

Hogeschool van Arnhem en Nijmegen – NotS Project

Onderzoeksrapport

Conversational User Interfaces



Gemaakt voor: Hogeschool van Arnhem en Nijmegen

Gemaakt door: Kevin Aarns (554012)

Lucas Bos (562527)

Mike van Essen (552584)

Miguel Snelder (555115)

Jeroen Spans (547442)

Niels Wijers (564429)

Professional Skills: Jeannette Danes

Begeleider: Martijn Driessen

Semester: .Net on the Server (NotS)

Versie: 1.9.6

Datum: 13 juni 2018

Voorwoord

Voor u ligt het onderzoeksrapport “Conversational User Interfaces”. Dit onderzoeksrapport is geschreven in kader van het NotS project voor de opleiding Informatica aan de Hogeschool van Arnhem Nijmegen.

Van 5 februari 2018 tot en met 15 juni 2018 zijn wij bezig geweest met het onderzoeken en schrijven van dit onderzoeksrapport.

Wij bedanken de heer Driessen (begeleider) en mevrouw Danes (professional skills docent) die tijdens dit project ons hebben begeleid.

Veel plezier met het lezen van dit onderzoeksrapport!

De CUI, Arnhem, 13 juni 2018

Samenvatting

De overheid wil dat patiënten minder snel het ziekenhuis bezoeken, maar eerder contact opnemen met de huisarts. Dit heeft als gevolg dat ongeveer tachtig procent van de huisartsen symptomen vertoont van overspannenheid (Maes, 2017). Één van de oorzaken hiervoor is dat 29 procent van de patiënten onnodig naar de huisarts gaat, wat resulteert in overvolle agenda's (AVRO Tros, 2017) (Buskoop, 2017)

Het doel van dit onderzoek is om het aantal patiënten dat onnodig naar de huisarts gaat te verminderen. Dit gebeurt aan de hand van de volgende vraag: "Hoe kan een chatbot de ernst van een medische klacht classificeren?". Wanneer bekend is hoe de ernst geclassificeerd kan worden, kan een assistent bepalen of het noodzakelijk is om de patiënt te ontvangen. Dit zou de werkdruk van de huisarts moeten verlagen door minder onnodige bezoeken van patiënten.

Voor dit onderzoek is eerst gekeken naar een geschikt chatbot framework, in combinatie met een geschikte Natural Language Processing (NLP) tool. Dit gebeurde door criteria op te stellen en meerdere prototypes te maken. Nadat de keuze is gemaakt, is begonnen aan de chatbot. Hierbij zijn de problemen die het projectteam onderzocht en op basis van het onderzoek de oplossingen geïmplementeerd.

Het resultaat is dat het projectteam kennis heeft opgedaan over chatbots en op basis hiervan een prototype heeft gerealiseerd. Dit prototype kan een conversatie houden en enkele klachten classificeren. Binnen dit prototype wordt de ernst niet gemeten, maar wordt op basis van de klacht advies gegeven. Hieruit blijkt dat een chatbot technisch de mogelijkheid biedt om te helpen bij het oplossen van de problemen.

Dat het technisch mogelijk is om de chatbot te realiseren betekent niet dat het probleem is opgelost. Voordat de implementatie volledig gebruikt kan worden moet onderzoek gedaan worden naar verschillende onderwerpen. Een tweetal voorbeelden hiervan zijn:

- Privacy waarborging
- Medisch beroepsgeheim

Inhoudsopgave

Voorwoord.....	2
Samenvatting.....	3
Inhoudsopgave.....	4
1. Afkortingenlijst.....	8
2. Figuren en Tabellen	9
2.1. <i>Figuren.....</i>	9
2.2. <i>Tabellen</i>	10
3. Inleiding.....	11
4. Onderzoeks methode	12
4.1. <i>Methoden</i>	12
4.2. <i>Soorten onderzoek</i>	13
5. Uit welke onderdelen is een chatbot opgebouwd?	14
5.1. <i>Introductie</i>	14
5.2. <i>Network architecture overview.....</i>	14
5.3. <i>Webserver.....</i>	17
5.3.1. <i>API.....</i>	17
5.3.2. <i>Chatbot applicatie.....</i>	17
5.3.3. <i>Database</i>	17
5.4. <i>Chatbot.....</i>	17
5.4.1. <i>Platformen</i>	17
5.4.2. <i>Framework connector.....</i>	17
5.4.3. <i>Cognitieve diensten</i>	17
5.4.4. <i>Natural language processing</i>	17
5.4.5. <i>Image recognition.....</i>	18
5.4.6. <i>Speech recognition.....</i>	18
5.4.7. <i>Websearch</i>	18
5.4.8. <i>Translation</i>	18
6. Welke onderdelen zijn relevant voor dit onderzoek?.....	19
6.1. <i>De relevante onderdelen</i>	19
6.1.1. <i>Platform</i>	19
6.1.2. <i>Framework connector.....</i>	19
6.1.3. <i>Webserver – Chatbot applicatie</i>	19
6.1.4. <i>Webserver – API</i>	19
6.1.5. <i>Webserver – Database</i>	20
6.1.6. <i>Cognitieve diensten</i>	20
6.2. <i>De werking van de relevante onderdelen</i>	20
6.2.1. <i>Chatbot applicaties</i>	20
6.2.2. <i>Natural Language Processing</i>	20

6.3. <i>Conclusie</i>	21
7. Welke tools zijn beschikbaar om de onderdelen te realiseren?	22
7.1. <i>Criteria</i>	22
7.2. <i>Chatbot applicatie tools</i>	22
7.3. <i>NLP tools</i>	24
7.4. <i>Onderzoek tools</i>	25
7.5. <i>Chatbot applicatie tool - Hubot</i>	25
7.5.1. <i>Conclusie</i>	26
7.6. <i>Chatbot applicatie tool – Microsoft Bot Framework</i>	26
7.6.1. <i>Wat is het Microsoft Bot Framework?</i>	26
7.6.2. <i>Soorten chatbots</i>	26
7.6.3. <i>Commerce chatbot</i>	26
7.6.4. <i>Informatie chatbot</i>	26
7.6.5. <i>Enterprise chatbot</i>	27
7.6.6. <i>Microsoft Bot Framework onderdelen</i>	27
7.6.6.1. <i>Microsoft Bot Builder</i>	27
7.6.6.2. <i>Microsoft Cognitieve Services</i>	27
7.6.6.3. <i>Microsoft Azure Bot Service</i>	28
7.6.7. <i>Hoe werkt het Microsoft Bot Framework?</i>	28
7.6.8. <i>.NET SDK</i>	29
7.6.9. <i>Node.js SDK</i>	31
7.6.10. <i>Server(Express, Restify)</i>	31
7.6.11. <i>Chatconnector</i>	31
7.6.12. <i>Listen voor berichten</i>	31
7.6.13. <i>Response of message</i>	32
7.6.14. <i>Chatbot emulator</i>	32
7.6.15. <i>Conclusie</i>	33
7.7. <i>NLP tool – Dialogflow</i>	33
7.7.1. <i>Wat is Dialogflow?</i>	33
7.7.2. <i>Hoe werkt Dialogflow?</i>	33
7.7.3. <i>Conclusie</i>	36
7.8. <i>Conclusie</i>	36
8. Welke combinatie van tools is het meest kansrijk voor het realiseren van een chatbot?.....	37
8.1. <i>Hubot</i>	37
8.1.1. <i>Wit.ai</i>	37
8.1.2. <i>Dialogflow</i>	38
8.2. <i>Microsoft Bot Framework</i>	38
8.2.1. <i>Wit.ai</i>	38
8.2.2. <i>Dialogflow</i>	39
8.3. <i>Voor- en nadelen</i>	40
8.3.1. <i>Frameworks</i>	41
8.3.2. <i>Hubot</i>	41
8.3.3. <i>Microsoft Bot Framework</i>	41
8.3.4. <i>NLP tools</i>	42
8.3.5. <i>Dialogflow</i>	42
8.3.6. <i>Wit.ai</i>	42

8.4. <i>Conclusie</i>	42
9. Welke taken kan NLP uitvoeren?	44
9.1. <i>Natural Language Processing</i>	44
9.2. <i>De onderdelen van Natural Language Processing</i>	44
9.2.1. Tokenization	45
9.2.2. Stop word removal	46
9.2.3. N-Grams	46
9.2.4. Word sense disambiguation.....	47
9.2.5. Part Of Speech Tagging	47
9.2.6. Verkleinen van woorden	47
9.2.7. Toepassing.....	48
Rule-based approach	48
9.3. <i>Taken die uitgevoerd kunnen worden door Dialogflow</i>	50
9.3.1. Named entity recognition	50
9.3.2. Spellingscontrole	51
9.4. <i>Conclusie</i>	55
10. Hoe houdt de chatbot een conversatie met de gebruiker?	56
10.1. <i>Conversatie methoden</i>	56
10.1.1. Waterfall	56
10.1.2. Intent based	56
10.1.3. Flow based	57
10.1.4. Combinatie.....	57
10.2. <i>Decision algoritmes</i>	57
10.2.1. Tree.....	57
10.2.2. Graph	58
10.3. <i>Implementatie</i>	59
10.3.1. Flowchart	59
10.3.2. Implementatie flowchart	61
10.4. <i>Conclusie</i>	66
11. Meetup bij KLM digital studio.....	67
11.1. <i>Verwachting meetup</i>	67
11.2. <i>Verkregen informatie</i>	67
11.2.1. Groeperen van entiteiten.....	67
11.2.2. Opbouw chatbot.....	68
11.2.3. Context.....	69
11.3. <i>Conclusie</i>	69
12. Conclusie	70
13. Adviesrapport.....	71
13.1. <i>Bepaal het doel van de chatbot</i>	71
13.2. <i>Maak de nodige onderdelen duidelijk</i>	71
13.3. <i>Kies de conversatie methode</i>	72

Bibliografie.....	73
Bijlage A: Lijst van stopwoorden	79
Bijlage B: Prototype.....	79

1. Afkortingenlijst

In dit hoofdstuk worden de afkortingen die tijdens het onderzoek gebruikt worden beschreven.

Afkorting	Geheel
API	Application Programming Interface
NLP	Natural Language Processing
HTTP	Hypertext Transfer Protocol
SDK	Software developer kit
URL	Uniform Resource Locator
OO	Object Oriented
CUI	Conversational User Interface
JSON	JavaScript Object Notation
IDE	Integrated Development Environment
REST	Representational state transfer
NPM	Node Package Manager
MSBF	Microsoft Bot Framework

Tabel 1: Afkortingenlijst

2. Figuren en Tabellen

2.1. Figuren

Figuur 1: HBO-i methode toolkit.....	12
Figuur 2: Essentiële onderdelen van een chatbot.....	14
Figuur 3: Een chatbot bruikbaar voor gebruikers.....	15
Figuur 4: Een potentieel multi-platform chatbot.....	15
Figuur 5: Een chatbot met opslag	16
Figuur 6: Een intelligente chatbot.....	16
Figuur 7: Het starten van Hubot	25
Figuur 8: Hubot code	25
Figuur 9: Hubot reactie.....	25
Figuur 10: .NET Bot Buidler templatebestanden	29
Figuur 11: MessageController klasse	29
Figuur 12: RootDialog Klasse.....	30
Figuur 13: Het starten van de Node.js chatbot	31
Figuur 14: Restify server setup	31
Figuur 15: MSBF chatconnector.....	31
Figuur 16: Listen to endpoint.....	31
Figuur 17: Waterval dialoog	32
Figuur 18: Chatbot Emulator credentials	32
Figuur 19: Training zinnen	33
Figuur 20: JSON object met informatie van de zin	34
Figuur 21: Voorbeeldcode communicatie via SDK	35
Figuur 22: Voorbeeld JSON object response	35
Figuur 23: Hubot met Wit.ai connectie	37
Figuur 24: De body retourneerd door Wit.ai	37
Figuur 25: Foutmelding bij het starten van Hubot nadat de instellingen correct ingesteld waren.....	38
Figuur 26: Koppeling van Wit.ai met het Microsoft Bot Framework	38
Figuur 27: Intent match	39
Figuur 28: Connectie tools	39
Figuur 29: Intent match code.....	40
Figuur 30: Intent match UI.....	40
Figuur 31: Geen intent match UI	40
Figuur 32: Het resultaat van Tokenization, met bovenstaande input	45
Figuur 33: Stop word removal	46
Figuur 34: N-Grams voorbeeld van Microsoft.....	46
Figuur 35: Voorbeeld waarbij het hoofd homoniem is.....	47
Figuur 36: Stemming voorbeeld	48
Figuur 37: Voorbeeld lemmatization.....	48
Figuur 38: Rule-based flow	49
Figuur 39: Proces van machine learning approach	49
Figuur 40: Dialogflow zin.....	50
Figuur 41: Dialogflow synoniemen	51
Figuur 42: Dialogflow resultaat man	51
Figuur 43: Buik verkeerd gespeld.....	52
Figuur 44: Training tab met de verkeerd gespelde zin.....	52
Figuur 45: intent toevoegen	53
Figuur 46: Entiteit koppelen aan het woord	53

Figuur 47: Approven van een zin	53
Figuur 48: Toegevoegde trainingsdata zin	54
Figuur 49: Entiteit zonder biuk.....	54
Figuur 50: Biuk toegevoegd	54
Figuur 51: Typfout met juiste intent.....	55
Figuur 52: Intent based conversatie. Overgenomen uit Twitter van (Zadeh, 2016).....	56
Figuur 53: Voorbeeld van een decision tree	57
Figuur 54: Directed graph	58
Figuur 55: Undirected graph	58
Figuur 56: Een voorbeeld van een graph	59
Figuur 57: Graph flowchart.....	60
Figuur 58: Eenvoudig dialoog in JSON formaat	61
Figuur 59: Dialog stack	62
Figuur 60: Dialog initialisatie	62
Figuur 61: initialDialog implementatie.....	62
Figuur 62: Code vergelijking met de chart.....	63
Figuur 63: Actions om de losse dialogen te starten.....	63
Figuur 64: Eerste bericht in de GlobalQuestionsDialog	63
Figuur 65: Choice Prompt	64
Figuur 66: Time Prompt	64
Figuur 67: Confirm Prompt.....	64
Figuur 68: Beslissing in de GlobalQuestionsDialog	64
Figuur 69: Bot operatie 'getConclusion'	65
Figuur 70: Gebruik van 'getConclusion'.....	65
Figuur 71: Kenmerken van soorten hoofdpijn in JSON.....	66
Figuur 72: Entity voedsel.....	67
Figuur 73: Zin output.....	68
Figuur 74: Huidige opzet BB	68
Figuur 75: Alternatief waarbij KLM geen controle had	69
Figuur 76: Opbouw van een chatbot	70

2.2. Tabellen

Tabel 1: Afkortingenlijst.....	8
Tabel 2: Chatbot applicatie tools	23
Tabel 3: NLP tools	24
Tabel 4: Voor- en nadelen van frameworks.....	41
Tabel 5: Voor- en nadelen van de NLP tools.....	42
Tabel 6: Voor- en nadelen van rule-based approach.....	49
Tabel 7: Voor- en nadelen van machine learning approach voor het maken van NLP	49

3. Inleiding

Voor het NotS semester wordt een onderzoek uitgevoerd die gericht is op CUI's. Het doel van dit onderzoek is om de mogelijkheden te bepalen waarmee een CUI de ernst van een klacht kan classificeren en daarbij de huisarts kan assisteren.

Binnen dit onderzoek zal gefocust worden op chatbots die tekstverwerken. Hiervoor is gekozen omdat het spraak omzetten naar tekst een stap is die geen invloed zal hebben op de conclusie van het onderzoek. Daarnaast moet CUI ingeperkt worden omdat dit onderwerp te breed is om goed te onderzoeken binnen de beschikbare tijd.

De resultaten die in dit onderzoek worden gevonden kunnen gebruikt worden bij het realiseren van het prototype.

De conclusie van elk hoofdstuk zal praktisch aangetoond worden met een proof of concept mits dit haalbaar is binnen de tijd. Na het onderzoek zal een prototype gerealiseerd worden die de samenwerking tussen alle proof of concepts demonstreren.

Het maatschappelijke voordeel van het prototype is dat de CUI kan ondersteunen bij de vraag van de patiënt, daarbij een afspraak bij de huisarts kan adviseren en/of maken. De CUI kan de patiënt geruststellen wanneer de klacht als licht wordt geëvalueerd en een afspraak mogelijk overbodig is.

De overheid wil dat patiënten minder snel het ziekenhuis bezoeken, maar eerder contact opnemen met de huisarts om de medische kosten te drukken. Het resultaat hiervan is dat de huisartsen kampen met overvolle wachtkamers. 29 procent van de patiënten blijkt uit onderzoek overbodig een afspraak te hebben bij de huisarts. Het resultaat hiervan is dat ongeveer tachtig procent huisartsen symptomen vertonen van overspannenheid. (AVRO Tros, 2017) (Buskoop, 2017) (Maes, 2017)

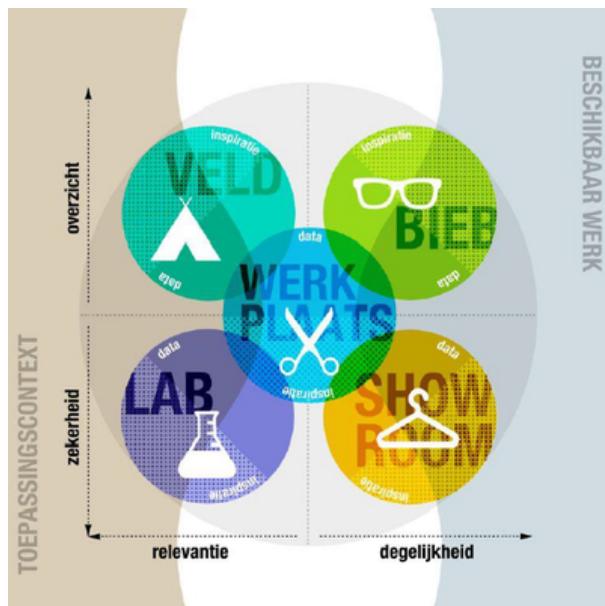
Met dit onderzoek wordt geprobeerd voortgang te boeken bij het verhelpen van dit maatschappelijke probleem.

4. Onderzoeks methode

Hier worden de methoden beschreven die toegepast zijn tijdens het onderzoek.

