audio stuk gemaakt. De tijd kan zelf worden ingesteld in de configuratie. In de praktijk was het gebleken dat het direct versturen van de audiodata naar de spraakherkenning service voor problemen zorgde. Voor elke kleine audiodata werd er een request gemaakt. Het gevolg hiervan was een tijdelijke ban van de spraakherkenning service.

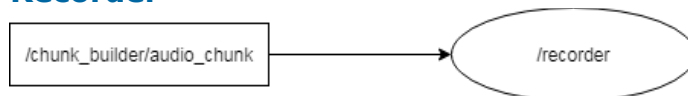
5.1.4 Speech_to_text



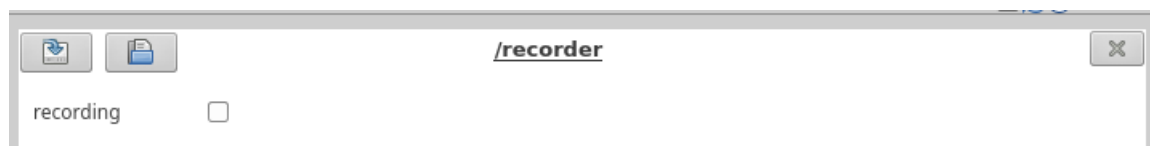
Figuur 22: Speech to text node

De speech to text node abonneert zich op het audio chunk topic van de chunk builder node. In deze node wordt de Python Speech Recognition Library gebruikt om de audio via een request op te sturen naar Google of IBM. Vervolgens wordt via een respons de tekst ontvangen. Het is bij deze node gebleken dat de responsetijd te traag was en dat de audio niet goed werd omgezet naar tekst.

5.1.5 Recorder



Figuur 23: Recorder node



Figuur 24: Recorder configuratie

De recoder node abonneert zich op het audio chunk topic van de chunk builder node. Deze node schrijft de audio weg naar een bestand. Het luisteren naar het bestand geeft een beeld over de audiokwaliteit. De audio kan ook handmatig worden opgestuurd naar een spraakherkenning service om te controleren of de uitkomst hetzelfde is als hoe het systeem nu draait in ROS. Hierdoor kwam naar voren dat het systeem niet dezelfde uitkomst en dit probleem opgelost moest worden.

5.1.6 Overzicht

Hieronder in de tabel is een overzicht te vinden van alle topics en nodes.

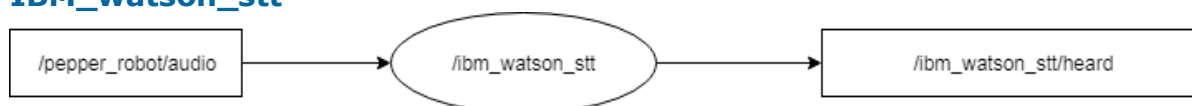
Node	Topic	Functionaliteit
Channel Splitter	Abonneert: audioBuffer Publiceert: Audio_Channel_1 Audio_Channel_2 Audio_Channel_3 Audio_Channel_4	Splits elke microfoon van Pepper in aparte audiokanalen. De audiodata wordt in een stream verstuurd.
Noise Gate	Abonneert: channel_splitter/audio_channel_{} Publiceert: Audio_fragment	Luistert naar het geluid van een van de microfoons en laat alleen geluid door als het over een bepaald geluidsniveau (threshold) komt
Chunk builder	Abonneert: Noise_gate/audio_fragment Publiceert: audio_chunk	Maakt van kleine audio stukjes een groter stuk
Recorder	Abonneert: Chunk_builder/audio_chunk Publiceert: N/A	Schrijft de audio naar een .wav bestand
Speech_to_text	Abonneert: Chunk_builder/audio_chunk Publiceert: transcription	Gebruikt de Speech Recognition Library, waarmee de audio opgestuurd kan worden naar Google Cloud of IBM Watson.

Tabel 6: Request realisatie

5.2 Socket

In de socket realisatie wordt de connectie opengehouden tussen IBM Watson en Pepper. De audiodata wordt constant real time verstuurd naar IBM Watson. Als de spraak wordt omgezet naar tekst wordt dit terug gestuurd als een respons. Vervolgens wordt deze tekst verstuurd naar IBM Assistant om te bepalen wat de intent is.

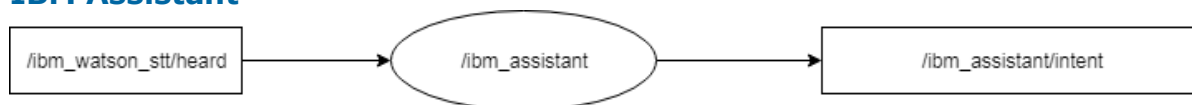
5.2.1 IBM_watson_stt



Figuur 25: IBM watson stt node

De IBM Watson STT node abonneert zich op het audio topic van de Pepper robot node. In vergelijking tot de channel splitter node is het niet meer nodig om de audiokanalen te splitsen. In de IBM Watson STT node kan via de parameter aangegeven worden welke audiokanaal gebruikt moet worden. De audiodata wordt op deze manier opgehaald en via een open connectie constant verstuurd naar de IBM Watson service. Als IBM Watson de spraak heeft omgezet naar tekst wordt de tekst teruggestuurd naar de node, zodat het uiteindelijk gepubliceerd kan worden.

5.2.2 IBM Assistant



Figuur 26: IBM assistant node

De IBM Assistant node gebruikt het heard topic van de IBM Watson STT node om te bepalen wat de bedoeling is van de tekst. De intent wordt gepubliceerd zodat andere nodes een actie erop kunnen uitvoeren.

5.2.3 Overzicht

Hieronder in de tabel is een overzicht te vinden van alle topics en nodes.

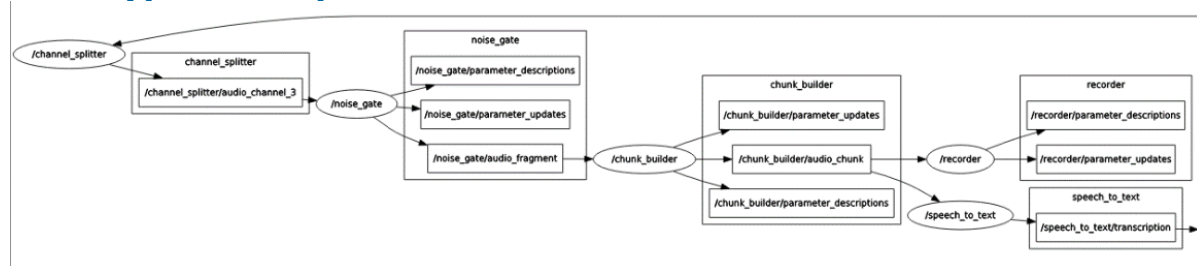
Node	Topic	Functionaliteit
Ibm_watson_stt	Abonneert: audioBuffer Publiceert: Heard	Zet de spraak om naar tekst door middel van een web socket
Ibm_assistant	Abonneert: Ibm_watson_stt/heard Publiceert: Intent	Bepaald wat de bedoeling is van de tekst. Verstuurt de intent, entities en de dialoog terug naar de node.