4.1. Methoden

Binnen dit onderzoek worden verschillende onderzoeks methoden gebruikt. Deze methoden komen uit de HBO-i methoden toolkit (HBO-i, 2018). Per deelvraag wordt benoemd welke specifieke methoden gebruikt worden. De methoden Veld, Bieb, Werkplaats, Lab, en Showroom worden gebruikt in dit onderzoek. (HBO-i, 2018)



Figuur 1: HBO-i methode toolkit

Veld

Bij de Veld methode worden de belangrijke aspecten van de leefomgeving van de gebruiker goed bestudeerd.

Bieb

De Bieb methode bouwt op informatie gevonden in literatuur.

Werkplaats

Met de Werkplaats methode wordt meer informatie gevonden door toe te passen.

Lab

De Lab methode verkrijgt informatie door te meten.

Showroom

De Showroom methode vergelijkt zorgvuldig, verantwoordt en benoemt verschillen.

4.2. Soorten onderzoek

Ook roepen deelvragen verschillende soorten onderzoek op. Deze soorten worden gedefinieerd door middel van de vraagstelling (Swaen, 2012). De soorten beschrijvend, adviserend, explorerend of vergelijkend onderzoek worden gebruikt. (Swaen, 2012)

Beschrijvend onderzoek

Beschrijvend onderzoek geeft meer informatie over een specifiek onderwerp

Adviserend onderzoek

Een adviserend onderzoek geeft een advies op basis van gevonden informatie en afwegingen.

Explorerend onderzoek

Een explorerend onderzoek gaat in op de oorzaken.

Vergelijkend onderzoek

Een vergelijkend onderzoek weegt verschillende mogelijkheden af.

5. Uit welke onderdelen is een chatbot opgebouwd?

In de eerste deelvraag wordt onderzocht uit welke onderdelen een chatbot bestaat, of deze extern beschikbaar zijn en waar ze voor dienen. Voor deze vraag beantwoord gaan worden, wordt eerst een korte introductie gegeven over wat chatbots kunnen.

5.1. Introductie

Een chatbot is een online tool om automatisch gesprekken te kunnen voeren met fysieke gebruikers, zonder dat de chatbot wordt ondersteund door een persoon. Net zoals bij menselijke gesprekken vindt een dialoog plaats. (Schlicht M. , The complete beginner's guide to chatbots, 2016) (Ina, 2017)

Chatbots zijn te verdelen in twee categorieën:

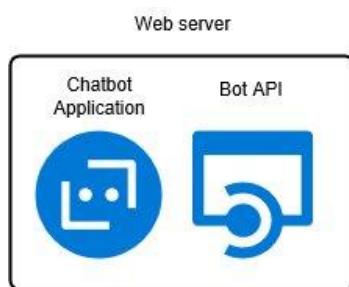
- Chatbot met een waterval pad, deze chatbot heeft de antwoorden en vragen al vooraf vastgesteld en kan daarom niet afwijken van zijn patroon.
- Chatbot met een relatief pad, deze chatbot kan omgaan met afwijkende antwoorden en vragen. Het pad is relatief en daarom staan de antwoorden en vragen niet al vastgesteld.

Chatbots hebben tegenwoordig verschillende doeleinden, bijvoorbeeld:

- Vragen beantwoorden binnen een webshop (Bol.com)
- Vluchtgegevens opvragen (KLM)

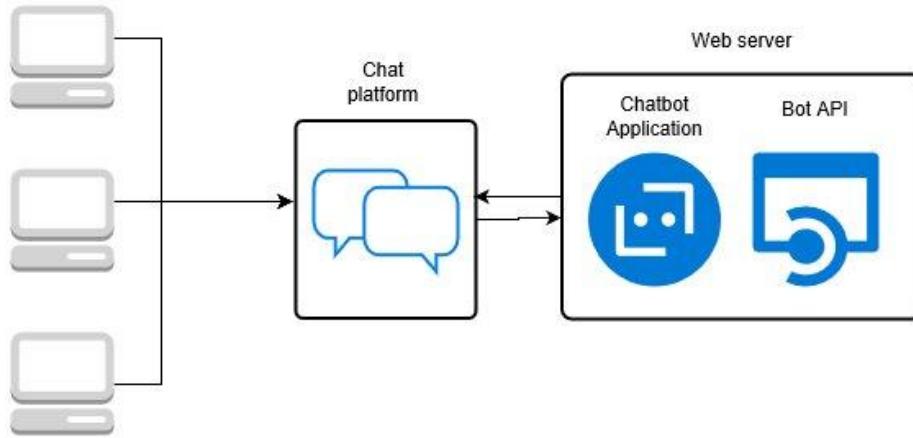
5.2. Network architecture overview

In deze paragraaf wordt de opbouw van een algemene chatbot uitgelegd. Deze afbeeldingen bevatten ook optionele componenten, bij elke afbeelding zal worden vermeld of dit het geval is. Een chatbot applicatie is software die draait op een server. De chatbot applicatie moet toegankelijk zijn vanuit de buitenwereld zodat deze de berichten kan ontvangen. Hier komt een API (Zie *Figuur 2*) in actie, deze API draait naast de chatbot applicatie op dezelfde server. (Botpress, 2017)



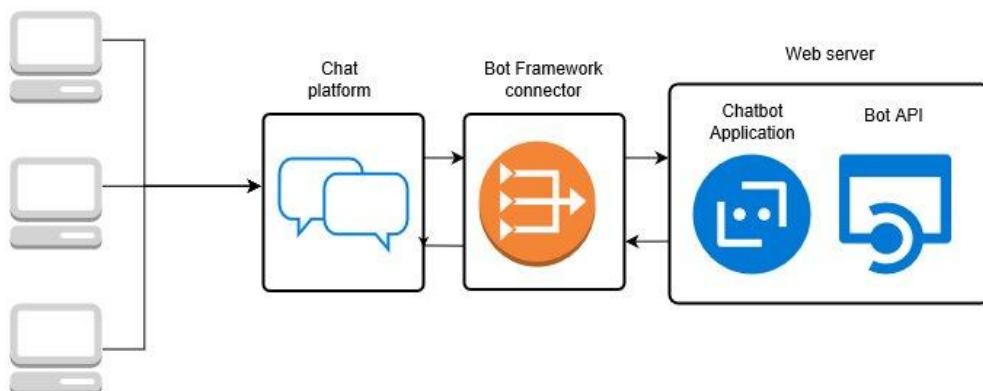
Figuur 2: Essentiële onderdelen van een chatbot

Deze implementatie van een chatbot is toegankelijk vanaf de buitenwereld, maar alleen voor technisch onderlegde gebruikers. Om het toegankelijk te maken voor alle gebruikers, is een chat platform (Zie Figuur 3) nodig. Dit platform zal de berichten van de gebruiker ontvangen en omzetten naar een standaardformaat die door de API uitgelezen kan worden. (Maruti Techlabs, Z.D.) (Kompella, 2018)



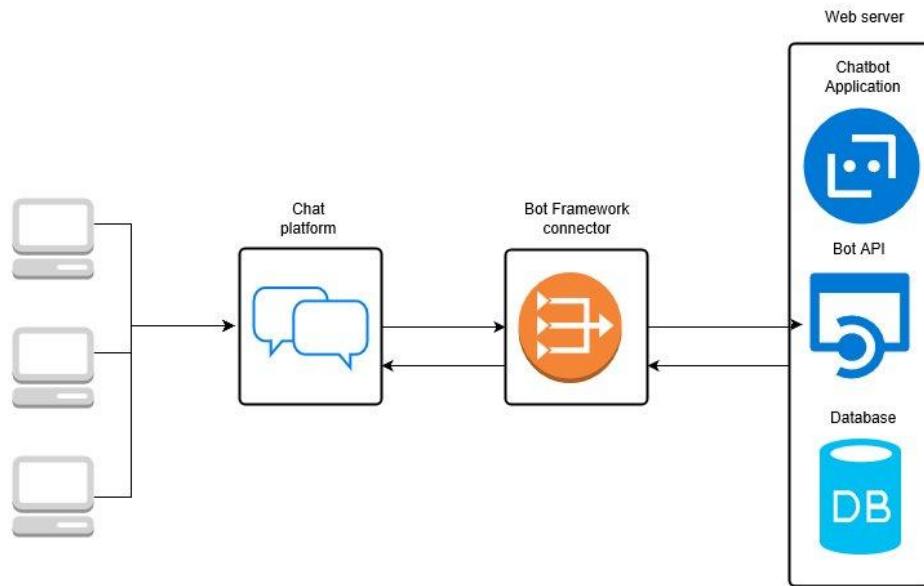
Figuur 3: Een chatbot bruikbaar voor gebruikers

Om een chatbot menselijk over te laten komen zijn verschillende aspecten nodig. Daarom bieden enkele instanties een framework/SDK aan, die toegankelijk zijn voor ontwikkelaars. Deze frameworks geven ontwikkelaars de mogelijkheid om gebruik te maken van grote chat platformen zoals bijvoorbeeld: Skype, Slack en Facebook Messenger. Een aantal van deze frameworks komen met een bot framework connector (Zie Figuur 4). Dit is software dat berichten vanuit het platform gelijk maakt en doorstuurt naar de API.



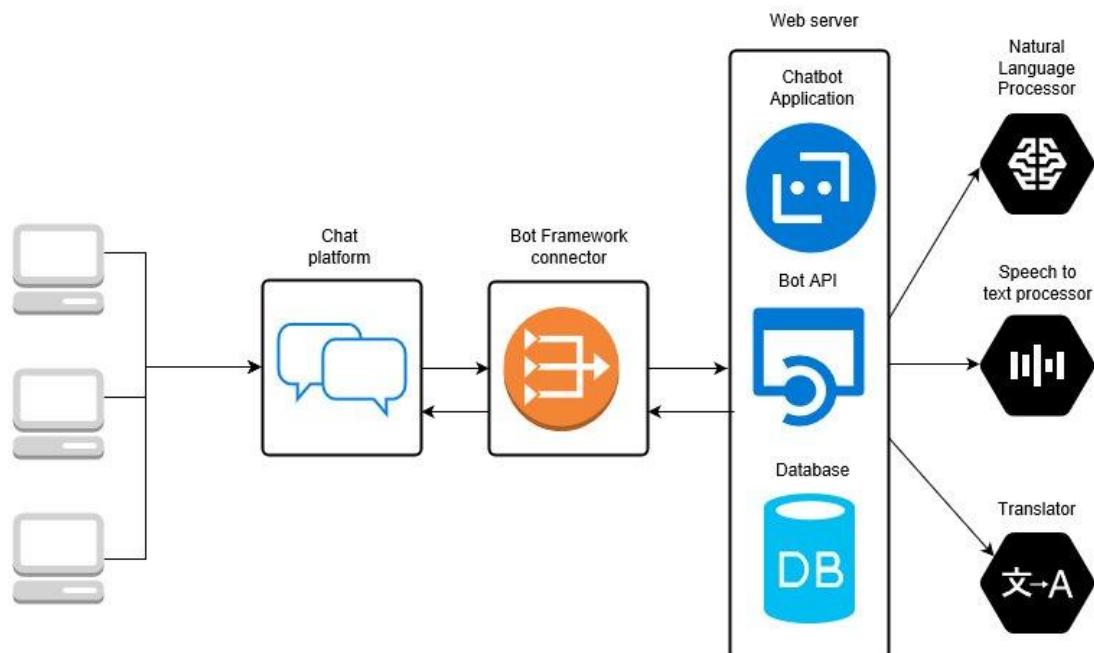
Figuur 4: Een potentieel multi-platform chatbot

Een gebruiker heeft de mogelijkheid om een gesprek te starten met een chatbot. Door de voorgaande berichten op te slaan kan de chatbot belangrijke onderwerpen van het gesprek onthouden. Dit zal de chatbot menselijker maken. Om deze informatie op te kunnen slaan is het mogelijk om een database te gebruiken (Zie Figuur 5).



Figuur 5: Een chatbot met opslag

Met de opstelling zoals zichtbaar in Figuur 6 kan een chatbot menselijker functioneren. Met deze componenten is het alleen mogelijk om specifiek geformuleerde berichten te herkennen. Om een chatbot intelligenter te maken, kan gebruik worden gemaakt van cognitieve services, deze services zijn optioneel. Een voorbeeld van een cognitive service is een translator, die een woord of zin ontvangt en de vertaling ervan terugstuurt.



Figuur 6: Een intelligente chatbot

5.3. Webserver

In deze paragraaf wordt de webserver van een chatbot applicatie beschreven. De webserver is opgebouwd uit verschillende componenten. Deze worden hieronder benoemd.

5.3.1. API

De API binnen de webserver zorgt voor de communicatie tussen de buitenwereld en de chatbot applicatie. De API van de chatbot applicatie zorgt ervoor dat de chatbot applicatie berichten kan ontvangen. (Tuil, Z.D.)

5.3.2. Chatbot applicatie

Hier worden de berichten gelezen, antwoorden bepaald en de conversatie gevoerd. (Schlicht M. , 2016)

5.3.3. Database

De database binnen de webserver is optioneel en kan worden gebruikt voor verschillende doeleinden. Zo kunnen de vragen en antwoorden worden opgeslagen, op deze manier kan de chatbot leren van vorige vragen en antwoorden.

5.4. Chatbot

In deze paragraaf worden verschillende component beschreven die van toepassing zijn voor de chatbot applicatie.

5.4.1. Platformen

Bedrijven kiezen ervoor om de mogelijkheden van een chatbot in eigen beheer te houden. Om dit te kunnen beheren is het vereist om de chatbot via een eigen platform aan te bieden. Om de chatbot in eigen beheer te houden loopt de communicatie via de backend van het bedrijf.

5.4.2. Framework connector

Om een chatbot met meerdere chat platforms te laten communiceren moet deze op een vooraf afgesproken manier berichten versturen. De connector zet de invoer van de API om naar het platform specifieke formaat en vice versa.

5.4.3. Cognitieve diensten

Cognitieve diensten zijn externe software, toegankelijk via een API. Deze diensten verwerken de gegeven data en sturen het resultaat terug. Hier een aantal voorbeelden van een cognitieve dienst. (Microsoft, Z.D.)

5.4.4. Natural language processing

Met NLP wordt een zin vertaald naar een intentie met daarbij beschrijvende steekwoorden. Om zinnen goed te kunnen laten vertalen door een Natural Language Processor moet deze getraind worden. (Liddy, NAtural Language Processing, 2001) (Chowdhury, Natural language processing, 2005)

5.4.5. Image recognition

Image recognition is een dienst die patronen kan herkennen in zowel afbeeldingen/foto's als videobeelden. De software die zorgt voor het herkennen van beelden, geeft meerdere objecten terug met een percentage. Dit percentage geeft aan hoe groot de kans is dat het object voorkomt in de afbeelding (TensorFlow, 2018).

5.4.6. Speech recognition

Speech recognition omvat het vertalen van gesproken taal naar geschreven taal. Hierbij zal de software de geluidsgolven herkennen. (Google, 2018)

5.4.7. Websearch

Met websearch kan software direct een betekenis vinden bij opgegeven woorden. Deze betekenis kan gebruikt worden om later gerichtere antwoorden te geven. (Microsoft, z.d)

5.4.8. Translation

Een vertaler kan een zin of woord vertalen. Omdat de gesproken taal kan verschillen met geschreven taal, moet de vertaler blijven leren. (Microsoft, Z.D)

6. Welke onderdelen zijn relevant voor dit onderzoek?

In deze deelvraag wordt gekeken naar de relevantie van de onderdelen die gevonden zijn in de eerste deelvraag. Hierbij zal ook toegelicht worden waarom deze relevant zijn. Vervolgens zullen de relevante onderdelen uitgelegd worden.

6.1. De relevante onderdelen

Uit de eerste deelvraag zijn de volgende onderdelen gekomen. Het platform, de framework connector, de cognitieve diensten en de webserver met daarbinnen de chatbot applicatie, de API en de database. Deze onderdelen zijn stuk voor stuk belangrijk om een chatbot te realiseren. Elk onderdeel zal hieronder worden beschreven en zal worden aangegeven of het relevant is voor het onderzoek.

6.1.1. Platform

Het platform is de applicatie waarmee een gebruiker converseert met de chatbot. Het platform is over het algemeen een reeds gerealiseerd onderdeel. Ook zijn deze platforms niet specifiek gemaakt om te communiceren met chatbots. Omdat het platform geen invloed heeft op de werking van de chatbots is dit onderdeel niet relevant om verder te onderzoeken.

6.1.2. Framework connector

De framework connector is het deel waarmee het platform met de webserver verbonden wordt via de API. Dit onderdeel is optioneel omdat het platform ook direct kan communiceren met de API. Het gebruik van een framework connector is om platform specifieke HTTP berichten te vertalen naar HTTP berichten specifiek voor de chatbot. Met deze vertaling is het mogelijk om meerdere platforms met de chatbot te laten communiceren. Een framework connector is overbodig wanneer de chatbot zich alleen richt op maar één platform. Dit onderdeel is daarom niet relevant om verder te onderzoeken.

6.1.3. Webserver – Chatbot applicatie

De chatbot applicatie is waar de conversatie in stand gehouden wordt. Hier wordt het bericht van de gebruiker ontvangen en stuurt de chatbot een relevante vraag terug. Dit is het brein van de chatbot. Omdat de chatbot applicatie essentieel is in het maken van een chatbot is dit onderdeel relevant om verder te onderzoeken.

6.1.4. Webserver – API

De API maakt de chatbot bereikbaar van buiten de webserver. Ook zal deze berichten naar buiten kunnen versturen. Het dient binnen de webserver als schakel tussen de interne onderdelen. Een API is een stukje software wat niet chatbot specifiek is, dit onderdeel is daarom ook niet relevant om verder te onderzoeken. (Gazarov, 2016)

6.1.5. Webserver – Database

De database is waar alle informatie van de gesprekken opgeslagen wordt. Deze informatie is belangrijk daarom is het belangrijk een database te gebruiken bij het realiseren van een chatbot. Echter is dit onderdeel niet specifiek gemaakt om data van chatbots op te slaan, daarom is dit onderdeel niet relevant om verder te onderzoeken. (Rouse, 2017)

6.1.6. Cognitieve diensten

De cognitieve diensten zijn additionele externe applicaties waar een chatbot gebruik van kan maken. In de eerste deelvraag zijn een aantal voorbeelden genoemd zoals spraakherkenning of NLP. Omdat de chatbot in dit onderzoek zinnen moet gaan ontleden is van de cognitieve diensten alleen NLP relevant om verder te onderzoeken.

6.2. De werking van de relevante onderdelen

6.2.1. Chatbot applicaties

Wanneer een chatbot applicatie een zin binnen krijgt kan de applicatie hier iets mee doen. De zin kan met een NLP ontleed worden, of de zin kan verder verwerkt worden. De chatbot gaat vervolgens met de gevonden informatie een antwoord kiezen of maken.

Het is mogelijk om als chatbot op basis van een bijvoorbeeld een specifiek steekwoord een antwoord terug te geven. Dit antwoord kan dan een bestaande zin zijn, maar met een cognitieve service kan op basis van bijvoorbeeld dit steekwoord ook een antwoord gegenereerd worden. De chatbot applicatie zorgt dus voor het ontvangen van vragen en hier informatie uithalen om vervolgens een antwoord te maken of kiezen en terug te sturen. (Schlicht M. , The Complete Beginner's Guide To Chatbots, 2016)

6.2.2. Natural Language Processing

Wanneer een chatbot een bericht ontvangt van een gebruiker, weet de bot niet wat hier mee moet gebeuren wanneer het op een andere manier geformuleerd wordt dan de bot kent. Hier komt een Natural Language Processor in actie. Dit is een manier om een menselijke zin te ontleden en daaruit te kunnen begrijpen wat hiermee bedoeld wordt.

Natural Language Processor ontwikkelaars verzamelen informatie over hoe de mens de taal begrijpt en gebruikt om iets duidelijk te maken aan de ander. Met deze informatie worden computers getraind. Na deze training is het systeem in staat om de bedoeling van een zin, ongeacht hoe deze geformuleerd is, te achterhalen. De meeste Natural Language Processors zullen een zin categoriseren onder een intentie; wat wil de gebruiker bereiken met de zin? Daarbij wordt de zin ontleed in entiteiten; welke steekwoorden worden herkend. Deze woorden zullen de specificaties duidelijk maken, zoals tijden, kleuren, plaatsen en nog veel meer.

NLP maakt gebruik van het machine learning spectrum. Met iedere nieuwe intentie en entiteiten van een zin leert de Natural Language Processor om nog beter te

ondervinden wat nou de bedoeling is van de zin. (Liddy, Natural Language Processing, 2001) (Chowdhury, Language and Representation, 2005)

6.3. Conclusie

De onderdelen die relevant zijn voor dit onderzoek zijn de chatbot applicatie en NLP. De chatbot applicatie ontvangt berichten van de gebruiker, verwerkt de zin en stuurt op basis hiervan een reactie terug. Met NLP wordt een zin ontleedt om daar de intentie en de entiteiten uit te halen.

7. Welke tools zijn beschikbaar om de onderdelen te realiseren?

Uit hoofdstuk 5 en 6 is duidelijk geworden welke componenten nodig zijn en welke componenten gebruikt gaan worden, de tools om deze componenten te implementeren worden in dit hoofdstuk onderzocht. Een lijst met gevonden tools zal gefilterd worden aan de hand van enkele criteria. De tools die overblijven worden zowel in theorie als praktijk beschreven.

7.1. Criteria

Om tot tools te komen die voldoende kwaliteit hebben, zijn criteria opgesteld. Aan deze criteria moeten de tools voldoen om verder onderzocht te worden.

7.2. Chatbot applicatie tools

De criteria die zijn opgesteld voor de tools worden hieronder beschreven. Wanneer een tool niet voldoet aan één van de gestelde criteria, dan wordt deze uitgesloten. De criteria worden visueel uitgebeeld in Tabel 2.

1. De tool moet kosteloos in gebruik genomen kunnen worden.
 - Dit project heeft geen budget beschikbaar, daarom is het niet mogelijk om betaalde tools te gebruiken.
2. De tool werkt met JavaScript, C#, Java of PHP.
 - Het projectteam is bekend met de benoemde programmeertalen en het is niet realistisch om binnen het project een nieuwe taal te leren.
3. De tool is maximaal een half jaar geleden geüpdateert.
 - Het onderzoek betreft de nieuwe tools en technologieën voor een chatbot, het is daarom essentieel dat deze nog actief bijgewerkt worden.
4. De tool kan uitgebreid worden met externe tools.
 - Wanneer een dienst stopt moet deze vervangen kunnen worden zonder alles opnieuw te programmeren.
 - Door modulariteit is het mogelijk om ongebruikte tools niet te integreren in de chatbot.
 - Om in de toekomst nieuwe of vernieuwde tools te kunnen gebruiken moet de tool uitbreidbaar zijn.
5. De tool heeft een community die in de afgelopen maand actief is geweest op één van de beschikbaar gestelde hulpmiddelen.
 - Met behulp van een actieve community kan essentiële informatie ingewonnen worden.
 - Beschikbare hulpmiddelen zijn minimaal: Stack Overflow of GitHub
6. De tool is voorzien van een quickstart.
 - Om snel van start te kunnen gaan is het essentieel dat een start documentatie aanwezig is.

Naam	Criteria 1	Criteria 2	Criteria 3	Criteria 4	Criteria 5	Criteria 6
Aspect CXP (Aspect, Z.D.)	Red	-	-	-	-	-
Microsoft Bot Framework (Microsoft, Z.D.)	Green	JavaScript, C#	13-12-2017	Green	Green	Green
Botpress (Botpress, Z.D.)	Green	JavaScript	9-3-2018	Green	Red	-
Smooch (Smooch Technologies, Z.D.)	Red	-	-	-	-	-
Botkit (Botkit, Z.D.)	Red	-	-	-	-	-
Bottr (James Campbell, Z.D.)	Green	JavaScript	27-11-2016	-	-	-
Hubot (Github, Z.D.)		JavaScript	16-2-2018	Green	Green	Green
Nestor (Zerobotlabs, 2016)	Green	JavaScript	15-02-2016	-	-	-
Botmaster AI (Botmaster, Z.D.)	Green	JavaScript	27-01-2018	Green	Red	-
Flow xo (Flow Xo, Z.D.)	Red	-	-	-	-	-
Chatterbot (Gunthercox, 2018)	Green	Python	-	-	-	-

Tabel 2: Chatbot applicatie tools

7.3. NLP tools

De criteria die zijn opgesteld voor de NLP tools worden hieronder beschreven. Wanneer een NLP tool niet voldoet aan één van de gestelde criteria, dan wordt deze uitgesloten van gebruik. De criteria worden visueel uitgebeeld in Tabel 3.

- a) De NLP tool moet kosteloos in gebruik genomen kunnen worden.
 - Dit project heeft geen budget beschikbaar, daarom is het niet mogelijk om betaalde tools te gebruiken.
- b) De NLP tool beschikt over een reeds bestaande API.
 - Om geen tijd te besteden aan het schrijven van een eigen API voor de NLP tool kan de relevante stof dieper onderzocht worden
- c) De NLP tool ondersteunt de Nederlandse taal.
 - De chatbot moet geschikt zijn voor Nederlandse huisartsen.

Naam	Criteria A	Criteria B	Criteria C
Wit.ai (Facebook, Z.D.)			
LUIS (Microsoft, Z.D.)		-	-
FROG (Radboud University, Z.D.)			-
TextRazor (TextRazor Ltd., Z.D.)		-	-
opeNER (OpeNER, Z.D.)			-
spaCy (Eplosion AI, Z.D.)			-
OpenNLP (The Apache Software Foundation, Z.D.)			-
Dialogflow (Google, Z.D.)			

Tabel 3: NLP tools

7.4. Onderzoek tools

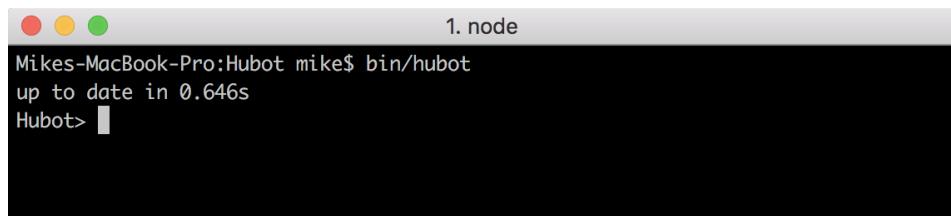
In deze paragraaf worden de tools die aan de criteria voldoen in theorie en in de praktijk beschreven. In de kopteksten staat ook aangegeven voor welk onderdeel van een chatbot deze tool is. Ook wordt per onderdeel een conclusie gegeven en een keuze gemaakt welke tool gebruikt gaat worden in het prototype.

7.5. Chatbot applicatie tool - Hubot

In deze paragraaf zal eerst de basis van Hubot behandelen. Vervolgens zal uitgelegd worden hoe Hubot werkt, hierbij zal een voorbeeld gegeven worden. Aan het eind van de paragraaf wordt de conclusie geschreven.

Hubot is een bot framework geschreven door het team van GitHub (Github Inc., Z.D.). Het is bedoeld om chats van bedrijven gedeeltelijk te automatiseren. Dit kan bijvoorbeeld door Hubot teksten te laten vertalen, afbeeldingen op te laten zoeken of om een landkaart te geven. Ook is het mogelijk om eigen functionaliteiten te schrijven.

Hubot is geschreven in het JavaScript framework CoffeeScript (Jashkenas, Z.D.). Na de installatie te hebben gevolgd die te vinden is op de GitHub van Hubot (Github, Z.D.) (Github, Z.D.) kan de chatbot gestart worden via de terminal (Zie Figuur 7). Dit is de standaard om te beginnen met Hubot, het is ook mogelijk om een platform zoals Slack te gebruiken in plaats van de terminal.



```
1. node
Mikes-MacBook-Pro:Hubot mike$ bin/hubot
up to date in 0.646s
Hubot>
```

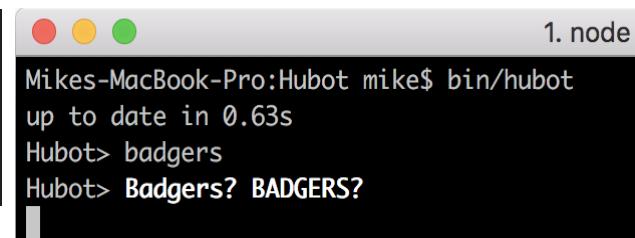
Figuur 7: Het starten van Hubot

Wanneer een bericht gestuurd wordt naar Hubot, geeft hij nu geen reactie. In de map "script" kunnen scripts geschreven worden die reageren op de verstuurde berichten. Deze scripts zijn opgebouwd uit standaardzinnen of woorden waar Hubot vervolgens op zal reageren. De code hierachter is te zien op Figuur 8. In Figuur 9 is de reactie in de terminal te zien



```
10
11  module.exports = (robot) =>
12
13    |   robot.hear /badger/i, (res) =>
14    |     res.send "Badgers? BADGERS?"
15
```

Figuur 9: Hubot reactie



```
1. node
Mikes-MacBook-Pro:Hubot mike$ bin/hubot
up to date in 0.63s
Hubot> badgers
Hubot> Badgers? BADGERS?
```

Figuur 8: Hubot code

Naast de robot.hear functie heeft Hubot nog meer functies, veel hiervan zijn specifiek voor wanneer Hubot gebruik maakt van Slack. Wel kan robot.respond gebruikt worden. Hierbij reageert Hubot alleen wanneer zijn naam wordt genoemd en de exacte zin opgegeven is in de code. Bij het bovenstaande voorbeeld, figuur 8 en 9, kan bijvoorbeeld ook een hele andere zin staan met het woord “badger” erin, hier zal Hubot dan nog steeds op reageren.

Hubot zelf is geen slimme bot. Hij biedt geen mogelijkheid om tekst te begrijpen of keuzes te maken. Wel heeft Hubot met de robot.http functie de mogelijkheid om HTTP requests te sturen en ontvangen. Dit biedt de mogelijkheid om met een andere tool te communiceren en Hubot een “slimme” bot te maken.

7.5.1. Conclusie

Hubot is geen intelligente bot. Dankzij de mogelijkheid om te kunnen communiceren met externe tools is het wel een geschikte tool om de fungeren als Framework Connector, zie het eerste hoofdstuk. Omdat Hubot met http requests verbinding kan maken met externe tools zal deze tool meegenomen worden in verder onderzoek.

7.6. Chatbot applicatie tool – Microsoft Bot Framework

Het MSBF voldoet aan de gestelde criteria, daarom kan het MSBF worden onderzocht. In deze paragraaf zal eerst de basis van het MSBF behandelen. Vervolgens zal uitgelegd worden hoe het MSBF werkt per programmeertaal, hierbij worden voorbeelden gegeven. Als laatste wordt de conclusie gegeven over het MSBF.

7.6.1. Wat is het Microsoft Bot Framework?

Het MSBF is ontwikkeld door Microsoft en wordt gebruikt om het ontwikkelen van een bot te vergemakkelijken door meerdere tools en services aan te bieden. Deze onderdelen zullen in deze paragraaf worden uitgelicht. (Microsoft, Z.D.)

7.6.2. Soorten chatbots

Microsoft heeft de chatbots verdeeld in drie categorieën. Deze worden hieronder beschreven. (Microsoft, 2017)

7.6.3. Commerce chatbot

De commerce chatbot kan een vervanging worden van een email en telefoon binnen een bedrijf waar vragen aangesteld worden. Een voorbeeld hiervan is roomservice bij een hotel. Om een commerce chatbot te realiseren zijn de volgende externe tools nodig:

- Azure AD for Authentication
- Cognitive services: LUIS
- Application Insights

7.6.4. Informatie chatbot

De informatie chatbot kan vragen beantwoorden die veel gesteld worden. Om een informatie chatbot te realiseren zijn de volgende externe tools nodig:

- Azure AD for Authentication
- Cognitive services: QnA Maker
- Azure Search
- Application Insights

7.6.5. Enterprise chatbot

De enterprise chatbot kan de productiviteit van gebruikers verbeteren door een chatbot te implementeren in bijvoorbeeld de Office 365 agenda. Door eenvoudige vragen te stellen aan de chatbot krijgt de gebruiker de exacte informatie die nodig is. Een vraag aan de enterprise chatbot kan zijn “Wanneer is mijn volgende afspraak?”. Om een enterprise chatbot te realiseren zijn de volgende externe tools nodig:

- Azure AD for Authentication
- Cognitive services: QnA Maker
- Azure Search
- Application Insights

7.6.6. Microsoft Bot Framework onderdelen

Het MSBF is verdeeld in drie hoofdcomponenten. Deze drie componenten worden hieronder beschreven.

- Microsoft Bot Builder
- Microsoft Cognitieve services
- Microsoft Azure bot framework

7.6.6.1. Microsoft Bot Builder

De Microsoft Bot Builder is één van de drie hoofdcomponenten van het MSBF. Microsoft Bot Builder bevat de volgende componenten:

- SDK
- Libraries
- Voorbeelden
- Debug tools

Met de SDK is het mogelijk om een chatbot vanaf de grond op te bouwen. De SDK ondersteund .NET en Node.js. Het is ook mogelijk om de API te gebruiken de programmeertaal is dan onafhankelijk. (Microsoft, 2017)

Met behulp van de emulator is het mogelijk om de chatbot te debuggen. Met de channel inspector is het mogelijk om de chatbot op verschillende platforms te testen om bijvoorbeeld de werking van de chatbot te controleren. (Microsoft, 2018)

7.6.6.2. Microsoft Cognitieve Services

Zie deelvraag 2 Cognitieve diensten voor een beschrijving over cognitieve services.

De Microsoft cognitive services zijn externe tools die de mogelijkheid bieden om de chatbot intelligenter te maken. De Microsoft cognitieve services van het MSBF zijn onder te verdelen in de volgende categorieën (Microsoft, Z.D.):

- Zicht
- Spraak
- Taal
- Intelligentie

- Zoeken

7.6.6.3. Microsoft Azure Bot Service

De Azure Bot Service is een grootschalige applicatie die het maken van een bot makkelijker maakt door middel van een gebruiksvriendelijke interface. De service biedt de mogelijkheid om een bot gemaakt met de Microsoft Bot Builder te deployen op Microsoft servers en openbaar te maken voor het internet. De service biedt dus de mogelijkheid om te dienen als connector. Met Azure Bot Service kan een bot worden gebouwd zonder kennis van programmeren te hebben. Dit kan door middel van een gebruiksvriendelijke interface in samenwerking met een drag and drop systeem gebouwd worden. (Microsoft, Z.D.)

Wanneer een bot is geregistreerd via Azure Bot Service kan deze verbonden worden met verschillende platforms als GroupMe, Messenger, Kik, Skype, Slack, Telegram en nog veel meer. Ontwikkelaars hebben toegang tot kant-en-klare sjablonen ontworpen voor onder andere basic, formulier, taalbegrip, vraag en antwoord en proactieve bots. Het wordt makkelijker gemaakt om deze bot te verbinden met de cognitive services van Microsoft. Echter is het niet niet verplicht om Microsoft Azure Bot Service hiervoor te gebruiken.

Wanneer gebruik gemaakt wordt van deze service komt de gebruiker direct terecht in het dashboard waar alle informatie zichtbaar wordt van de geregistreerde bots. Dit houdt in dat inzicht wordt gemaakt in bijvoorbeeld hoeveel verkeer door de bot gaat of hoeveel berichten de bot verstuurd naar bepaalde services. (Microsoft, Z.D.)

7.6.7. Hoe werkt het Microsoft Bot Framework?

Voor het werken met het MSBF is een keuze tussen de programmeeromgeving van Microsoft met .NET en C#, of de Node.js omgeving met JavaScript. Het werken met .NET vereist dat met object georiënteerd principes gewerkt wordt, waarbij het werken met JavaScript de mogelijkheid biedt om te werken via de functional principes. Beide manieren bieden dezelfde mogelijkheden en hebben daarom naast persoonlijke voorkeuren geen voor of nadelen tegenover elkaar.

7.6.8. .NET SDK

Om te werken met het .NET framework wordt aangeraden gebruik te maken van de Visual studio IDE van Microsoft doordat hier in een .Net compiler is meegeleverd. Daarbij moet de SDK worden toegevoegd aan de Visual Studio templates.