Tabel 7: Socket realisatie

6 Resultaten

In dit hoofdstuk worden de resultaten van de uitgewerkte realisaties besproken. De uitwerkingen zijn prototypes waarbij een beeld wordt gegeven hoe goed Pepper kan functioneren als zelfstandig vraagbak en assistent.

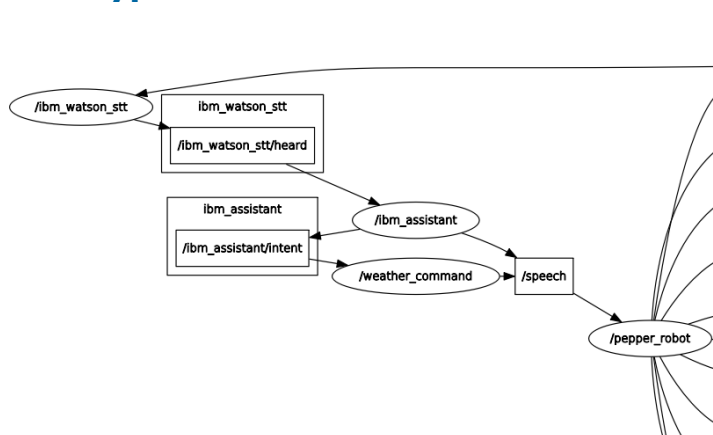
6.1 Prototype 1 - Request



Figuur 27: Prototype 1 - Request

Uit de eerste prototype, zie figuur 27, is gebleken dat de audiokwaliteit van Pepper de spraakherkenning belemmert. De motorgeluiden en de stem van Pepper worden opgenomen en verstuurd naar de spraakherkenning. De stem van Pepper blijft een probleem, omdat deze niet naar tekst hoeft te worden omgezet. De tijd om de audio om te zetten naar tekst duurt te lang en belemmert de ervaring om met Pepper op een natuurlijke manier te praten.

6.2 Prototype 2 - Socket



Figuur 28: Prototype 2 - Socket

Het tweede prototype, zie figuur 28, is een verbetering van het eerste prototype. Het is in staat om audio realtime om te zetten naar tekst. De tekst kan worden opgestuurd naar intelligente diensten, zoals naar IBM Watson, waar de bedoeling van de tekst wordt gecontroleerd. Het topic die door de IBM Assistant wordt gepubliceerd kan worden gebruikt door andere systemen. Zoals in figuur 28 te zien is kan een weather service worden gekoppeld aan IBM Assistant om het weer op te vragen.

De stem van Pepper wordt nog steeds opgenomen wat de performance belemmert. Dit is voor een kleine deel opgelost door een tijd mee te geven wanneer de audiodata niet hoeft te worden verstuurd naar IBM Watson speech-to-text. IBM Assistant heeft nog een dataset nodig wat hoort bij de servicedesk, zodat het getraind kan worden.

7 Conclusie

In dit onderzoek wordt de volgende vraag beantwoord: Hoe kan er op Pepper spraak worden omgezet naar tekst en daarvan de context worden bepaald? Het beantwoorden van deze vraag kan worden opgedeeld in drie delen: het opnemen van spraak, het omzetten van spraak naar tekst, en de context bepalen van de tekst.

Voor het opnemen van spraak is de positie van de gebruiker en Pepper belangrijk. Op het hoofd van Pepper zit er vier richtingsgevoelige microfoons die elk het geluid opnemen van de omgeving. Een gebruiker praat meestal voor het gezicht van Pepper. Spraak zal dan door de twee voorste microfoons het beste worden opgenomen.

De volgende stap is het verkrijgen van deze spraak als realtime audiodata, zodat het omgezet kan worden als tekst door een spraakherkenningssysteem. Hiervoor is ROS geschikt, omdat het de audiodata kan ophalen van de microfoons. Door een eigen gemaakte softwarepakket in ROS kan door nodes en topics de audiodata worden verstuurd naar IBM Watson Speech-To-Text.

De Noise gate node zorgt ervoor dat alleen geluid van een bepaald niveau kan worden opgenomen. Er hoeft niet constant overbodige ruis en achtergrondgeluiden verstuurd te worden naar IBM Watson. Er zal altijd rekening gehouden moeten worden met de omgeving waar Pepper zich bevindt. Geluiden boven de threshold van de noise gate belemmeren de performance van de spraakherkenning. Bij de servicedesk kan het geluidsniveau te hoog zijn en het is goed om hier bewust van te zijn als Pepper daar gebruikt zal worden.

De IBM Watson Speech-To-Text node is gemaakt als een socket die de connectie openhoudt tussen Pepper en IBM Watson. Hierdoor kan constant de audiodata worden verstuurd naar de IBM Watson service. Dit levert een snelle responsetijd op waardoor Pepper snel spraak om kan zetten naar tekst. Een probleem wat nog opgelost moet worden is dat naast stem van de gebruiker, ook de stem van Pepper wordt opgenomen. Dit belemmert de spraakherkenning, omdat de spraak van Pepper ook wordt omgezet naar tekst terwijl alleen die van de gebruiker nodig is.

Voor het bepalen van de context van de tekst kan IBM Watson Assistant gebruikt worden als onderdeel van Natural Language Understanding. IBM Watson Assistant vereist een dataset om getraind te worden. Dit ontbreekt nog, omdat er geen specifieke dataset is voor de servicedesk. Het gebruiken van andere Natural Language Understanding services kan helpen om de context beter te definiëren van een tekst.

Als de bovengenoemde problemen kunnen worden opgelost dan zal Pepper op het gebied van spraak beter kunnen functioneren als zelfstandig vraagbak en assistent. In het volgende hoofdstuk wordt hierover een aanbeveling gedaan.

8 Aanbeveling

Zoals in de conclusie uitgelegd moet er nog veel gewerkt worden aan Pepper om het als zelfstandig vraagbak en assistent te laten functioneren.