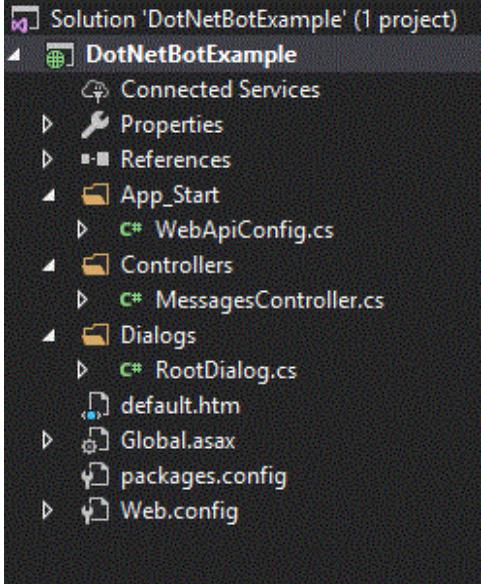
Bij het aanmaken van een project met de MSBF template, worden een aantal bestanden aangemaakt. Deze bestanden zijn cruciaal bij het gebruik van het framework.

Met de bestanden die automatisch gegenereerd worden is, kan de applicatie direct worden gebruikt. Wanneer deze applicatie wordt gebruikt zal verbinding gemaakt kunnen worden met het lokale netwerk op poort 3978. De bot zal echter niets anders doen dan vertellen tegen de gebruiker wat de bot heeft ontvangen en hoe lang het bericht is.

Er worden bij de template configuratie en klasse bestanden meegeleverd voor een API.

Deze API is nodig om voor de bot berichten te ontvangen.

De bot applicatie begint bij de *MessagesController klasse* (Zie Figuur 11). In deze klasse wordt bepaald welke dialogen initieel worden afgelopen. Het is in dit geval mogelijk om berichten van het systeem op te vangen en af te handelen, bijvoorbeeld wanneer een gebruiker het gesprek verlaat.



```
namespace DotNetBotExample
{
    [BotAuthentication]
    public class MessagesController : ApiController
    {
        /// <summary>
        /// POST: api/Messages
        /// Wanneer er een HTTP POST bericht binnen komt, wordt deze met deze methode afgevangen.
        /// </summary>
        public async Task<HttpResponseMessage> Post([FromBody]Activity activity)
        {
            // Als het bericht een type 'message' heeft...
            if (activity.Type == ActivityTypes.Message)
            {
                // Handel het bericht af in de RootDialog.
                await Conversation.SendAsync(activity, () => new Dialogs.RootDialog());
            }
            else
            {
                // Handel het bericht af in HandleSystemMessage.
                HandleSystemMessage(activity);
            }
            var response = Request.CreateResponse(HttpStatusCode.OK);
            return response;
        }

        private Activity HandleSystemMessage(Activity message)
        {
            if (message.Type == ActivityTypes.DeleteUserData)
            {
                // Hier wordt het verwijderen van een gebruiker afgehandeld.
            }
            else if (message.Type == ActivityTypes.ConversationUpdate)
            {

```

Figuur 11: MessageController klasse

Door de template wordt een voorbeeld dialoog aangemaakt, de RootDialog klasse (Zie figuur 12). Hierin wordt het bericht ontvangen en een eenvoudig bericht teruggestuurd.

```
namespace DotNetBotExample.Dialogs
{
    [Serializable]
    public class RootDialog : IDialog<object>
    {
        /// <summary>
        /// Start het dialoog.
        /// </summary>
        /// <param name="context"></param>
        /// <returns></returns>
        public Task StartAsync(IDialogContext context)
        {
            context.Wait(MessageReceivedAsync);

            return Task.CompletedTask;
        }

        /// <summary>
        /// Zelfgemaakte methode waar het binnengekomen bericht wordt afgehandeld.
        /// </summary>
        /// <param name="context"></param>
        /// <param name="result"></param>
        /// <returns></returns>
        private async Task MessageReceivedAsync(IDialogContext context, IAwaitable<object> result)
        {
            // In de activity zit alle informatie van het bericht.
            var activity = await result as Activity;

            // Berekend de lengte van het bericht om het weer terug te sturen.
            int length = (activity.Text ?? string.Empty).Length;

            // Stuurd een bericht terug naar de gebruiker.
            await context.PostAsync($"You sent {activity.Text} which was {length} characters");

            // Blieft deze methode herhalen tot de gebruiker de verbinding verbreekt.
            context.Wait(MessageReceivedAsync);
        }
    }
}
```

Figuur 12: RootDialog Klasse

7.6.9. Node.js SDK

Om een chatbot te realiseren in Node.js is het essentieel om Node.js geïnstalleerd te hebben. Naast Node.js is het aan te raden om de Microsoft Bot Emulator te installeren, dit wordt gebruikt voor het lokaal runnen van de chatbot en het debuggen van de chatbot. De chatbot kan gestart worden via de terminal zie Figuur 13.

```
ws-165-219:NodeJSMicrosoftBot jeroenpans$ nodemon app.js
[nodemon] 1.17.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: ***!
[nodemon] starting `node app.js`
restify listening to http://[::]:3978
```

Figuur 13: Het starten van de Node.js chatbot

Met Node.js moet de server opnieuw gestart worden na aanpassingen in de code. Door gebruik te maken van de Nodemon tool wordt de server na aanpassingen automatisch opnieuw gestart. Nodemon heeft geen voordeelen voor de chatbot, maar wel voor de developer. Zo hoeft de developer niet steeds na code aanpassingen de server te herstarten. De code van een chatbot in Node.js is onder te verdelen in vier onderdelen.

7.6.10. Server(Express, Restify)

Hoe de server gemaakt wordt voor de chatbot is te zien op Figuur 14. In dit voorbeeld is gekozen voor Restify, omdat in deze tool ook in de voorbeelden van Microsoft worden gebruikt. Later zal hier ExpressJs worden gebruikt door de huidige kennis van deze tool.

```
var server = restify.createServer()
server.listen(process.env.port || process.env.PORT || 3978, function () {
  console.log('%s listening to %s', server.name, server.url)
})
```

Figuur 14: Restify server setup

7.6.11. Chatconnector

In Figuur 15 wordt de verbinding gemaakt met het bot framework. Om een chatbot framework via Azure te laten werken is een Microsoftaccount nodig. Wanneer gekozen wordt voor een lokale chatbot is geen Microsoftaccount nodig.

```
var connector = new builder.ChatConnector({
  appId: process.env.MicrosoftAppId,
  appPassword: process.env.MicrosoftAppPassword
})
```

Figuur 15: MSBF chatconnector

7.6.12. Listen voor berichten

De chatbot luistert naar de API, in Figuur 16 is te zien hoe dit opgezet wordt.

```
server.post('/api/messages', connector.listen())
```

Figuur 16: Listen to endpoint

7.6.13. Response of message

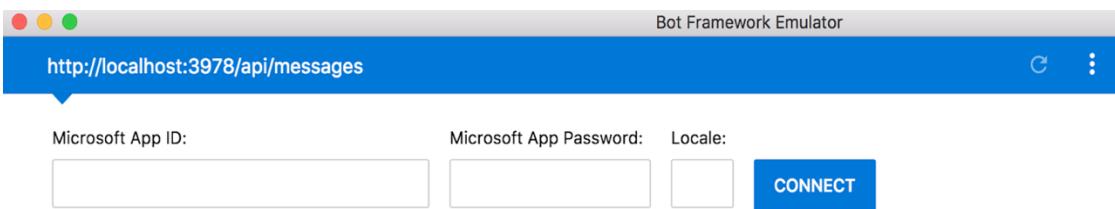
In Figuur 17 wordt de waterval chatbot gerealiseerd. De eerste vraag die gesteld wordt is “What is your name?” het is een open vraag waardoor geen specifiek antwoord is aangemaakt. Het antwoord wordt opgeslagen in de session en wordt gebruikt bij het antwoord van de chatbot. De chatbot kan dit antwoord vervolgens gebruiken in een vervolgzin door gebruik te maken van “session.result”. Bij de vraag “What language do you code Node.js using?” zijn drie vaste antwoorden opgesteld. Wanneer de gebruiker een ander antwoord geeft krijgt de gebruiker een melding dat de chatbot het antwoord niet begrijpt.

```
bot.dialog('/', [
  function (session) {
    builder.Prompts.text(session, 'What is your name?')
  },
  function (session, results) {
    session.userData.name = results.response
    builder.Prompts.number(session, 'Hi ' + results.response + ', How many years have you been coding?')
  },
  function (session, results) {
    session.userData.coding = results.response
    builder.Prompts.choice(session, 'What language do you code Node using?', ['JavaScript',
      'CoffeeScript', 'TypeScript'])
  },
  function (session, results) {
    session.userData.language = results.response.entity
    session.send('Got it... ' + session.userData.name +
      " you've been programming for " + session.userData.coding +
      ' years and use ' + session.userData.language + '.')
  }
])
```

Figuur 17: Waterval dialoog

7.6.14. Chatbot emulator

Nadat de chatbot lokaal is gestart moet een emulator gestart worden. Na het opstarten van de emulator moet verbinding worden gemaakt met de chatbot (zie Figuur 18).



Figuur 18: Chatbot Emulator credentials

De standaard endpoint van een lokale chatbot is [“http://localhost:3978/api/messages”](http://localhost:3978/api/messages). Naar deze URL luistert de chatbot voor nieuwe berichten (Microsoft, 2017). Omdat het om een lokale chatbot gaat, hoeft geen ID of wachtwoord ingevuld te worden. Deze twee velden moeten wel worden ingevuld wanneer de chatbot niet lokaal draait, maar op een externe server. De chatbot die gerealiseerd is in dit voorbeeld is een waterval methode (Microsoft, 2017). Dit houdt in dat de antwoorden al zijn opgesteld bij de vragen. Wanneer de gebruiker afwijkt van het exacte antwoord weet de chatbot niet exact wat de gebruiker bedoeld.

7.6.15. Conclusie

Het MSBF biedt de mogelijkheid om zowel met .NET als Node.js te werken. Tijdens het onderzoeken is gebleken dat beide talen dezelfde mogelijkheden bieden en daarom de keuze vooral ligt bij de persoonlijke voorkeuren in de taal en werkomgeving.

In de .NET omgeving wordt verwacht dat de developer OO te werk gaat. In de Node.js omgeving kan gebruik worden gemaakt van functioneel programmeren. Het onderzoek wordt voortgezet in de Node.js omgeving omdat deze taal en werkomgeving meer bekend is binnen het team. Ook is functioneel ontwikkelen een eenvoudigere keuze om een prototype te ontwikkelen.

7.7. NLP tool – Dialogflow

In deze paragraaf wordt uitgelegd wat Dialogflow is. Als eerst zal een korte uitleg gegeven worden over wat Dialogflow is. Vervolgens zal dieper ingegaan worden op de werking van Dialogflow. Hierbij worden voorbeelden gegeven aan de hand van afbeeldingen. Als laatste wordt de conclusie gegeven over Dialogflow.

7.7.1. Wat is Dialogflow?

Dialogflow is een NLP van Google. Het biedt de mogelijkheid voor NLP met machine learning. Dit gebeurt door te communiceren met een platform dat verbonden is aan Dialogflow. Op deze manier worden de zinnen opgeslagen die verstuurd worden vanuit het platform. Vervolgens kan de beheerder van de applicatie kijken of de juiste context uit de zinnen gehaald is en wordt op deze manier de applicatie getraind.

7.7.2. Hoe werkt Dialogflow?

Dialogflow heeft een grafische interface waarmee deze NLP tool getraind kan worden. Dit kan door intenties toe te voegen met verschillende onderdelen. Een intentie is het doel van een zin. Om tot het doel van een zin te komen worden verschillende training zinnen aangemaakt. Daarnaast kan een gebeurtenis plaatsvinden op basis van de intentie. Ook heeft een intentie context en een set antwoorden die passen bij deze intentie. De training zinnen hebben verschillende entiteiten, net als Wit.ai. De entiteiten kunnen ook zelf aangemaakt worden met synoniemen.

PARAMETER NAME	ENTITY	RESOLVED VALUE
Pijnpunt	@Pijnpunt	been

Figuur 19: Training zinnen

In Figuur 19 is te zien dat een aantal zinnen zijn toegevoegd om mee te trainen. De entiteiten van de zinnen kunnen geel gemaarkeerd worden. Met deze informatie kan

Dialogflow vervolgens leren. Wanneer een zin wordt gestuurd met een gelijke opbouw, maar een onbekende entiteit, ziet Dialogflow toch dat het in dit geval een "Pijnpunt" entiteit is.

Zodra een zin binnenkomt bij Dialogflow wordt deze ontleed tot een JSON object. In dit object staat alle informatie over deze zin, welke intentie deze heeft, welke entiteit(en) etcetera. Hieronder is een voorbeeld gegeven.

```
1  {
2      "id": "3f9a9240-a726-4d62-8340-e1cb417a8c82",
3      "timestamp": "2018-04-03T12:46:55.105Z",
4      "lang": "nl",
5      "result": {
6          "source": "agent",
7          "resolvedQuery": "Ik heb hoofdpijn",
8          "action": "",
9          "actionIncomplete": false,
10         "parameters": {
11             "Pijnpunt": "hoofdpijn"
12         },
13         "contexts": [],
14         "metadata": {
15             "intentId": "a291a1fc-41da-4228-93d4-1981113fdcd3",
16             "webhookUsed": "false",
17             "webhookForSlotFillingUsed": "false",
18             "intentName": "Welzijn"
19         },
20         "fulfillment": {
21             "speech": "",
22             "messages": [
23                 {
24                     "type": 0,
25                     "speech": ""
26                 }
27             ]
28         },
29         "score": 1
30     },
31     "status": {
32         "code": 200,
33         "errorType": "success",
34         "webhookTimedOut": false
35     },
36     "sessionId": "87cd0cbe-945f-4974-8655-32afe14f4d44"
37 }
```

Figuur 20: JSON object met informatie van de zin

Zoals te zien is in Figuur 20 is de zin "Ik heb hoofdpijn" meegegeven. In dit object staat dat de parameter, of ook wel de entiteit, "Pijnpunt" is gevonden met de waarde "hoofdpijn". Ook is de intentie duidelijk. Daarnaast kan ook een context mee gegeven worden wanneer via de API met deze tool wordt gecommuniceerd om de intentie makkelijker te vinden, maar dit hoeft niet. Een eventueel antwoord wordt ook mee teruggestuurd. Verder is een score variabele aanwezig die aangeeft hoe groot de kans is dat de rest van de data correct is.

Dialogflow kan makkelijk verbonden worden met verschillende chat platforms. Hieronder vallen bijvoorbeeld Google Assistant, Facebook Messenger, Slack en nog

meer. Daarnaast heeft Dialogflow een groot aantal SDK's beschikbaar gesteld om met de API te communiceren. In Figuur 21 staat een uitwerking met de Node.js SDK.

```
1  var apiai = require('apiai');
2
3  var app = apiai("fad94235aabf41f19331faff602e930e");
4
5  var request = app.textRequest('Mijn hart is gebroken', {
6      sessionId: '<unique session id>'
7  );
8
9  request.on('response', function(response) {
10     console.log(response);
11 });
12
13 request.on('error', function(error) {
14     console.log(error);
15 });
16
17 request.end();
```

Figuur 21: Voorbeeldcode communicatie via SDK

```
{ id: '9b0f34c7-e0d7-4d5e-915e-2bbbcc9d11ea',
  timestamp: '2018-04-03T14:14:56.707Z',
  lang: 'nl',
  result:
    { source: 'agent',
      resolvedQuery: 'Mijn hart is gebroken',
      action: '',
      actionIncomplete: false,
      parameters: { Pijnpunt: 'hart' },
      contexts: [],
      metadata:
        { intentId: 'a291a1fc-41da-4228-93d4-1981113fdcd3',
          webhookUsed: 'false',
          webhookForSlotFillingUsed: 'false',
          intentName: 'Welzijn' },
      fulfillment: { speech: '', messages: [Array] },
      score: 0.9300000071525574 },
    status: { code: 200, errorType: 'success', webhookTimedOut: false },
    sessionId: '<unique_session_id>' }
```

Figuur 22: Voorbeeld JSON object response

Zoals te zien in Figuur 22 wordt apiai gebruikt. Dit is de Node.js SDK van Api.Ai, de eerste versie van Dialogflow. Dit is omdat de SDK's voor versie twee, Dialogflow, nog in beta is.

7.7.3. Conclusie

Dialogflow is een veelbelovende NLP tool. Het heeft een goede mogelijkheid om te leren en is makkelijk te trainen. Dankzij de grafische interface is het makkelijk om deze training uit te voeren. De zinnen worden ontleed naar entiteiten die gekoppeld zijn aan intenties. Ook heeft het SDK's wat het makkelijk maakt om te verbinden met andere onderdelen van een chatbot. De functionaliteiten die Dialogflow biedt worden niet onderdrukt met de oudere SDK's. Deze NLP tool zal meegenomen worden in verder onderzoek.

7.8. Conclusie

Na onderzoek te hebben gedaan naar verschillende chatbot applicaties zijn twee tools door de criteria gekomen. Deze tools zijn het MSBF en Hubot. Allebei bieden ze interessante functionaliteiten. Hubot is een eenvoudige tool om snel een chatbot te maken. Het MSBF heeft echter een grote hoeveelheid mogelijkheden te bieden.

Na onderzoek te hebben gedaan naar verschillende NLP tools zijn twee tools door de criteria gekomen. Deze tools zijn Wit.ai en Dialogflow. Deze twee tools lijken qua interface en mogelijkheden veel op elkaar, maar hebben toch een aantal verschillen. Dialogflow maakt bijvoorbeeld gebruik van intenties en entiteiten terwijl Wit.ai alleengebruik maakt van entiteiten. Hierbij verwacht Wit.ai dat een intent als entiteit gezien kan worden.

Voor verbinding met andere onderdelen heeft Dialogflow alleen SDK's, terwijl Wit.ai SDK's en een API heeft. Al deze tools hebben potentie in dit onderzoek en zullen om deze reden meegenomen worden.

8. Welke combinatie van tools is het meest kansrijk voor het realiseren van een chatbot?

In dit hoofdstuk staan de bevindingen beschreven die zijn opgedaan tijdens het in de praktijk koppelen van de verschillende tools. Het betreft de frameworks Hubot en MSBF. Voor beide frameworks zal een werkplaats onderzoek plaatsvinden om te onderzoeken of Wit.ai en Dialogflow te combineren zijn met de frameworks.

8.1. Hubot

Het is mogelijk om binnen Hubot een koppeling te maken naar andere tools. In de volgende paragrafen wordt beschreven hoe Hubot werkt in combinatie met de NLP tools Wit.ai en Dialogflow.

8.1.1. Wit.ai

Om Hubot met Wit.ai te koppelen, moet een bericht verstuurd worden naar de API van Wit.ai. In deze call moet een message meegegeven worden. Vervolgens moet een header toegevoegd worden aan de request. In deze header wordt de token van de Wit.ai applicatie meegegeven.

In Figuur 23 is te zien dat Hubot luistert naar een bericht die aan de reguliere expressie “/(.?)+/i” voldoet, dit zijn alle berichten. Vervolgens wordt het eerdergenoemde bericht verstuurd naar Wit.ai. De body is de informatie die door Wit.ai terug wordt gegeven, hier kan vervolgens een zin mee opgebouwd worden.

```
module.exports = (robot) ->
  TOKEN = "RK3HUK5TXE230CYUSQELT2AFZK54ACWS"

  robot.hear /( .?)+/i,(msg) ->
    message = msg.match[0]

    msg.robot.http("https://api.wit.ai/message?v=20180411&q=#{message}")
      .header("Authorization","Bearer #{TOKEN}")
      .get() (err,res,body)->
        console.log(body)
```

Figuur 23: Hubot met Wit.ai connectie

De body die de API call terugkrijgt wordt in dit voorbeeld gelogd. Het resultaat is in Figuur 24 te zien.

```
{"_text":"ik heb mijn arm gebroken","entities": {"message_body": [{"suggested":true,"confidence":0.85038,"value":"heb","type":"value"}],"pijnpunt": [{"confidence":1,"value":"arm","type":"value"}],"pijnssoort": [{"confidence":1,"value":"gebroken","type":"value"}]}, "msg_id": "0GEP4UiwyhE0cOSZ9"}
```

Figuur 24: De body retourneerd door Wit.ai

8.1.2. Dialogflow

Het is niet gelukt om Hubot te koppelen met Dialogflow. De poging begon bij het zoeken naar een NPM package om Dialogflow en Hubot te koppelen. Dit leek het onderzoeksteam eenvoudiger dan zelf alle requests maken. Hierbij is een NPM package gevonden waarbij enkel variabelen ingesteld moeten worden.

```
Hubot> (node:7636) UnhandledPromiseRejectionWarning: Error: Unexpected error while acquiring application default credentials: Could not load the default credentials. Browse to https://developers.google.com/accounts/docs/application-default-credentials for more information
    at GoogleAuth.<anonymous> (D:\projects\hubot\node_modules\google-auth-library\build\src\auth\googleauth.js:235:31)
    at step (D:\projects\hubot\node_modules\google-auth-library\build\src\auth\googleauth.js:47:23)
    at Object.next (D:\projects\hubot\node_modules\google-auth-library\build\src\auth\googleauth.js:28:53)
    at fulfilled (D:\projects\hubot\node_modules\google-auth-library\build\src\auth\googleauth.js:19:58)
    at <anonymous>:null:11
    at process._tickCallback (internal/process/next_tick.js:188:7)

(node:7636) UnhandledPromiseRejectionWarning: An unhandled promise rejection. This error originated either by throwing inside of an async function without a catch block, or by rejecting a promise which was not handled with .catch(). (rejection id: 3)
(node:7636) [DEP0018] DeprecationWarning: Unhandled promise rejections are deprecated. In the future, promise rejections that are not handled will terminate the Node.js process with a non-zero exit code.
(node:7636) UnhandledPromiseRejectionWarning: Error: Unexpected error while acquiring application default credentials: Could not load the default credentials. Browse to https://developers.google.com/accounts/do
    at GoogleAuth.<anonymous> (D:\projects\hubot\node_modules\google-auth-library\build\src\auth\googleauth.js:235:31)
    at step (D:\projects\hubot\node_modules\google-auth-library\build\src\auth\googleauth.js:47:23)
    at Object.next (D:\projects\hubot\node_modules\google-auth-library\build\src\auth\googleauth.js:28:53)
    at fulfilled (D:\projects\hubot\node_modules\google-auth-library\build\src\auth\googleauth.js:19:58)
    at <anonymous>:null:11
    at process._tickCallback (internal/process/next_tick.js:188:7)
```

Figuur 25: Foutmelding bij het starten van Hubot nadat de instellingen correct ingesteld waren

De foutmeldingen zoals zichtbaar in Figuur 25 kwamen direct na het starten van de chatbot. Na de URL te volgen die stond aangegeven in de foutmelding stond vermeld dat OAuth2 gebruikt kon worden. Deze suggestie is niet van toepassing, omdat een gebruiker niet kan inloggen met OAuth2 bij Dialogflow. Om deze reden is besloten verder te zoeken naar een andere oplossing voor deze fout.

Na verder onderzoek naar een oplossing zonder resultaat is besloten om het onderzoek naar Hubot met Dialogflow te staken. Hubot met Dialogflow wordt voor dit onderzoek als niet functioneel beschouwd.

8.2. Microsoft Bot Framework

Het is mogelijk om binnen het MSBF een koppeling te maken naar andere tools. In de volgende paragrafen wordt beschreven hoe MSBF werkt in combinatie met de NLP tools Wit.ai en Dialogflow.

8.2.1. Wit.ai

Om Wit.ai met het MSBF te koppelen is een NPM module gebruikt (Joffreybvn, 2017). Hiermee is het mogelijk om een eigen Wit.ai project met het MSBF te koppelen. Het MSBF gebruikt een recognizer die een zin bestudeert en hier de intent uit haalt. De NPM package koppelt een eigen recognizer aan de MSBF recognizer. Om dit initialiseren moet de access token worden toegevoegd zoals te zien in Figuur 26.

```
const recognizer = new WitRecognizer('YAWN0BUHPP62GIT5FASUHN6H73J2UZQF');
// Start WIT.AI Recognizer
bot.recognizer(recognizer);
```

Figuur 26: Koppeling van Wit.ai met het Microsoft Bot Framework

Na vele pogingen om de NPM package werkend te krijgen is het niet gelukt om intents uit de zin terug te krijgen. Omdat Wit.ai geen intents gebruikt maar lookup strategies, is het onmogelijk om gebruik te maken van de functionaliteiten van de recognizer.

Vervolgens is geprobeerd middleware te gebruiken om de request van het bericht binnen te halen en hier handmatig de intent uit te halen. Dit werkte echter ook niet, omdat deze manier niet officieel ondersteund wordt, en daarom veel opties van het framework niet kan gebruiken.

Daarna is nog een andere NPM package gebruikt (Sylvester, 2018) om Wit.ai te koppelen. Dit werkte vrijwel hetzelfde als de andere package waar een eigen recognizer gebruikt wordt. Deze recognizer maakt een vertaling naar de recognizer van het MSBF. De implementatie voor het zoeken van de intent is bij deze module echter iets anders. Hier kan een intent meegegeven worden en vervolgens als zin doorgegeven worden. Wanneer entiteiten gevonden worden, wordt hier een match gemaakt. Indien dit niet het geval is wordt een standaardwaarde teruggegeven Figuur 27. Ook met deze recognizer werden de intents niet herkend door het framework.

```
intents.matches('intent.pijnpunt', (session, args) => { console.log('matches') })  
intents.onDefault(session => { console.log('default') })  
bot.dialog('/', intents)
```

Figuur 27: Intent match

8.2.2. Dialogflow

In deze paragraaf wordt beschreven of het mogelijk is om het MSBF met de NLP tool Dialogflow te koppelen. Het project wat wordt gebruikt voor het koppelen van de twee tools is gemaakt tijdens het beantwoorden van deelvraag drie.

Tijdens het zoeken naar informatie voor het koppelen van het MSBF en Dialogflow is een speciale NPM package gevonden die de connectie regelt. Deze NPM package heet “api-ai-recognizer”. (Akkulka, 2017)

Wel wordt gebruik gemaakt van een NPM package die gemaakt is door een externe partij. Dit kan als nadeel hebben dat de package niet meer wordt bijgewerkt.

De connectie wordt gemaakt door de client token van Dialogflow mee te geven aan de recognizer, zie Figuur 28 voor de connectie. Alle berichten worden via de “intents” verstuurd en worden gecontroleerd of ze tot de intent behoren.

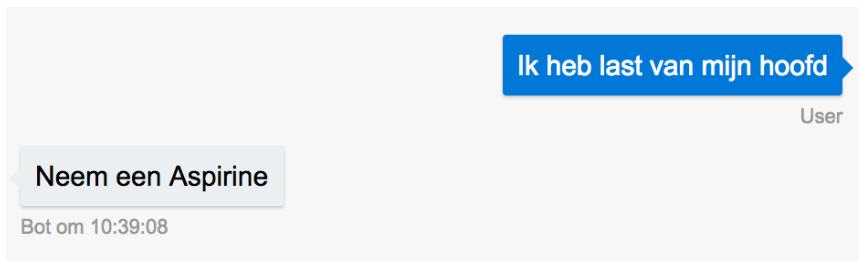
```
var recognizer = new apiairecognizer("54ae2c103cd42b5b65c4d4cd120ed25");  
var intents = new builder.IntentDialog({  
  |  recognizers: [recognizer]  
});
```

Figuur 28: Connectie tools

Het controleren van het bericht wordt gedaan via de “intents.matches”. Als het bericht de juiste intent heeft wordt gecontroleerd of de entiteit goed is. Als dit klopt zal een advies worden gegeven. Zie Figuur 29 voor de code en zie Figuur 30 voor het advies.

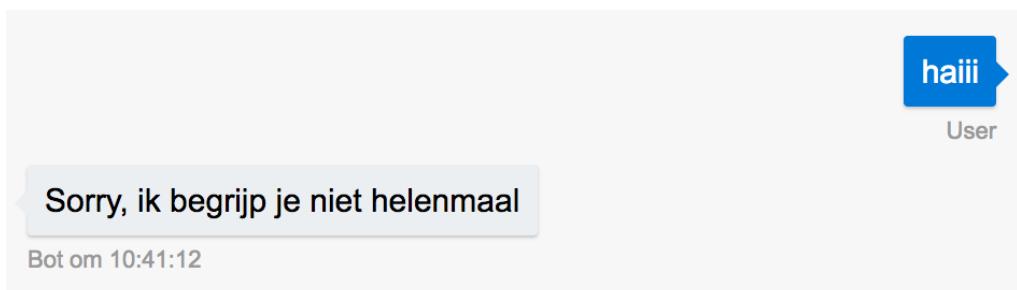
```
intents.matches('KrijgAdvies', [
  function(session, args){
    var pijnpunt = builder.EntityRecognizer.findEntity(args.entities, 'PijnPunt');
    if (pijnpoint){
      builder.Prompts.text(session, "Neem een Aspirine");
    }
  }
]);
```

Figuur 29: Intent match code



Figuur 30: Intent match UI

In het voorbeeld in Figuur 31 wordt een bericht verstuurd die niet binnen een intent valt. Wanneer dit gebeurt wordt een standaard bericht verstuurd.



Figuur 31: Geen intent match UI

8.3. Voor- en nadelen

In deze paragraaf worden de voor- en nadelen van de gevonden tools besproken. Zoals bij de vorige deelvragen zijn de tools opgedeeld in twee categorieën. Deze categorieën zijn frameworks en NLP tools. Beide frameworks zijn met beide NLP tools gekoppeld. De voor- en nadelen zijn naar aanleiding van experimenten gevonden.

8.3.1. Frameworks

In Tabel 4 zijn de meest opvallende functionaliteiten en eigenschappen van de frameworks opgesomd. Daarbij wordt per functionaliteit of eigenschap aangegeven of het team dit als positief of negatief punt beschouwd.

Hubot	MSBF
- CoffeeScript	+ Meer tools beschikbaar
- Weinig standaard functionaliteit	+ Meer informatie beschikbaar
	- Verwacht specifieke bestandsstructuur

Tabel 4: Voor- en nadelen van frameworks

8.3.2. Hubot

Het onderzoeksteam beoordeelt CoffeeScript als een negatief punt, omdat dit een andere syntax heeft als eerdere talen waar het team mee heeft gewerkt. Het leren van een nieuwe syntax tijdens het creëren van een prototype of proof of concept is gezien de beperkte tijd voor het onderzoek niet verstandig.

Daarnaast wordt de gebrekkige standaard functionaliteit als een negatief punt gezien. De standaard functionaliteit vereist dat veel zelf gedaan moet worden. Door het gebrek aan ervaring binnen het team wordt dit gezien als een negatief punt. Ook is de zelfgeschreven code minder goed getest dan onderdelen van frameworks.

8.3.3. Microsoft Bot Framework

Het eerste dat opviel bij het onderzoek was dat het internet veel informatie heeft met betrekking tot het MSBF. Veel van deze informatie is ook afkomstig van Microsoft zelf wat als betrouwbaar beschouwd kan worden.

Het MSBF heeft ook voor bijna alle gewenste functionaliteiten wel een package beschikbaar. Hierdoor zal vaker een package geïmplementeerd worden dan dat de functionaliteiten geprogrammeerd worden. Ondanks dat de vrijheid beperkt is bespaart het veel tijd.

Een nadeel is wel dat als een package niet beschikbaar is, het connector onderdeel een specifiek formaat van JSON berichten verwacht. Hierdoor moet altijd een wrapper geschreven worden voor een externe dienst. Dit is logisch, maar het kost als developer vaak wel extra tijd. Daarnaast is momenteel niet bekend of alle functionaliteiten van een andere tool volledig gebruikt kunnen worden wanneer het resultaat omgezet moet worden naar het voorgeschreven JSON formaat.

8.3.4. NLP tools

In Tabel 5 staan de meest opvallende punten van de verschillende NLP tools. Per onderdeel zal bekijken worden of het team dit als positief of negatief ervaart.

Dialogflow	Wit.ai
+ Heeft intents	- Geen werkende recognizer met MSBF
+ Werkende recognizer met MSBF	- Heeft lookup strategies
+ Werkt wel met MSBF	+ Werkt met beide frameworks
+ Kan zelf context bijhouden	- Kan geen context bijhouden
- Werkt niet met Hubot	

Tabel 5: Voor- en nadelen van de NLP tools

8.3.5. Dialogflow

Het team heeft het hebben van intents als positief punt beoordeeld. Aan een intent kunnen woorden en zinnen gekoppeld worden waarmee de intentie van de zin opgehaald kan worden. Dit zorgt ervoor dat binnen het MSBF de zin op de juiste manier wordt ontleed. Hierdoor hoeft het team zelf geen recognizer te schrijven om de intent van de zin op te halen. Dit is dan ook de reden dat de recognizer met het MSBF als pluspunt staat. Verder staat als pluspunt dat Dialogflow werkt met het MSBF. Het nadeel is dat hij niet werkend te krijgen is met Hubot. Hier was ook geen informatie over te vinden. Verder kan Dialogflow zelf zijn context bijhouden. Het is variabel hoelang de context onthouden moet worden.

8.3.6. Wit.ai

Het pluspunt dat Wit.ai te bieden heeft voor het team is dat het te koppelen is met beide frameworks. Wat een groot verschil is tussen Wit.ai en Dialogflow is dat Dialogflow met intents werkt en Wit.ai met lookup strategies. Met 'lookup strategies' moet voor elk woord dat ergens bij hoort gekeken worden of het een bijwoord of zelfstandig naamwoord is. Op deze manier wordt besloten bij welk onderwerp de vraag hoort. Doordat dit handmatig ingesteld moet worden wordt dit gezien als minpunt. De recognizer van het MSBF werkt op intenties. Daardoor is een gedeelte van het MSBF niet bruikbaar. Om deze reden is gekozen om dit als negatief punt te ervaren. Als laatste negatieve punt is ondervonden dat Wit.ai geen context bij kan houden. Dit zou zelf geprogrammeerd moeten worden.

8.4. Conclusie

Uit de vier combinaties vielen twee al snel af. De MSBF met Wit.ai combinatie werd geen succes, omdat Wit.ai geen gebruik maakt van intents waar het MSBF op bouwt.

De combinatie Hubot met Dialogflow liep te stroef om door te gaan met deze combinatie. Dit had te maken met de vele foutmeldingen waardoor veel tijd verloren ging aan het oplossen van deze fouten. Zo is het gezien de tijd niet mogelijk om de functionaliteiten en het koppelen hiervan zelf te realiseren. Doordat de syntax van CoffeeScript zo verschilt van wat binnen het team bekend is, kost het meer tijd om functionaliteit in CoffeeScript te realiseren. Het probleem hierbij is dat voor deze casus de tijd beperkt is.

De twee combinaties die overbleven, MSBF met Dialogflow en Hubot met Wit.ai, zijn onderzocht door de voor- en nadelen naast elkaar te zetten. Het resultaat hiervan was dat de combinatie MSBF met Dialogflow meer mogelijkheden biedt. Waar Hubot met Wit.ai stopt, gaat MSBF en Dialogflow verder. Het onderzoek zal met het MSBF in combinatie met Dialogflow worden voortgezet.

9. Welke taken kan NLP uitvoeren?

In dit hoofdstuk wordt beschreven wat NLP kan bieden voor deze casus. Dit wordt beschreven in de volgende onderdelen:

- Een beschrijving van NLP
- De onderdelen van NLP
- Het toepassen van NLP

9.1. Natural Language Processing

NLP is een onderdeel van Artificial Intelligence dat menselijke taal omzet naar een formaat dat de computer begrijpt. Wanneer een computer betekenis kan geven aan een menselijk bericht kunnen acties uitgevoerd worden op basis van de ontvangen berichten.

De volgende problemen kunnen zich voordoen bij het omzetten van menselijke berichten naar een formaat dat de NLP kan begrijpen:

- Woorden zijn homoniem.
- In verschillende talen hebben leestekens andere functionaliteiten.
- In verschillende talen gelden andere spelling- en grammaticaregels.
- Spelling- en grammaticaregels hebben uitzonderingen.
- Talen hebben dialecten.

Voor bovenstaande problemen zijn complexe oplossingen gevonden. Deze oplossingen bestaan voornamelijk uit wiskundige formules. Deze wiskundige formules zullen niet voorkomen in de komende paragraaven gezien de complexiteit.