Ten eerste is het verstandig om, als ROS als een basis platform wordt gebruikt, de Rosbridge uit te werken. Hierdoor kunnen de andere functionaliteiten van Pepper gebruikt worden, zoals animaties afspelen tijdens het praten en de gebruiker aan kijken.

Ten tweede is de omgeving belangrijk. Het is misschien verstandig om Pepper een speciale stand te geven waar het net iets rustiger is dan bij de servicedesk. Er zou nog gekeken kunnen worden naar de Google Cloud Speech-to-Text socket implementatie, omdat Google Cloud net wat beter omgaat met ruis en ook al de Nederlandse taalmodule heeft.

Ten derde is een dataset nodig, zodat Pepper getraind kan worden om als zelfstandig vraagbak en assistent te functioneren. Het verzamelen van vragen en processen bij de service kan als dataset worden gebruikt door IBM Assistant.

Ten vierde zou het opzetten van een backend handig zijn om bepaalde informatie op te slaan en op te halen. Dit zou uitstekend met IBM Assistant kunnen worden gekoppeld.

Waar het op neer komt dat een basis platform nodig is voor Pepper waar gemakkelijke andere services en diensten erop aan kunnen worden aan gesloten. De intelligentie van Pepper wordt dan hiermee vergroot.

Verklarende woordenlijst

<i>Actuator</i>	Apparaat dat iets in beweging kan brengen
<i>Compileren</i>	Software voor het omzetten van computercode geschreven in een taal naar een andere programmeertaal
<i>Cross compiler</i>	Het compileren van code voor meerdere platformen dan alleen de ontwikkeling platform
<i>Fonemen</i>	De verschillende basisklanten die samen onze taal vormen
<i>Humanoïde</i>	Een wezen of machine waarvan de lichaamsstructuur lijkt op die van een mens.
<i>Node</i>	Een node is een proces die een berekening uitvoert
<i>Robot</i>	Een robot is een programmeerbare machine, die meerdere verschillende taken uit kan voeren.
<i>Robot Operating System (ROS)</i>	Een systeem voor het besturen van robot componenten vanaf een pc
<i>Sampling</i>	De hoogte van een geluidsgolf op gelijkmatige verspreide punten meten
<i>Service</i>	Een softwarefunctionaliteit of een set van software functionaliteiten met als doel dat verschillende clients het kunnen hergebruiken voor verschillende doeleinden
<i>Spectrogram</i>	De vingerafdruk van het geluid die wordt verkregen door na te gaan hoeveel energie elke frequentieband heeft
<i>Spraakherkenning</i>	Het proces van woorden en zinsdelen identificeren als iemand praat en het omzetten naar een digitaal leesbaar formaat
<i>Stemherkenning</i>	Het identificeren van de persoon die aan het praten is
<i>Topic</i>	De bericht die wordt uitgewisseld door een node

Bronnenlijst

- (sd). Opgehaald van Opensource spraakherkenning: <http://www.opensource-spraakherkenning.nl/>
- Anderson, K. S. (sd). *kristen_soltis_anderson_867488*. Opgehaald van brainyquote.com: https://www.brainyquote.com/quotes/kristen_soltis_anderson_867488
- AudioBuffer. (2017, Januari 20). Opgehaald van ROS: http://docs.ros.org/jade/api/naoqi_bridge_msgs/html/msg/AudioBuffer.html
- Autonomous Life. (sd). Opgehaald van Aldebaran: http://doc.aldebaran.com/2-5/family/pepper_user_guide/life_pep.html
- Balideli. (2014, 8 21). *De huidige rol van robots in de samenleving*. Opgehaald van Wetenschap info nu: <https://wetenschap.infonu.nl/techniek/127602-de-huidige-rol-van-robots-in-de-samenleving.html>
- Basic Awareness. (sd). Opgehaald van Aldebaran: <http://doc.aldebaran.com/2-5/naoqi/interaction/autonomousabilities/albasicawareness.html#albasicawareness>
- Basic Channel. (sd). Opgehaald van Aldebaran: http://doc.aldebaran.com/2-5/family/pepper_user_guide/basic_channel_conversation_pep.html
- Bell, T. (2018, Maart 1). Opgehaald van <https://www.cio.com/article/3258837/artificial-intelligence/what-is-natural-language-processing-the-business-benefits-of-nlp-explained.html>
- Biran, O. (2017, Augustus 24). *benchmarking*. Opgehaald van recast: <https://recast.ai/blog/benchmarking-speech-recognition-api/>
- Bluemix. (2018, Januari 1). Opgehaald van Conversation: <https://console.bluemix.net/docs/services/conversation/index.html#about>
- Bobriakov, I. (2018, April 7). *A Comparative Analysis of ChatBots APIs*. Opgehaald van Medium: <https://medium.com/activewizards-machine-learning-company/a-comparative-analysis-of-chatbots-apis-f9d240263e1d>
- Cellan-Jones, R. (2014, december 2). *technology*. Opgehaald van bcc: <https://www.bbc.com/news/technology-30290540>
- Choregraphe. (sd). Opgehaald van Aldebaran: http://doc.aldebaran.com/2-5/software/choregraphe/choregraphe_overview.html
- Cloud Speech-To-Text. (sd). Opgehaald van Google: <https://cloud.google.com/speech-to-text/>
- Company Information. (sd). Opgehaald van Atos: https://saleservice.myatos.net/overall/en/index.cfm?page=about_us
- Conley, K. (2012, 2 3). *Nodes*. Opgehaald van ROS: <http://wiki.ros.org/Nodes>
- Crowe, S. (2016, Januari 7). *IBM watson makes Pepper robots smarter*. Opgehaald van roboticsbusinessreview: https://www.roboticsbusinessreview.com/rbr/ibm_watson_makes_pepper_robots_smarter/
- Dernoncourt, F. (2018, Maart 30). *ASR benchmark*. Opgehaald van Github: https://github.com/Franck-Dernoncourt/ASR_benchmark
- Forouher, D. (2014, 6 1). *Topics*. Opgehaald van ROS: <http://wiki.ros.org/Topics>
- frequentie. (sd). Opgehaald van hoorzaken: <https://www.hoorzaken.nl/het-oor/geluid/frequentie-van-geluid/>
- Gruhn, Rainer E., Minker, Wolfgang, Nakamura, Satoshi. (2011). *Statistical Pronunciation Modeling for Non-Native Speech Processing*. Springer-Verlag Berlin Heidelberg.
- Hawkins, J. (2015, Maart 2). *The terminator is not coming*. Opgehaald van recode: <https://www.recode.net/2015/3/2/11559576/the-terminator-is-not-coming-the-future-will-thank-us>
- Hope, C. (2017, 12 29). *Voice recognition*. Opgehaald van computerhope: <https://www.computerhope.com/jargon/v/voicereco.htm>
- Izhaki, R. (2011, Februari 3). *The Usage Of Gates Explained*. Opgehaald van prosoundweb: https://www.prosoundweb.com/topics/studio/audio_concepts_the_usage_of_gates_explained/#
- Microphone. (sd). Opgehaald van Aldebaran: http://doc.aldebaran.com/2-4/family/pepper_technical/microphone_pep.html