9.2. De onderdelen van Natural Language Processing

In deze paragraaf worden de interne onderdelen beschreven die samen een NLP vormen. Per onderdeel staat beschreven wat het onderdeel doet en welke problemen dit oplost. De volgende onderdelen worden hieronder beschreven:

- Tokenization
- Stop word removal
- N-Grams
- Word sense disambiguation
- Part of speech tagging
- Verkleinen van woorden

(Kolalapudi, 2016)

9.2.1. Tokenization

Tokenization wordt gebruikt om een menselijk bericht op te delen in stukken, zoals zinnen en woorden (Standford, 2009), (IBM, 2013). Door tokenization als eerste uit te voeren kunnen de andere onderdelen worden toegepast op het resultaat.

Hoe Tokenization plaatsvindt is afhankelijk van de taal. Zo gebruiken niet alle talen spaties om het einde van een woord aan te geven, zoals (Standford, 2009):

- Chinees
- Koreaans
- Thais

In de Nederlandse taal wordt vaak een ‘.’ als afsluitingen van een zin gezien, maar hier zijn ook uitzonderingen voor, zoals: “Dr. Janssen”. Hierbij wordt de ‘.’ gebruikt om aan te tonen dat het woord is afgekort. Een mogelijke oplossing voor deze uitzonderingen is het vervangen van de afkorting door de voluit geschreven woorden. Het woord wordt dan: “Dokter Janssen”. Hierna kunnen de zinnen op de juiste plekken worden gescheiden.

Het scheiden van zinnen kan ook worden uitgevoerd door de resterende leestekens. Dit wordt gedaan op basis van de hierboven genoemde leestekens. Deze methode kan echter problemen geven wanneer een zin eindigt met “etc.”. Hierbij is het resultaat dat “etc.” vervangen wordt door etcetera. Dit houdt in dat de zin niet meer wordt afgekapt met als gevolg dat de tekst verkeerd wordt opgedeeld.

Naast de afkortingen met punten doen zich ook problemen voor met het gebruik van leestekens, zoals bijvoorbeeld een apostrof of streepje. In de tekst moet ook rekening gehouden worden met entiteiten die niet overeenkomen met de taalregels zoals:

- Telefoonnummer
- E-mailadres
- Datum

Het resultaat van tokenization is te zien in Figuur 32.

Input	Ik heb hele erge hoofdpijn. Dit is gisteren begonnen.
Output	[‘Ik heb hele erge hoofdpijn’, ‘Dit is gisteren begonnen’]

Figuur 32: Het resultaat van Tokenization, met bovenstaande input

9.2.2. Stop word removal

Stop word removal wordt gebruikt voor het verwijderen van stopwoorden uit een menselijk bericht. Stopwoorden hebben verschillende toepassingen:

- Bij presentaties kunnen stopwoorden gebruikt worden voor het opvullen van pauzes.
- Bij een chatbot kan het voorkomen dat een gebruiker stopwoorden toevoegt om het bericht te verduidelijken.

(Etsel, 2015)

De berichten die binnenkomen worden gecontroleerd op stopwoorden. De stopwoorden worden herkend met behulp van een lijst met alle bekende stopwoorden in de gebruikte taal. (Geeks for Geeks, n.d.). Een voorbeeld van een lijst met bekende Nederlandse stopwoorden is bijgevoegd als Bijlage A.

De stopwoorden zullen verwijderd worden zonder dat de betekenis van de zin verloren gaat. Het voordeel van stop word removal is dat de processnelheid van NLP hoger. (Geeks for Geeks, n.d.)

Figuur 33 bevat een voorbeeld van de stop word removal. Zoals te zien is zijn de stopwoorden bij de output verwijderd.

Input	Ik heb hele erge hoofdpijn. Dit is gisteren begonnen.
Output	[‘hele erge hoofdpijn’, ‘gisteren begonnen’]

Figuur 33: Stop word removal

9.2.3. N-Grams

N-Grams kijkt naar een set woorden of leestekens die achter elkaar staan. Op deze manier vindt NLP een verband met de komende woorden. De woorden die achter elkaar geschreven worden hebben vaak een relatie met elkaar. (Mitra, 2002)

N-grams slaat de relatie op met een variabele n . n staat voor het aantal woorden of leestekens. n moet minimaal 1 zijn en maximaal de tekst lengte. Hierdoor kan de NLP tool controleren hoe de zin opgebouwd is. Vervolgens kan gecontroleerd worden of de zin klopt, als dit niet zo is zal de NLP tool dit kunnen verbeteren. Zoals zichtbaar in Figuur 34 kan een getraind model de zin ontleden als hier geen spaties in voorkomen.



Figuur 34: N-Grams voorbeeld van Microsoft

9.2.4. Word sense disambiguation

Word sense disambiguation wordt gebruikt om de definitie te bepalen bij woorden die homoniem zijn. Hierbij kan het voorkomen dat entiteiten op basis van de zin veranderen (Alok & Diganta, 2015). Zo kan “haar” een bezittelijk voornaamwoord zijn, maar het kan ook refereren naar lichaamshaar. Het woord haar is dus homoniem.

Wanneer de homoniemen zijn gevonden kunnen de juiste entiteiten worden geëxtraheerd. In Figuur 35 is een voorbeeld te zien van een woord die homoniem is.

Betekenis 1	Ik heb last van mijn hoofd.
Betekenis 2	Zij zit aan het hoofd van de tafel.

Figuur 35: Voorbeeld waarbij het hoofd homoniem is

(Hofmann & Ganea, 2017), (Hongzhao, Larry, & Heng, 2015)

9.2.5. Part Of Speech Tagging

Part of speech tagging wordt gebruikt voor het verder ontleden van een zin in individuele woorden. Aan elk van deze woorden zal één van de onderstaande tags worden toegevoegd:

- Adjective (Bijvoeglijk naamwoord)
- Adverb (Bijwoord)
- Conjunction (Voegwoord)
- Determiner (Verwijzing)
- Noun (Zelfstandig naamwoord)
- Number (Nummer)
- Preposition (Voorzetsel)
- Pronoun (Voornaamwoord)
- Verb (Werkwoord)

Deze tags staan gelijk aan woordsoorten, dit kan per taal verschillen door verschillende grammatica- en spellingsregels. Over het ontleden van een zin is geen onderzoek gedaan, omdat dit taalkundige regels zijn. Een voorbeeld is de ENGTWOL methode.

(Hongzhao, Larry, & Heng, 2015), (Robin, 2009), (Jurafsky & Martin, 2018)

9.2.6. Verkleinen van woorden

Stemming en lemmatization verkleinen alle varianten van een woord naar dezelfde basisvorm. Het voordeel van het verkleinen van woorden zorgt voor een snellere processtijd van de uitkomst. Een ander voordeel is dat, maar één entiteit nodig is voor meerdere varianten van een woord.

(Mohanoor, Natural Language Processing Glossary for Programmers, 2017), (Jivani, 2011)

Stemming

Bij de stemming methode wordt het woord verkleind naar de kortst mogelijk vorm. Hierbij zal niet worden gekeken of de betekenis van een woord nog klopt. De kortst mogelijke vorm wordt ook wel "stem" genoemd. (Mohanoor, Dialogflow Machine Learning Algorithm, 2018)

Input	Ik heb hele erge hoofdpijn. Dit is gisteren begonnen.
Output	ik heb hel erg hoofdpijn. dit is gister beg.

Figuur 36: Stemming voorbeeld

Het meest gebruikte stemming algoritme voor de Engelse taal is de Porter stemmer. Deze is ook beschikbaar voor de Nederlandse taal. De Porter stemmer bestaat uit vijf fases. Elke fase heeft een eigen wiskundige formule. Alternatieve algoritmes voor stemming zijn:

- Snowball algoritme
- Lovins algoritme
- Paice/Hush algoritme
- Dawson algoritme.

(Standford, 2009), (Kraaij & Pohlmann, n.d.)

Lemmatization

Bij de stemming methode wordt het woord verkleind naar de kortst mogelijk vorm, waarbij het woord nog zijn betekenis heeft. Hierbij wordt rekening gehouden met de vorm van het woord (part of speech tagging)

(Mohanoor, Natural Language Processing Glossary for Programmers, 2017)

één van de lemmatization algoritmes is het "Lemmald" algoritme. In dit algoritme wordt gebruik gemaakt van machine learning. (Nordström & Ranta, 2008) Voor lemmatization moeten de betekenis van woorden ergens geformuleerd zijn, bijvoorbeeld in een database. Een voorbeeld hiervan is WordNet. Om lemmatization toe te passen moet de part of speech tagging techniek uitgevoerd zijn. Een nadeel van lemmatization is dat wanneer een woord niet bekend is binnen de gebruikte bron voor bestaande woorden, lemmatization dit woord overslaat. (Tunkelang, 2017)

Input	Ik heb hele erge hoofdpijn. Dit is gisteren begonnen.
Output	ik heb hele erge hoofdpijn. dit is gister begin.

Figuur 37: Voorbeeld lemmatization

9.2.7. Toepassing

Binnen NLP zijn verschillende manieren om de NLP onderdelen uit te voeren, zoals:

- Rule-based approach
- Machine learning approach
- Hybrid approach

Rule-based approach

Binnen de rule-based approach worden alle regels handmatig toegevoegd, dit maken de regels statisch. In Figuur 38 wordt de flow van een rule-based approach getoond. Wanneer een mail binnenkomt zal door middel van de statische regels worden

gekeken of dit spam of ham is en wordt de locatie van de mail bepaald. Spam is data wat overbodig wordt bevonden en ham is data die van toepassing is.



Figuur 38: Rule-based flow

Voor- en nadelen rule-based approach

In Tabel 6 worden de voor- en nadelen weergeven van de rule-based approach.

Voordelen	Nadelen
Eenvoudig aan te passen	Developer moet kennis hebben van de taalregels
Geen trainingsdata nodig	Ontwikkelen van eerste versie duurt lang
Accuraat	

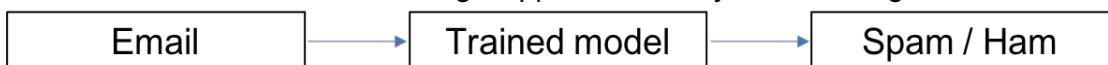
Tabel 6: Voor- en nadelen van rule-based approach

(Dorash, 2017), (Freed, Demo of natural language processing with rules and machine-learning based approaches, 2017)

Machine learning approach

Bij de machine learning approach wordt door middel van machine learning en voldoende trainingsdata de regels zelf gemaakt. Dit gebeurt op basis van eigenschappen in de data, hierdoor zijn de regels dynamisch. De flow van de NLP tool bij het gebruik van machine learning approach is zichtbaar op afbeelding d.

De data komt binnen en gaat door het getrainde model. Deze geeft een antwoord afhankelijk van waar het model op getraind is. Binnen het getrainde model is ook een dataset beschikbaar, deze is weer gekoppeld aan de dynamische regels.



Figuur 39: Proces van machine learning approach

Voor- en nadelen machine learning approach

In Tabel 7 worden de voor- en nadelen weergeven van de machine learning approach.

Voordelen	Nadelen
Dynamische regels	Voor elke taak die uitgevoerd moet worden is veel trainingsdata nodig
Leert van voorgaande resultaten	Moeilijk te debuggen
Makkelijk schaalbaar	

Tabel 7: Voor- en nadelen van machine learning approach voor het maken van eigen NLP

(Freed, Demo of natural language processing with rules and machine-learning based approaches, 2017), (Dorash, 2017), (Freed, Comparing rules and machine learning

natural language processing approaches, 2017) (Freed, Improving simple natural language processing models with rules or machine learning, 2017)

Hybrid approach

Bij de hybrid approach wordt gebruik gemaakt van zowel statische als dynamische regels. Hiermee worden de “rule-based” en “machine learning” manieren gecombineerd om het beste van beide te gebruiken. Hierbij worden initieel statische regels gebruikt, die worden aangevuld met de regels die machine learning maakt. (Dorash, 2017)

9.3. Taken die uitgevoerd kunnen worden door Dialogflow

Wanneer de onderdelen van NLP zijn uitgevoerd en de zin helemaal ontleed is kunnen de taken van Dialogflow worden uitgevoerd. De volgende taken zijn toegepast in deze casus met Dialogflow:

- Named entity recognition
- Spellingscontrole

Omdat Dialogflow de source-code niet vrijgeeft maakt het onderzoeksteam de hypothese dat het volgende gebeurd:

- Het menselijke bericht wordt ontleed door tokenization
- De stopwoorden zullen worden verwijderd door stop word removal
- Aan de woorden zullen woordsoorten worden toegevoegd door part of speech tagging
- De woordsoorten worden vergeleken met de entiteiten die door de gebruiker zijn gedefinieerd.
- De woordsoorten worden gekoppeld aan een entiteit.

9.3.1. Named entity recognition

Het koppelen van woordsoorten met entiteiten wordt ‘Named entity recognition’ genoemd.

Dit houdt in dat delen van berichten aparte tags krijgen, zoals pijnpunt en pijnsoort. (Stanford, n.d.)

„ Mijn vader heeft hevige pijn aan zijn voorhoofd			
PARAMETER NAME	ENTITY	RESOLVED VALUE	
Geslacht	@Geslacht	vader	×
Intensiteit	@Intensiteit	hevige	×
PijnPunt	@PijnPunt	voorhoofd	×

Figuur 40: Dialogflow zin

Zoals in Figuur 40 is te zien geeft de gebruiker aan dat zijn/haar vader hevige pijn aan zijn voorhoofd heeft. Wanneer het bericht van de gebruiker is behandeld door Dialogflow kunnen meerdere entiteiten worden gereturneerd. In het voorbeeld van afbeelding a zijn dat drie verschillende entiteiten:

- Geslacht
- Intensiteit
- PijnPunt

Binnen de entiteit geslacht zijn twee varianten beschikbaar dit zijn 'man' en 'vrouw'. De gebruiker gaf aan dat het om zijn of haar vader ging, omdat vader zich als synoniem in de variant man bevindt wordt vader gecategoriseerd op het geslacht man (zie Figuur 41).

Figuur 41: Dialogflow synoniemen

Het voordeel van synoniemen bij een variant in de entiteiten is dat Dialogflow sneller kan herkennen dan wanneer maar één variant beschikbaar is. Elke gebruiker kan zijn of haar vader anders noemen zoals bijvoorbeeld pap. In Figuur 42 wordt aangetoond dat daadwerkelijk gegroepeerd wordt op het geslacht man.

Figuur 42: Dialogflow resultaat man

9.3.2. Spellingscontrole

Bij spellingscontrole wordt gecontroleerd op de relatie van letters en woorden. Hierna kan Dialogflow berekenen welke woorden achter elkaar horen. Als hier een

spellingsfout in zit, wordt door middel van de wiskundige formule een suggestie gedaan met het correct gespelde woord.

Dialogflow beschikt niet standaard over een spellingscontrole. Om ervoor te zorgen dat de chatbot een spellingsfout kan begrijpen, moet dit geconfigureerd worden binnen Dialogflow.

Om ervoor te zorgen dat Dialogflow de tekst kan begrijpen, kunnen synoniemen voor een entiteit worden opgegeven. Om veel voorkomende spelfouten te vinden is de trainings optie binnen Dialogflow beschikbaar. Hier worden alle regels toegevoegd die naar de chatbot getypt worden. In Figuur 43 is een voorbeeld te zien van een gebruiker die buik verkeerd schrijft.

De zin in Figuur 43 is nu zichtbaar binnen de training pagina, zoals te zien is in Figuur 44

The screenshot shows the Dialogflow Agent interface. At the top, there is a blue button labeled 'Set-up Google Assistant integration.' Below this, the word 'Agent' is written in blue. The interface is divided into sections: 'USER SAYS' on the left and 'COPY CURL' on the right. A user message 'ik heb last van mijn biuk' is listed under 'USER SAYS'. To the right of this message is a 'COPY CURL' button. Below this section is a 'DEFAULT RESPONSE' section with a dropdown menu currently set to 'IntentUnknown'. Further down, there is an 'INTENT' section labeled 'Default Fallback Intent'. At the bottom, there is an 'ACTION' section labeled 'input.unknown'.

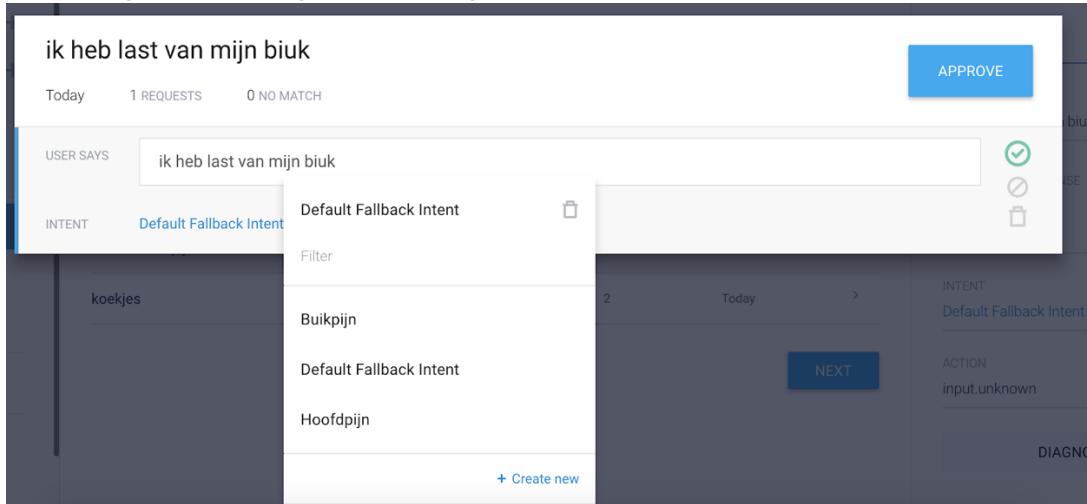
Figuur 43: Buik verkeerd gespeld

The screenshot shows the Dialogflow Training tab. At the top, there is a 'Training' button with a graduation cap icon and a 'UPLOAD' button. Below this is a table with columns: 'Conversation', 'Requests', 'No match', 'Date', and a delete icon. The table contains the following data:

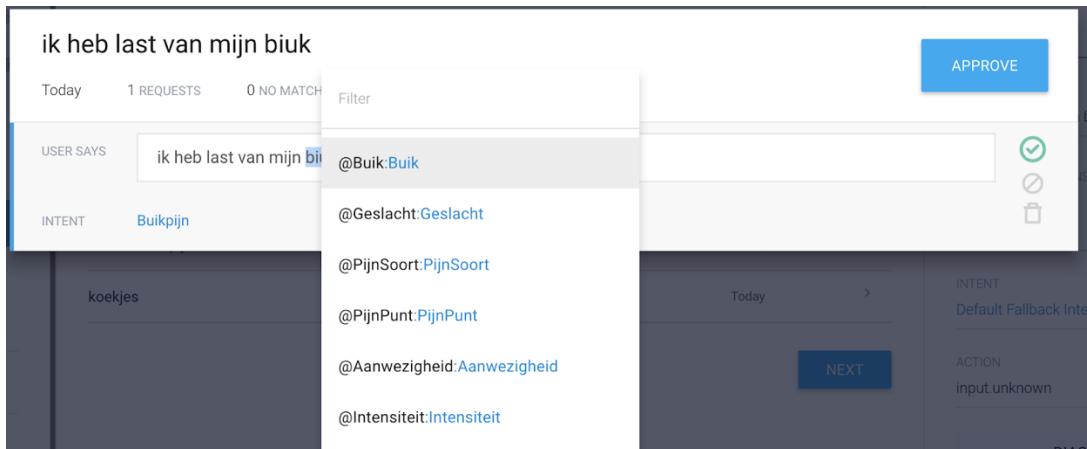
Conversation	Requests	No match	Date	
ik heb last van mijn biuk	1	0	Today	>
Mijn vader heeft buikpijn	6	0	Today	>
ik heb buikpijn	5	4	Today	>
ik heb hevige buikpijn onder in mijn maag	1	0	Today	>

Figuur 44: Training tab met de verkeerd gespelde zin

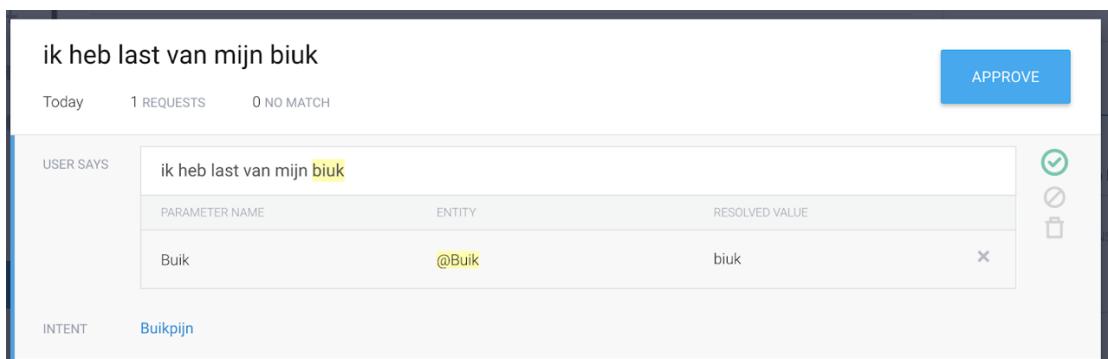
Wanneer de zin wordt geselecteerd zal een nieuw venster worden geopend, met daarin de intenties en entiteiten die gekoppeld kunnen worden aan de zin. Dit is te zien in Figuur 45 en Figuur 46. in Figuur 47 is het resultaat te zien.



Figuur 45: intent toevoegen



Figuur 46: Entiteit koppelen aan het woord



Figuur 47: Approven van een zin

Wanneer de zin toegevoegd is aan de trainingsdata is deze beschikbaar op de “intents pagina” binnen Dialogflow, zoals in Figuur 48.

 Add user expression
 ik heb last van mijn biuk
 ik heb last van mijn buik
 ik heb hevige krampen rechtsonder mijn maag
 Mijn vader heeft een opgeblazen gevoel in zijn maag
 ik heb een brandend gevoel in mijn bovenbuik

Figuur 48: Toegevoegde trainingsdata zin

De laatste stap is om synoniemen toe te voegen aan de varianten van een entiteit, zoals in Figuur 49 wordt “buikpijn” toegevoegd aan de variant “Buik”.

Buik	Buik	Bast	Pens	Kas	Maag
Schoot	Enter synonym				
Buikpijn	Buikpijn				

Figuur 49: Entiteit zonder biuk

Het resultaat van het toevoegen van de synoniem aan de variant “Buikpijn” is te zien in Figuur 50.

Buik	Buik	Bast	Pens	Kas	Maag	
Schoot	Biuk	Enter synonym				
Buikpijn	Buikpijn					

Figuur 50: Biuk toegevoegd

Als nu de zin verkeerd getypt wordt, zal deze toch aan de juiste entiteit worden verbonden, zoals te zien is in Figuur 51. Dialogflow voegt helaas zelf niet de synoniem toe wanneer dit aangegeven wordt in de training. Dit moet handmatig worden toegevoegd.

USER SAYS COPY CURL

ik heb last van mijn buik

DEFAULT RESPONSE ▾

Not available

INTENT

Buikpijn

ACTION

Not available

PARAMETER	VALUE
Intensiteit	
PijnPunt	
Aanwezigheid	
Buik	biuk
Geslacht	
PijnSoort	

Figuur 51: Typfout met juiste intent

9.4. Conclusie

Wanneer de onderdelen van NLP samengevoegd worden, is het mogelijk om menselijke berichten om te zetten naar een formaat die door de computer begrepen wordt.

NLP kan op drie verschillende manieren uitgevoerd worden:

- De rule-based approach is een intensieve manier, wanneer een eerste versie opgeleverd moet worden. Dit komt doordat de regels statisch worden geformuleerd. Hierbij wordt het snel complex wanneer uitzonderingen op regels toegevoegd moeten worden.
- De Machine learning approach. Hierbij moet de developer kennis hebben van machine learning. Het voordeel hiervan is dat de regels dynamisch worden geformuleerd, echter kost het veel tijd om alle trainingsdata goed voor te bereiden.
- De hybrid approach. Hierbij worden statische regels geformuleerd, zodat minder trainingsdata benodigd is voor dezelfde resultaten.

De NLP manier die wordt gebruikt binnen deze casus is de hybrid approach. Deze keuze is gemaakt omdat beperkte tijd beschikbaar was voor dit project, daarnaast bevat het projectteam geen taalkundige om alle regels op te stellen. Hierdoor kan het machine learning aspect van de hybrid approach verfijning bieden.

Binnen deze casus wordt Dialogflow gebruikt voor het aanleveren van entiteiten, deze worden vervolgens gebruikt bij het classificeren van een klacht.

10. Hoe houdt de chatbot een conversatie met de gebruiker?

Een chatbot moet op basis van een gegeven antwoord een vervolgvraag kunnen stellen. Ook moet een chatbot om kunnen gaan met onverwachte gebruikersinvoer, zoals typefouten of ongerelateerde informatie. In dit hoofdstuk wordt onderzocht:

- Welke methoden bestaan om een conversatie te houden met een gebruiker.
- Hoe verschillende algoritmes hiervoor gebruikt worden.
- Hoe deze algoritmes werken.
- Welke tools het MSBF hier beschikbaar voor stelt.
- Welke externe tools met deze kwestie kunnen helpen.

10.1. Conversatie methoden

Verschillende methoden bestaan om een conversatie gaande te houden. In deze paragraaf worden deze methoden onderzocht en gekeken naar mogelijkheden hiervoor.

10.1.1. Waterfall

Bij deze methode worden de vragen in een vooraf gedefinieerde volgorde gesteld door de chatbot. Nadat de gebruiker de vraag heeft beantwoord stelt de chatbot de vervolgvraag. De vervolgvragen worden bepaald door eigen geschreven hardcoded regels die gelden voor het gegeven antwoord van de gebruiker. De gebruiker zal bij deze methode nooit een eerder gestelde vraag opnieuw kunnen beantwoorden. (Vo, Iqbal, & Standefer, Manage conversation flow with dialogs, 2017)

10.1.2. Intent based

De intent based methode reageert op basis van de intentie en entiteiten die gevonden zijn in de zin door middel van een NLP tool. Dit levert altijd een conversatie van niet meer dan één bericht van de gebruiker. Het bericht van de chatbot is namelijk volledig gebaseerd op wat de gebruiker stuurt. De vervolgvraag wordt bepaald door de gegeven intentie, entiteit en mogelijke context. De context geeft aan welk antwoord gegeven is. Op Figuur 52 is een intent based conversatie te zien zonder context. (new gen apps, 2018)



Figuur 52: Intent based conversatie.
Overgenomen uit Twitter van (Zadeh, 2016)

10.1.3. Flow based

Deze methode is grotendeels hetzelfde als de *waterfall* methode. Met deze methode wordt echter meer gebruik gemaakt van wat de gebruiker zegt. De chatbot kan een set vragen stellen op basis van wat de gebruiker stuurt. Op deze manier kan het onderwerp van een conversatie van globaal naar specifiek gaan. Daarnaast is het ook mogelijk om terug te gaan naar een eerder punt in het gesprek. Om deze methode te implementeren wordt een decision algoritme gebruikt. Deze methode is van de drie het meest flexibel omdat op basis van wat de gebruiker zegt berichten worden gestuurd, maar dit ook beïnvloed kan worden door de code. (new gen apps, 2018)

10.1.4. Combinatie

Door een combinatie van *intent based* en *flow based* te maken kunnen sets met vragen gekozen worden op basis van de intentie en de entiteiten. Dit levert het optimale resultaat omdat de chatbot verschillende sets vragen kan stellen om specifieke informatie te vinden en ook optimaal op de informatie van de gebruiker kan inspelen met de intenties en entiteiten.

Om deze combinatie te implementeren wordt een NLP tool gebruikt in combinatie met een decision algoritme. Decision algoritmes wordt in het volgende paragraaf uitgelegd. Deze methode zal tijdens het maken van het prototype gebruikt worden, omdat deze methode de meeste mogelijkheid voor complexiteit biedt.

10.2. Decision algoritmes

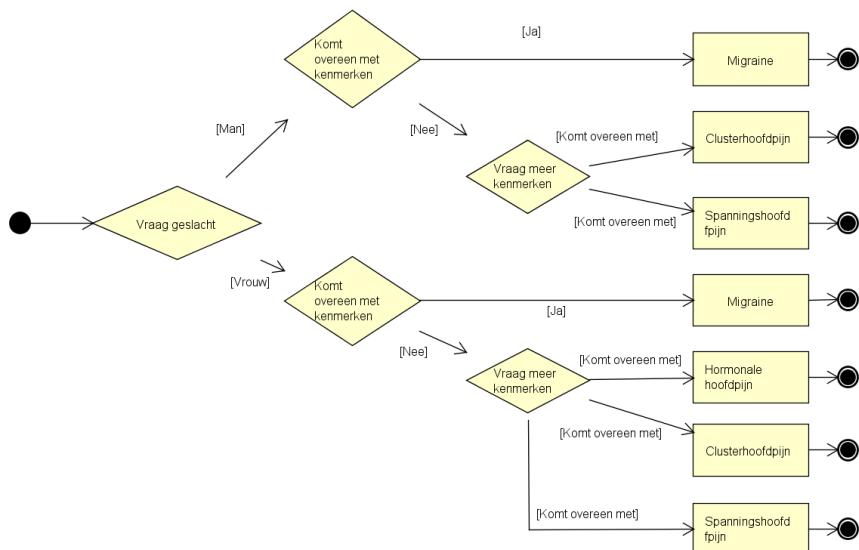
Zoals in de casus van dit onderzoek is beschreven moet duidelijk worden hoe de ernst van een klacht geclasseerd kan worden. Om te kunnen classificeren zijn algoritmes nodig.

In deze paragraaf worden twee decision algoritmes behandeld:

- Tree
- Graph

Hoe deze werken en wat de verschillen zijn wordt hier beschreven.

10.2.1. Tree



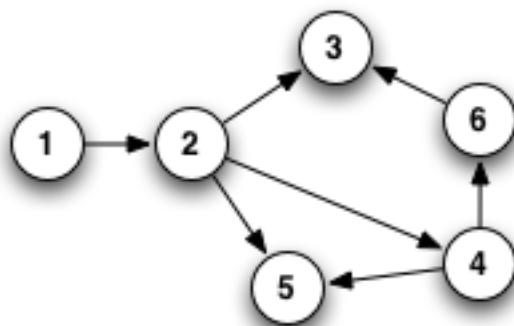
Figuur 53: Voorbeeld van een decision tree

In Figuur 53 staat een voorbeeld van een tree. Een tree is een algoritme dat een dataset classificeert. Elk onderwerp van een dataset wordt een root node (parent) genoemd. De uitkomsten hiervan zijn leaf nodes (children). De beslissing welk onderwerp gesplitst moet worden, is gebaseerd op resultaten die verkregen worden van de eerdere splitsingen. Figuur 53 geeft een voorbeeld van het classificeren van hoofdpijn.

Classificaties voor trees hebben vaak twee mogelijke uitkomsten. Maar meerdere uitkomsten zijn ook mogelijk. Daarnaast kan ook geklassificeerd worden door een variabele te gebruiken die groter of kleiner is dan een gegeven hoeveelheid. Bijvoorbeeld "Wat is uw gewicht?". Als het gewicht groter is dan bijvoorbeeld tachtig kilogram slaat deze een ander pad in dan wanneer het gewicht lager zou zijn.

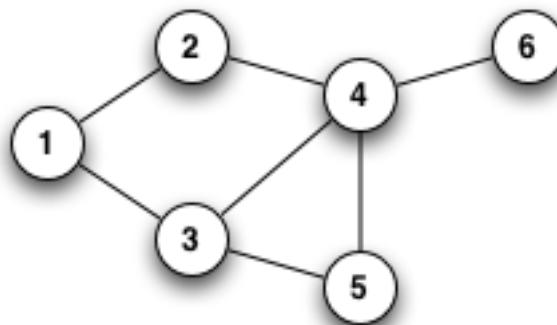
10.2.2. Graph

Een graph is een complexere versie van een tree. Zo is een tree een hiërarchisch model waar een graph een netwerkmodel is. Dit houdt in dat deze in twee richtingen op kan bewegen. Hierdoor kan het model bijvoorbeeld door zichzelf lopen of een bepaalde circuit volgen. Graphs zijn onder te verdelen in twee types: directed en undirected graphs. Bij een directed graph is het pad bepaald en kan deze alleen gevuld worden in één richting (zie Figuur 54). Het is hier wel mogelijk om terug te komen op een plek die al bezocht is. Denk hierbij aan straten met eenrichtingsverkeer.



Figuur 54: Directed graph

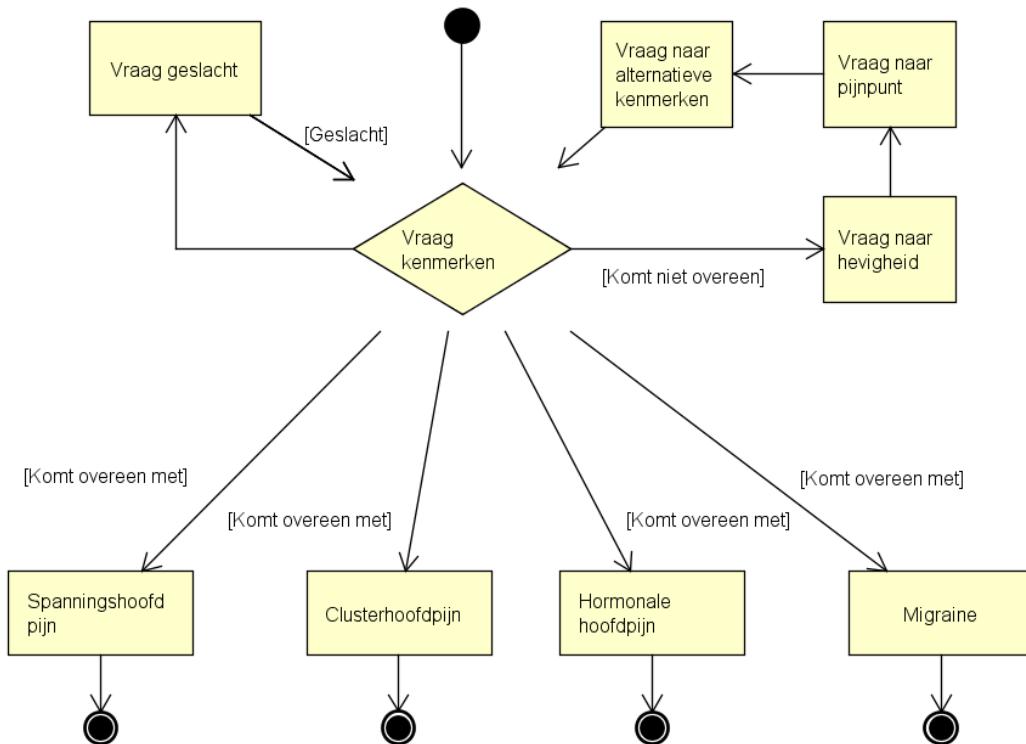
Een undirected graph kan in elke richting afgelegd worden. Denk hierbij aan kaart van een metro. (Zie Figuur 55)



Figuur 55: Undirected graph

Nog een verschil is dat geen child-parent relatie aanwezig is bij het graph based algoritme.