- Monsters, D. (2017, Mei 11). *25 Chatbot Platforms: A Comparative Table*. Opgehaald van Chatbotsjournal: <https://chatbotsjournal.com/25-chatbot-platforms-a-comparative-table-aeeefc932eaff>
- N, K. (2015, 10 10). Opgehaald van ijarcse: http://ijarcse.com/Before_August_2017/docs/papers/Volume_5/10_October2015/V5I10-0228.pdf
- Organization*. (sd). Opgehaald van Atos: <https://source.myatos.net/thegroup/organization/Pages/default.aspx>
- Pepper*. (sd). Opgehaald van Softbank Robotics: <https://www.softbankrobotics.com/emea/en/robots/pepper>
- Pepper Technical overview*. (sd). Opgehaald van Aldebaran: http://doc.aldebaran.com/2-4/family/pepper_technical/index_pep.html
- Robotics, C. (2014, Januari 29). *ros-101-intro-to-the-robot-operating-system*. Opgehaald van <http://robohub.org>: <http://robohub.org/ros-101-intro-to-the-robot-operating-system/>
- Roeland van Oers, Eric Wesselman. (2016, Juni). *Social Robots*. Opgehaald van KPMG: <https://assets.kpmg.com/content/dam/kpmg/pdf/2016/06/social-robots.pdf>
- Rouse, M. (2016, December). *Speech recognition*. Opgehaald van techtarget: <https://searchcrm.techtarget.com/definition/speech-recognition>
- Soledad, Clément, Sandrine. (2017, September). *Pepper guidelines*. Opgehaald van aldebaran: http://doc.aldebaran.com/download/Pepper_B2BD_guidelines_Sept_V1.5.pdf
- Sophia*. (2017, oktober 25). Opgehaald van CNBC: <https://www.cnn.com/video/2017/10/25/watch-cnbc-sorkin-interview-a-lifelike-robot-named-sophia.html>
- Vervoort, D. (2017, September 27). *Spraakherkenning*. Opgehaald van techpulse: <https://www.techpulse.be/achtergrond/218906/hoe-werkt-spraakherkenningstechnologie/>
- Watson Speech-To-Text*. (sd). Opgehaald van IBM: <https://www.ibm.com/watson/services/speech-to-text/>
- Zhang, A. (sd). *SpeechRecognition*. Opgehaald van pypi: <https://pypi.org/project/SpeechRecognition/>

Figurenlijst

Tabel 1: Versiebeheer	6
Tabel 2: Databronnen	11
Tabel 3: AudioBuffer	19
Tabel 4: Soorten spraakherkenningssystemen	20
Tabel 5: Word Error Rate	22
Tabel 6: Request realisatie	31
Tabel 7: Socket realisatie	32
Figuur 1: Organogram Atos Nederland	8
Figuur 2: hardware overzicht	12
Figuur 3: Microfoons	13
Figuur 4: Choregraphie	14
Figuur 5: Architectuur	16
Figuur 6: ROS	17
Figuur 7: Audio topic	18
Figuur 8: Pepper Robot	19
Figuur 9: Sampling	21
Figuur 10: Spectrogram	21
Figuur 11: Neuraal netwerk	21
Figuur 12: Noise Gate	23
Figuur 13: Natural Language Processing	27
Figuur 14: Intent	28
Figuur 15: Entities	28
Figuur 16: Platform	28
Figuur 17: Channel Splitter Node	29
Figuur 18: Noise gate Node	29
Figuur 19: Noise gate configuratie	29
Figuur 20: Chunk builder node	30
Figuur 21: Chunk builder configuratie	30
Figuur 22: Speech to text node	30
Figuur 23: Recorder node	30
Figuur 24: Recorder configuratie	30
Figuur 25: IBM watson stt node	32
Figuur 26: IBM assistant node	32
Figuur 27: Prototype 1 - Request	33
Figuur 28: Prototype 2 - Socket	33

Bijlage A: Smart Robotics Project

Project

Projectnummer Projecttitel
1601 Ontwikkeling Smart Robotics

Projectgegevens

Projecttitel * Ontwikkeling Smart Robotics
Type project * Ontwikkelingsproject
Zwaartepunt * Programmatuur
Het project wordt/is gestart op * 01-05-2016

Geef een algemene omschrijving van het project. Heeft u eerder WBSO aangevraagd voor dit Project? Beschrijf dan de stand van zaken bij de vraag "Updateproject". *
(Maximaal 1.500 tekens)

Robotica ontw. Zich sterk de laatste jaren. Atos heeft 2 kleine robots (type Aldebaran Robotics Nao H25 humanoïde robot) aangeschaft om klanten op een innovatieve manier te verrassen. Zij zijn voorzien van lichaam motie controles, spraakherkenning, geluid lokalisatie, gezicht en beeldherkenning. Zij hebben beperkte processorcapaciteit, waardoor mogelikh. En interactie met mens beperkt mogelijk is. Dmv additionele progr.ontw. wil Atos funct. v/d robots aanzienlijk uit breiden. In dit project ontw. Atos een 'Smart Robotics' opl., waarbij robots taken en aanwezigheid v/d mens over kunnen nemen. Deze opl. zijn techn. Volledig nieuw voor Atos en de wereld. In het project wordt gefaseerd toe gewerkt naar de volgende doelst. Robot als:

1. Volwaardige vraagbaak met fysieke vaste locatie incl. echte conversatievoering (Q&A en vervolg antwoorden door daadwerkelijke conversatieherkenning);
2. 'Pick up' assistant waarbij de robot obv beschikbare gegevens kan constateren of een medewerker van Atos aanwezig is, verstand heeft van gevraagde zaken en in staat is de bezoekende personen te begeleiden naar de betreffende Atos medewerker/gespreklocatie en afspraken maken;
3. Co-presence opl. bijv. tijdens meetings. Idee is dat de robot de persoon in kwestie (die op afstand zit) letterlijk kan vervangen dmv robot met 360 graden camera, VR en nabootsing van de positie van het lichaam (bijv. obv Kinect).

In 2017 wordt in eigen beheer gecontinueerd met doelst. 1 en gestart met doelst. 2.

Samenwerking

Leverd een of meer partijen (buiten uw fiscale eenheid) een bijdrage aan het project? *
Ja
Nee

Fasering werkzaamheden

Ontwikkelings-/ onderzoeksactiviteit * Datum gereed *
Omschrijf per fase uw eigen S&O-werkzaamheden (S&O-werkzaamheden van het bedrijf waarvoor u deze WBSO-aanvraag indient). Uit de fasering moet ook de vermoedelijke einddatum van het S&Oproject blijken.

WBSO 2018 6 / 40

Opstellen techn. specs Doorlopend
Technisch ontwerp Doorlopend
Ontwikkelen proto's Gereed
Testen proto's 31-12-2017
Herontwikkelingen 31-03-2018
Testen proto's 30-06-2018
Eindontwikkeling 31-08-2018

Update project

Vermeld de voortgang van uw S&O-werkzaamheden. Zijn er wijzigingen in de oorspronkelijke projectopzet of -planning? Geef dan aan waarom dit het geval is.
(maximaal 1.500 tekens)

Project loopt conform planning. Status per deelontwikkeling:

1) Informatievoorziening: Atos heeft een Pepper robot aangeschaft en haar 1e basisscripten geprogrammeerd voor het aanhoren, interpreteren en beantwoorden van vragen. Technische werking hiervan is dmv testen aangetoond. Vervolgstep is ontwikkeling obv Natural Language. Hierbij moet de robot mbv AI leren verschillende vragen aan te horen, te interpreteren en te beantwoorden. Gebruik v/d tablet v/d robot is op dit moment nog de centrale functie. Het is technisch lastig om de robot verschillende dialecten te laten herkennen. Ruis en omgevingsgeluiden bemoeilijken ook het opnemen en interpreteren v/d vraag. Bijkomend TK is de vraag opnemen, naar de cloud sturen, antwoord vanuit de cloud naar de robot sturen, waarbij het proces snel verloopt en tegelijk natuurlijk overkomt. In Q1 2018 ligt de focus op de ontwikkeling v/h platform waarmee de robot (later meerdere robots) kan communiceren. 2) Pick-up assistant: Eind 2017 heeft Atos een koppeling ontwikkeld tussen de robot en Exchange server zodat de robot een afspraak kan verifiëren tijdens ontvangst van gasten. Daarnaast wil Atos de robot indoor kunnen laten navigeren en gasten dmv handgebaren de weg kunnen wijzen. TK hierbij is het veilig indoor manoeuvreren v/d robot waarbij oa rekening moet worden gehouden met div. mate van lichtintensiteit (zonlicht, kunstlicht,

schaduw). Voor Q1 2018 is een aantal onderwerpen geselecteerd voor pilottesten, zoals indoor navigation.

1. Technische knelpunten programmatuur

Geef aan welke concrete technische knelpunten u zelf tijdens het ontwikkelen van de programmatuur moet oplossen om het gewenste projectresultaat te bereiken. Vermeld geen aanleidingen, algemene randvoorwaarden of functionele eisen van de programmatuur.* (max 1.500 tekens)

Te ontw. SW-componenten:

1. Geintegr. spraak systeem & vraagbaak SW en conversatie parameter manager tool

2. Indoor navigatie en locatie mapping en configuratie opl. tbv beheer "aflever" punten

3. Receiver SW; Video en audio streaming for robots en remote aansturing van ledematen en scharnietpunten. Controller SW: walking dmv Virtuix

Omni en lichaamstaal dmv Kinect.

Techn. problemen bij de ontw. v/d componenten zijn:

1. Conversatieherkenning (stemherkenning en vervolgvraagherkenning). Conversatie voeren (groep vragen die op elkaar door bouwen) boom

structuur versus vrije vorm? Hoe kan één specifiek antwoord worden gegeven met een enorme hoeveelheid beschikbare data? Hoe om te gaan met

ruis doordat meerdere door elkaar praten? Hoe om te gaan met meerdere personen (stem herkenning/gezichtsherkenning)?

2. Constante positie bepaling en positionering in een pand icm ontwijken van obstakels en andere personen.

3. Schaalbaarheid/performance/latency problemen ivm de grote hoeveelheid data bij 360 graden videobeelden en VR icm nabootsing lichaamsbewegingen. Belangrijke factoren daarbij zijn: bandbreedte, kwaliteit en snelheid. Locatie bepaling en omgevings mapping; hoe weet het

systeem in de robot zijn eigen locatie te bepalen in een indoor situatie?

Integratie en performance diverse hardware componenten; gaat de afhandeling van de verschillende hardware leiden tot comprissen in performance

en hoe groot is de impact op performance en bruikbaarheid.

Probleemstelling en oplossingsrichting

2. Oplossingsrichtingen programmatuur

Geef voor ieder genoemd technisch knelpunt aan wat u specifiek zelf gaat ontwikkelen om het knelpunt op te lossen. * (maximaal 1.500 tekens)

1. Ontw. opl. waarbij spraak herkend wordt en omgezet wordt in tekst. Deze tekst vormt het uitgangspunt voor conversatie herkenning.

Idee is ontw.

van patroonherkenning en conversatie algoritmes, verrijkt met een kennisbasis van mogelijke onderwerpen om zo een doorlopende conversatie tussen

mens en robot te faciliteren.

2. Positioneren in pand, beacons, WiFi spot voor plaatsbepaling. Robot dient zijn eigen positie ten opzichte van verschillende ruimtes in een kantoor

pand bepalen. Idee is opl. te ontwikkelen met een triangulatie opzet, gefaciliteerd door iBeacon, ultrasoon of WiFi technologie. Deze locatie punten

worden aangekoppeld op een virtuele kaart van een pand zodat de robot "weet" waar hij is en waar hij een bezoeker heen zou moeten leiden. Hiertoe

ontw. van een appl. welke autom. beacon punten op een CAD tekening van een kantoor omgeving kan mappen.

3. Schaalbaarheid/performance/latency: User experience is leidend in de opl. niet zozeer het meest effectief uitbaten v/d hardware (ivm VR is user

experience meest belangrijk ivm bijeffect als misselijkheid). Idee is de opl. in optimale schaalbaarheid/performance en te ontw. door perfecte balans

tussen netwerk bandbreedte, optimalisatie data laod en performance verbetering distribueren van reken taken, ie. berekeningen aan server side of

client side uitvoeren. Dit zal constant getest moeten worden. Idee is hiertoe een log analyse tool te ontw. welke automatisch HW en SW test, maar

ook direct benchmark/vergelijkt.

WBSO 2018 7 / 40

3. Programmeertalen, ontwikkelomgevingen en tools

Geef aan welke programmeertalen, ontwikkelomgevingen en tools u gebruikt bij de ontwikkeling van technisch nieuwe programmatuur. * (maximaal 1.500 tekens)

Technische nieuwheid programmatuur

Programmeer talen & software pakketten: C# / Virtual Studio, Unity, Naoqi / Choreograph, Python

Hardware: Aldebaran Nao, Virtuix Omni, Oculus Rift, Microsoft Kinect

4. Technische nieuwheid

Geef aan waarom de hiervoor genoemde oplossingsrichtingen technisch nieuw voor u zijn.

Oftewel beschrijf waarom het project technisch vernieuwend en uitdagend is en geef aan welke technische risico's en onzekerheden u hierbij verwacht. Om technische risico's en onzekerheden in te schatten kijkt RVO.nl naar de stand van de technologie. * (maximaal 1.500 tekens)

Speech-to-tekst conversaties, locatie bepaling en human gesture bepaling.

Bijlage B: Interview

Interviewer	Ahmed Tyghri
Omschrijving opname	Interview
Tijdsduur	10 minute
Bestandsnaam	Interview_Tom_Verloop_2018_06_29
Soort transcriptie	Globale transcriptie De transcriptieschrijver heeft de opname eenmalig beluisterd en tijdens het beluisteren de hoorbare tekst zo goed mogelijk uitgetypt. Er is <i>niet</i> gestreefd naar volledigheid. Passages die net meteen goed verstaanbaar waren, zijn overgeslagen. Niet-relevante tekst (zoals tussenwerpsels, gestotter, herhalingen) is weggelaten.
Uitvoering	Atos Amstelveen
Geïnterviewde	Tom Verloop

Vraag 1: **Wie bent u en wat is uw functie binnen Atos?**

Mijn naam is Tom Verloop, ik ben de laatste maanden bezig geweest met mijn afstuderen. Vandaag is ook mijn laatste dag en ik heb sinds november gewerkt aan de Pepper robot in combinatie met ROS.

Vraag 2: **Waarom is gekozen om in combinatie met ROS te werken?**

ROS is gekozen om een aantal redenen. De belangrijkste rede is dat alle functionaliteiten die we voor de robot willen schrijven niet gelinkt is aan de robot. ROS is niet afhankelijk aan de soort robot. ROS is geschreven voor alle softwareontwikkeling voor robots en dat zijn de pakketten van de Pepper robot niet. Daarom wilde we het gebruiken van die pakketten zo min mogelijk houden.

Vraag 3: **Is deze opdracht vanuit Atos gekomen of vanuit u zelf?**

Toen ik hier begon was het idee om verder te werken aan wat we al hebben. Ik heb in twee weken tijd alle documentaties over de Pepper robot doorgelezen. Ook de ontwikkelsoftware geanalyseerd en mijn conclusie was dat het beter was om een nieuw platform/ basis te beginnen waarbij ROS aan het daglicht kwam. Dit was niet het idee van Atos, dit heb ik zelf bedacht en daarop is ingestemd.

Vraag 4: **Kan u wat problemen opnoemen waar u tegen aanliep voor dit besluit?**

Heel veel functionaliteiten van Pepper, ik weet even geen een te bedenken, maar veel dingen worden in de documentatie van Pepper beschreven dat het van alles kan, maar als je het dan gaat proberen en uiteindelijk de bug logs door kijkt, kom je erachter dat heel veel eigenlijk niet werkt. Dat is ten eerste jammer

Vraag 5: ROS staat dus nu geïnstalleerd op een server. Wat kan vanaf nu ermee verder worden gedaan?

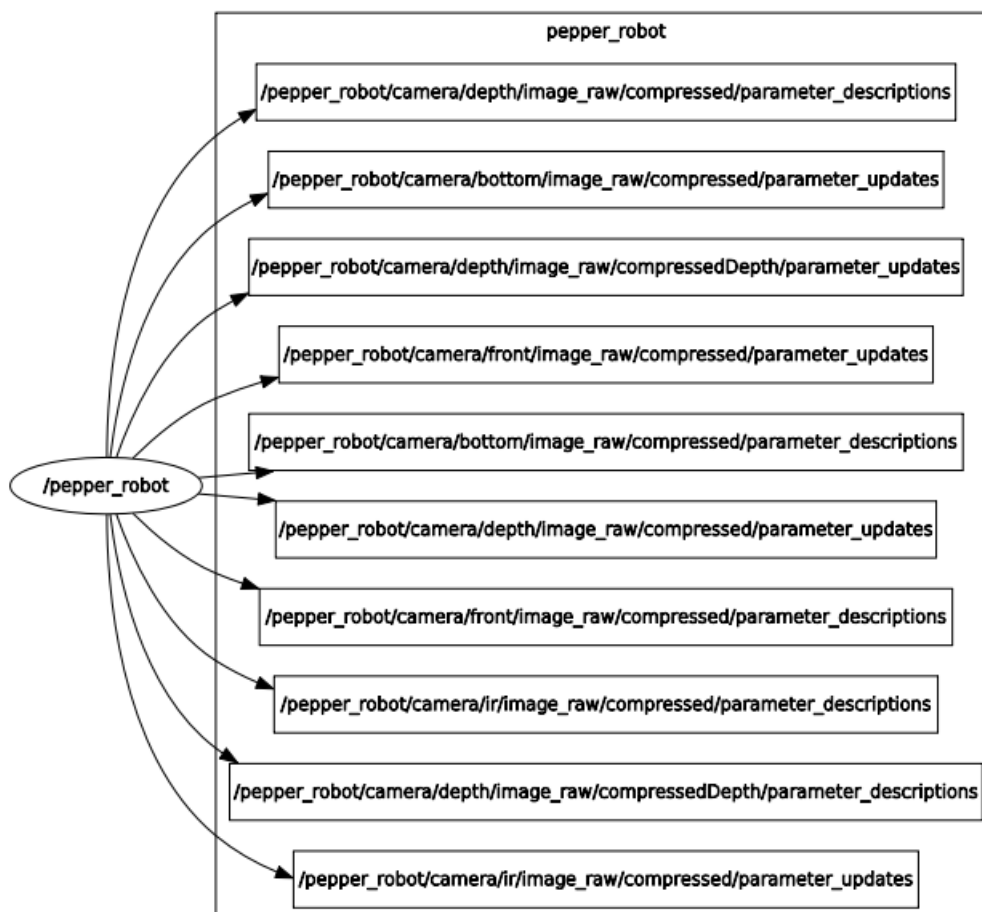
Door ROS te gebruiken kunnen we abstracte functionaliteiten voor de robot schrijven. Wat niet direct gelinkt is aan de robot. Stel je wil de robot iets laten doen, iets iemand laten weten. Afhangend van het soort robot kan het op vele manieren. Het idee is dat aan de hoogste laag gezegd wordt, laat iets weten en de robot interpreteert dat in zijn eigen manier. Kan de robot geluid afspelen, iets uitspreken, heeft die alleen een scherm en zo voort. Je probeert zo min mogelijk vast te zitten aan de robot die je hebt. Daarnaast komt ROS nog met paar andere voordelen. Ten eerste is er met ROS heel veel van dit soort functionaliteiten al gemaakt. Zoals remote control, mapping, object detection doormiddel van de camera, ook geeft het een gedistribueerd model, waarbij rekenintensieve functionaliteit buiten de robot kan worden geplaatst. Ergens waar rekenkracht is, in plaats van dat kleine CPU van de robot.

Vraag 6: U bent er dus helemaal voor om door te gaan met ROS voor Pepper?

Dat licht vooral aan wat je wil bereiken ermee. Wat het probleem is dat er een tussen laag nodig is die de softwarepakketten van Pepper vertaalt naar ROS en terug. Om dit goed te doen, moet ROS op de Pepper robot draaien, waardoor we dus ROS moeten cross compileren voor de architectuur van de robot

Bijlage C: Nodes & Topics

Pepper Node:



Pepper Topics:

```

ahmed@ahmed-VirtualBox:~$ rostopic list
/chunk_builder/parameter_descriptions
/chunk_builder/parameter_updates
/cmd_vel
/diagnostics
/joint_angles
/joint_states
/move_base_simple/goal
/noise_gate/parameter_descriptions
/noise_gate/parameter_updates
/pepper_robot/audio
/pepper_robot/bumper
/pepper_robot/camera/bottom/camera_info
/pepper_robot/camera/bottom/image_raw
/pepper_robot/camera/bottom/image_raw/compressed
/pepper_robot/camera/bottom/image_raw/compressed/parameter_descriptions
/pepper_robot/camera/bottom/image_raw/compressed/parameter_updates
/pepper_robot/camera/bottom/image_raw/compressedDepth/parameter_descriptions
/pepper_robot/camera/bottom/image_raw/compressedDepth/parameter_updates
/pepper_robot/camera/depth/camera_info
/pepper_robot/camera/depth/image_raw
/pepper_robot/camera/depth/image_raw/compressed
/pepper_robot/camera/depth/image_raw/compressed/parameter_descriptions
/pepper_robot/camera/depth/image_raw/compressed/parameter_updates
/pepper_robot/camera/depth/image_raw/compressedDepth
/pepper_robot/camera/depth/image_raw/compressedDepth/parameter_descriptions
/pepper_robot/camera/depth/image_raw/compressedDepth/parameter_updates
/pepper_robot/camera/front/camera_info
/pepper_robot/camera/front/image_raw
/pepper_robot/camera/front/image_raw/compressed
/pepper_robot/camera/front/image_raw/compressed/parameter_descriptions
/pepper_robot/camera/front/image_raw/compressed/parameter_updates
/pepper_robot/camera/front/image_raw/compressedDepth/parameter_descriptions
/pepper_robot/camera/front/image_raw/compressedDepth/parameter_updates
/pepper_robot/camera/ir/camera_info
/pepper_robot/camera/ir/image_raw
/pepper_robot/camera/ir/image_raw/compressed
/pepper_robot/camera/ir/image_raw/compressed/parameter_descriptions
/pepper_robot/camera/ir/image_raw/compressed/parameter_updates
/pepper_robot/camera/ir/image_raw/compressedDepth/parameter_descriptions
/pepper_robot/camera/ir/image_raw/compressedDepth/parameter_updates
/pepper_robot/hand_touch
/pepper_robot/head_touch
/pepper_robot/imu/base
/pepper_robot/imu/torso
/pepper_robot/info
/pepper_robot/laser
/pepper_robot/odom
/pepper_robot/sonar/back
/pepper_robot/sonar/front
/recorder/parameter_descriptions
/recorder/parameter_updates
/rosout
/rosout_agg
/speech
/statistics
/tf
ahmed@ahmed-VirtualBox:~$

```

Bijlage D: Omgevingsfactoren

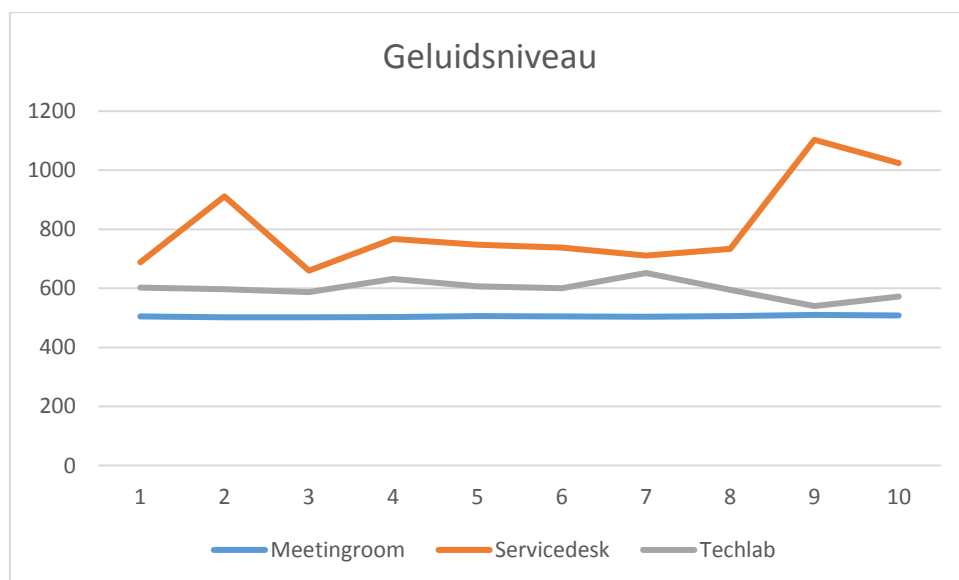
Toelichting:

Het geluidsniveau is gemeten met de microfoons van Pepper in drie verschillende omgevingen. Voor het uitrekenen van het geluidsniveau is de RMS-formule gebruikt.

Metingresultaten:

Tijd(min)	Meetingroom	Servicedesk	Techlab
1	505	688	602
2	502	912	597
3	502	660	587
4	503	767	632
5	506	748	607
6	505	738	600
7	504	711	652
8	506	734	595
9	510	1103	540
10	508	1024	572
Gemiddelde:	505	809	598

Geluidsniveau:



Bijlage E: Benchmarks

```

ATygh@DESKTOP-NF735US MINGW64 /d/Projects/ASR_benchmark-master/src
$ python benchmark.py
asr_systems: ['google', 'ibm']
data_folders: ['./data/example_dataset_en']

Working on data folder "../data/example_dataset_en"
Detected speech file type: wav

### Call the ASR engines to compute predicted transcriptions

Google Speech Recognition:
asr_time_elapsed: 1.869 seconds
transcription: what are the dataset the articles useless

IBM Speech Recognition:
asr_time_elapsed: 4.520 seconds
transcription: without the data set the article is useless

Google Speech Recognition:
asr_time_elapsed: 0.749 seconds
transcription: I've got to go to him

IBM Speech Recognition:
asr_time_elapsed: 2.669 seconds
transcription: I've got to go to him

Google Speech Recognition:
asr_time_elapsed: 0.836 seconds
transcription: and you know it

IBM Speech Recognition:
asr_time_elapsed: 2.976 seconds
transcription: and you know it

Google Speech Recognition:
asr_time_elapsed: 2.088 seconds
transcription: down below in the darkness where hundreds of people sleeping in peace

IBM Speech Recognition:
asr_time_elapsed: 6.201 seconds
transcription: down below in the darkness where hundreds of people sleeping in peace

### Final evaluation of all the ASR engines based on their predicted jurisdictions
google wer: 17.24138% (deletions: 1 ; insertions: 1 ; substitutions: 3 ; number_of_tokens_in_gold = 29)
Number of speech files: 4
Number of missing predicted prescription files: 0
Number of empty predicted prescription files: 0
ibm wer: 10.34483% (deletions: 0 ; insertions: 1 ; substitutions: 2 ; number_of_tokens_in_gold = 29)
Number of speech files: 4
Number of missing predicted prescription files: 0
Number of empty predicted prescription files: 0

ATygh@DESKTOP-NF735US MINGW64 /d/Projects/ASR_benchmark-master/src
$ |

```



```
A689855@MP145KQX MINGW64 ~/Desktop/ASR_benchmark-master/src
$ python benchmark.py
asr_systems: ['google', 'ibm']
data_folders: ['../data/example_pepper']

Working on data folder "../data/example_pepper"
Detected speech file type: wav

### Call the ASR engines to compute predicted transcriptions

Google Speech Recognition:
asr_time_elapsed: 2.075 seconds
transcription: hello can you help me I'm searching for a technology that do you know where it is

IBM Speech Recognition:
asr_time_elapsed: 8.393 seconds
transcription: can you help me I am searching for a technology that do you know where it is

Google Speech Recognition:
asr_time_elapsed: 3.433 seconds
transcription: hello can you help me I'm searching for the technology lab do you know where it is

IBM Speech Recognition:
asr_time_elapsed: 11.576 seconds
transcription: hello can you help me search for the technology lab do you know where it is

Google Speech Recognition:
asr_time_elapsed: 2.866 seconds
transcription: hello can you help me I am searching for the technology let do you know where it is

IBM Speech Recognition:
asr_time_elapsed: 10.901 seconds
transcription: hello can you help me I am searching for the technology that the you know where this

### Final evaluation of all the ASR engines based on their predicted jurisdictions
google wer: 12.96296% (deletions: 2 ; insertions: 0 ; substitutions: 5 ; number_of_tokens_in_gold = 54)
Number of speech files: 3
Number of missing predicted prescription files: 0
Number of empty predicted prescription files: 0
ibm wer: 18.51852% (deletions: 4 ; insertions: 0 ; substitutions: 6 ; number_of_tokens_in_gold = 54)
Number of speech files: 3
Number of missing predicted prescription files: 0
Number of empty predicted prescription files: 0

A689855@MP145KQX MINGW64 ~/Desktop/ASR_benchmark-master/src
$ |
```

Bijlage F: Metingen

Datum	29-06-2018
Doel	Het testen van twee IBM-speech-to-text implementaties in python.
Uitleg	<p>Implementatie 1: Bouwt een klein audio stukje van 3 seconde en verstuurt deze via een request naar IBM.</p> <p>Implementatie 2: Verstuurt elke audiodata direct door naar IBM via een socket</p>
Omgeving situatie	De test werd uitgevoerd in een stille vergaderruimte. De spreker stond een stap voor Pepper en keek het aan in het gezicht.

Implementatie 1 – Request

Nr.	Tekst	STT	Pogingen	Response
1	hello	hello	2	2.33
2	how are you today	how are you today	4	2.60
3	what is your name	what is your name	3	2.40
4	thank you	thank you	1	2.36
5	can you hear me	can you hear me	1	3.06
6	where are the toilets	where are the toilets	3	2.71
7	that is correct	that is correct	1	2.33
8	that is wrong	that is wrong	1	2.79
9	I have an appointment	I have an appointment	6	3.60
10	goodbye	goodbye	2	3.01

Implementatie 2 - Socket

Nr.	Tekst	STT	Pogingen	Response
1	hello	hello	1	0.20
2	how are you today	how are you today	1	0.20
3	what is your name	what is your name	1	0.20
4	thank you	thank you	1	0.25
5	can you hear me	can you hear me	1	0.18
6	where are the toilets	where are the toilets	1	0.21
7	that is correct	that is correct	1	0.08
8	that is wrong	that is wrong	1	0.17
9	I have an appointment	I have an appointment	1	0.12
10	goodbye	goodbye	1	0.40

Berekeningen

Implementatie 1:

Gemiddelde pogingen: 2
Gemiddelde tijd: 2,71 seconde

Implementatie 2:

Gemiddelde pogingen: 1
Gemiddelde tijd: 0.2 seconde

Aantal keer sneller = $\frac{2,71s}{0,2s} = 13,55$
Tijdwinst = $2,71 - 0,2 = 2,51$ seconde

Conclusie

Voor de request implementatie is de gemiddelde tijd 2,71 seconde en kost het ongeveer 2 pogingen om iets goed om te zetten naar tekst. Voor de socket implementatie is dat 0.2 seconde en kost het 1 poging. De socket implementatie is 13,55 sneller dan de request implementatie. De gemiddelde tijd die wordt gewonnen is 2,51 seconde.