Een graph heeft minder punten nodig, omdat de punten herbruikbaar zijn. Ook zijn minder uiteinden nodig dan bij een tree. Dit brengt dat de graph minder groot hoeft te zijn voor hetzelfde resultaat. Echter is een graph complexer. In Figuur 56 is een voorbeeld te zien van een directed graph. De directed graph zal gebruikt worden voor het prototype, omdat de casus gezien de complexiteit meer voordeel heeft bij een graph. (Lambert, 2018)



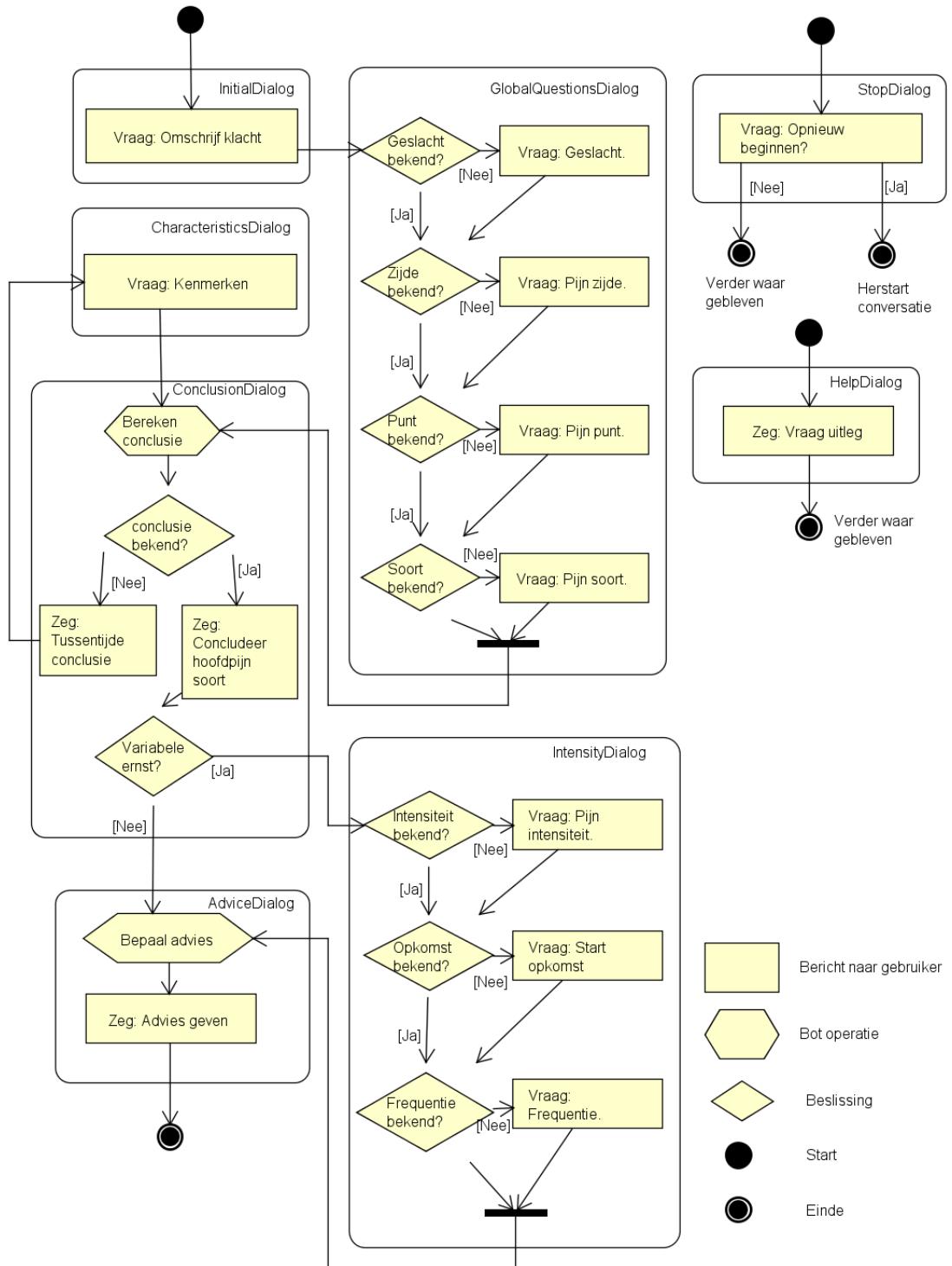
Figuur 56: Een voorbeeld van een graph

10.3. Implementatie

In deze paragraaf wordt uitgelegd hoe een graph is opgezet en geïmplementeerd in het prototype. Voor het implementeren is onderzocht hoe een conversatie gebouwd wordt met de onderdelen van het MSBF.

10.3.1. Flowchart

Om een graph te implementeren in de chatbot is een diagram ontwikkeld. Het diagram is in de vorm van een flowchart waarbij duidelijk te zien is wat gebeurt per dialoog. In Figuur 57 wordt het diagram getoond die gebruikt zal worden bij het implementeren van de dialogen van het prototype.



Figuur 57: Graph flowchart

Het diagram heeft hoofdpijn als onderwerp. Met deze flow kan de gebruiker achter de vorm van de hoofdpijn komen, door vragen te beantwoorden en kenmerken door te geven. Daarbij zal de chatbot achteraf nog advies geven over eventuele vervolgstappen die gezet kunnen worden.

10.3.2. Implementatie flowchart

Externe Tool

De enige gevonden tool is Bot graph dialog. Dit is een tool voor het MSBF om dialogen te implementeren (Turgman, 2017). Deze tool maakt het implementeren van een graph eenvoudiger. Dialogen worden opgeslagen in een JSON bestand in plaats van statisch in de code. Dit maakt het overzichtelijker en makkelijker om verschillende dialogen toe te voegen. Deze data kan weer dynamisch ingeladen worden. In Figuur 58 is een voorbeeld te zien van een dialoog in JSON formaat.

```
1  {
2    "id": "hoofdpijn",
3    "version": "1",
4    "type": "sequence",
5    "steps": [
6      {
7        "id": "pijnPunt",
8        "type": "prompt",
9        "data": {
10          "type": "choice",
11          "text": "Kunt u aangeven waar de pijn zich bevindt?",
12          "options": [ "Achter de ogen", "Links en rechts", "Voorhoofd" ],
13          "config": {
14            "listStyle": "button"
15          }
16        }
17      }
18    ]
19  }
```

Figuur 58: Eenvoudig dialoog in JSON formaat

Tijdens het gebruiken van deze tool is snel duidelijk geworden dat deze manier van dialogen schrijven problemen geeft. Het schrijven van een dialoog is makkelijk in dit formaat, maar daardoor ook simplistisch. Het is niet mogelijk om meerdere functies uit te voeren of extra logica aan de stappen toe te voegen. Hierdoor is het lastig om data te bewaren voor meerdere dialogen. Omdat deze tool simplicitet als vervanger van vrijheid biedt is besloten deze tool niet te gebruiken en met het MSBF zelf de graph te implementeren.

Onderdelen flowchart

De flowchart (Figuur 57) is onder te verdelen in een set onderdelen. Deze onderdelen zijn de dialogen, berichten, beslissingen en chatbot operaties. Per onderdeel wordt de gebruikte tool ook uitgelegd.

Dialogen

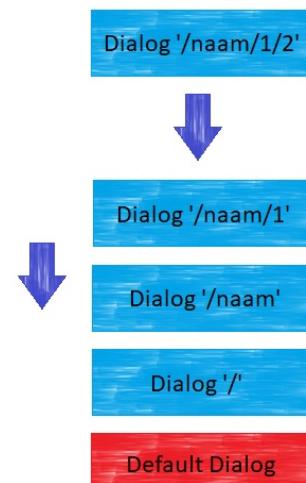
Het MSBF biedt de functionaliteit om dialogen te kunnen moduleren, deze functionaliteit heet 'dialogs'. Deze 'dialogs' krijgen een naam en een array met functies. De functies zijn de stappen die doorlopen worden in de 'dialog'. Het is mogelijk om binnen de functies te communiceren met andere onderdelen in de code of andere 'dialogs' te starten. Dialogs (Iqbal, Standefer, Mak, & Vo, Dialogs in the Bot Builder SDK for Node.js, 2017). De dialogs maken gebruik van de dialogs stack van een conversatie.

Dialog stack

Het bijhouden van 'dialogs' wordt met de zogenaamde 'dialog stack' gedaan. Wanneer een 'dialog' wordt gestart in een ander 'dialog' wordt deze nieuwe 'dialog' bovenop de stapel geplaatst en doorlopen. Zodra deze 'dialog' klaar is wordt deze verwijderd van de stapel en gaat het gesprek verder met de vorige 'dialog'. Wanneer de stapel geen 'dialogs' bevat wordt de 'default dialog' aangeroepen.

Dialog Implementatie

Aan de hand van de flowchart (Figuur 57) zijn de 'dialogs' nagebouwd. In Figuur 60 is te zien hoe de 'dialogs' geïnitialiseerd worden. Elke dialog wordt in een aparte file gezet en in een specifieke map. Alle dialogs in deze specifieke map worden dynamisch opgehaald en geïnitialiseerd.



Figuur 59: Dialog stack

```
fs.readdir('./Dialogs', (err, files) => {
  files.forEach(file => {
    let dialog = require('../Dialogs/' + file)(this.intents);
    this.bot.dialog(dialog.name, dialog.steps);
  });
})
```

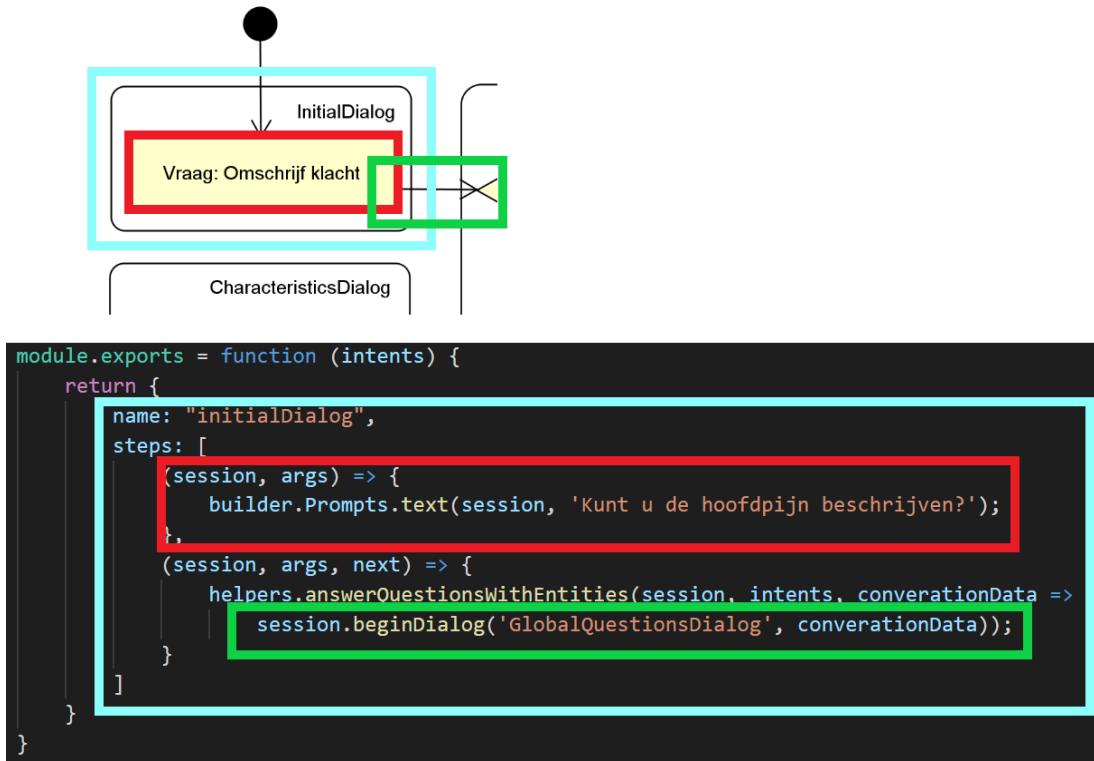
Figuur 60: Dialog initialisatie

In Figuur 61 is te zien hoe de 'initialDialog' uit de flowchart geïmplementeerd is. De naam en de stappen worden geëxporteerd. Iedere stap heeft de 'session' parameter, deze parameter wordt gebruikt om bijvoorbeeld berichten te sturen en 'dialogs' te beginnen. Daarnaast zijn nog twee optionele parameters beschikbaar. De tweede is de 'args' parameter, hier staat de eventuele informatie van de vorige stap in. De laatste parameter is 'next'. Dit is een functie die direct naar de volgende stap gaat wanneer deze aangeroepen wordt.

```
module.exports = function (intents) {
  return {
    name: "initialDialog",
    steps: [
      (session, args) => {
        builder.Prompts.text(session, 'Kunt u de hoofdpijn beschrijven?');
      },
      (session, args, next) => {
        helpers.answerQuestionsWithEntities(session, intents, conversationData =>
          session.beginDialog('GlobalQuestionsDialog', conversationData));
      }
    ]
  }
}
```

Figuur 61: initialDialog implementatie

In Figuur 62 is de vergelijking te zien tussen een 'dialog' in de flowchart en de daadwerkelijk geïmplementeerde dialog.



Figuur 62: Code vergelijking met de chart

Dialog triggers

Zoals in figuur P te zien is zijn twee losse 'dialogs' aanwezig. Deze worden niet aangeroepen in de normale flow, maar wanneer de gebruiker 'stop' of 'help' zegt. Deze dialogs zijn geïmplementeerd met MSBF's Actions (Iqbal, Standefer, & Mak, Handle user actions, 2017). Met Actions worden 'dialogs' gestart, herstart of gestopt. Binnen het prototype worden deze Actions gebruikt om op basis van een regex de 'stopDialog' of 'helpDialog' te starten, zie Figuur 63. In deze implementatie wordt gebruik gemaakt van de 'beginDialogAction'. Zoals de naam weggeeft start deze Action een 'dialog' wanneer een bericht van de gebruiker voldoet aan de meegegeven regex.

```

this.bot.beginDialogAction('StopAction', 'StopDialog', { matches: /^stop$/i });
this.bot.beginDialogAction('HelpAction', 'HelpDialog', { matches: /^help$/i });

```

Figuur 63: Actions om de losse dialogen te starten.

Berichten versturen en vragen stellen

In de flowchart zijn de berichten onderverdeeld in vragen en berichten zoals aangegeven met 'vraag' en 'zeg'. De berichten zijn te implementeren met het MSBF. Hiervoor wordt de 'session.send' functie voor gebruikt (Figuur 64). Met deze functie wordt een bericht verstuurd waarna de chatbot direct doorgaat naar de volgende stap.

```

session.send('Ik zal een aantal vragen stellen om een goede conclusie te kunnen geven. ' +
  'kunt gebruik maken van de knoppen.');

```

Figuur 64: Eerste bericht in de GlobalQuestionsDialog.

Wanneer een vraag gesteld wordt, moet gewacht worden op het antwoord. Daarom worden hier 'Prompts' voor gebruikt. (Vo, Iqbal, Standefer, Mak, & Delimarsky, Prompt for user input, 2017). Met Prompts kan een vraag worden gesteld die een specifiek soort antwoord verwacht. Wanneer het antwoord niet klopt wordt opnieuw gevraagd om correct antwoord te geven. Wanneer een correct antwoord is gegeven zal de chatbot pas naar de volgende stap gaan. In deze stap is het gegeven antwoord te vinden in de 'args' parameter. Een Prompt kan bijvoorbeeld een ja of nee verwachten, een getal of een datum. Voor iedere vraag van de flowchart is een Prompt gebruikt, zie Figuur 65, Figuur 66 en Figuur 67

```
builder.Prompts.choice(session, "Wat is uw geslacht?", "Man|Vrouw", { listStyle: builder.ListStyle.button });
```

Figuur 65: Choice Prompt

```
builder.Prompts.time(session, "Wanneer is de hoofdpijn begonnen met opkomen?");
```

Figuur 66: Time Prompt

```
builder.Prompts.confirm(session, "Wilt u opnieuw beginnen?", { listStyle: builder.ListStyle.button });
```

Figuur 67: Confirm Prompt

De Choice Prompt, zie Figuur 65 vraagt om één van de opgegeven keuzes in de derde parameter. Ook is als vierde parameter meegegeven dat het bericht knoppen moet krijgen voor makkelijk gebruik. De Time Prompt, zie Figuur 66, verwacht een datum of tijd. De Confirm Prompt, zie Figuur 67, lijkt op de Choice Prompt maar verwacht alleen een ja of een nee.

Beslissingen

Voor de implementatie van de beslissingen wordt niet een specifieke functionaliteit van het MSBF gebruikt. Hiervoor worden if-statements gebruikt. In Figuur 68 is te zien dat op basis van informatie opgehaald uit de helpers een beslissing wordt gemaakt. De helpers worden in de paragraaf 'Bot operaties' uitgelegd.

```
if (!helpers.questionAnswered('Geslacht', cData)) {
    builder.Prompts.choice(session, "Wat is uw geslacht?", "Man|Vrouw", { listStyle: builder.ListStyle.button });
} else {
    next();
}
```

Figuur 68: Beslissing in de GlobalQuestionsDialog

Bot operaties

Bot operaties zijn de functies die de chatbot uitvoert. Deze operaties verwerken de informatie van de gebruiker. De operaties zijn los van de dialogs geïmplementeerd. Hierdoor kan iedere dialog gebruik maken van deze operaties wanneer nodig. Ook wordt de communicatie met de NLP tool met chatbot operatie gedaan. Zo haalt een bot operatie de entites van de NLP op, die onder andere aangeven of de gebruiker een man of vrouw is.

```

let getConclusion = cData => {
  possibilities = [];
  let possibleConclusions = conclusions[cData.characteristics.type].types;

  for (let c = 0; c < Object.keys(possibleConclusions).length -1; c++) {
    possibilities[c] = {name: Object.keys(possibleConclusions)[c], score: 0}
    for (let s = 0; s < Object.keys(cData.characteristics).length; s++) {
      let sKey = Object.keys(cData.characteristics)[s];
      if (sKey != 'type') {
        if (cData.characteristics[sKey].constructor === Array) {
          let characteristicsArray = cData.characteristics[sKey];
          for (let a = 0; a < characteristicsArray.length; a++) {
            if (possibleConclusions[possibilities[c].name][sKey].has(characteristicsArray[a])) {
              possibilities[c].score += 1;
            }
          }
        } else {
          if (cData.characteristics[sKey] != null
            && possibleConclusions[possibilities[c].name][sKey].has(cData.characteristics[sKey])) {
            possibilities[c].score += 1;
          }
        }
      }
    }
  }

  possibilities.sort((a,b) => (a.score < b.score) ? 1 : ((b.score > a.score) ? -1 : 0) );
}

data = {
  possibilities: possibilities,
  final: possibilities[0].score - possibilities[1].score > 2,
  variableIntensity: hasVariableIntensity(cData.characteristics.type, possibilities[0].name)
}

return data;
}

```

Figuur 69: Bot operatie 'getConclusion'

```

let conclusion = helpers.getConclusion(cData);

```

Figuur 70: Gebruik van 'getConclusion'

De functie in Figuur 69 loopt door de kenmerken van 'cData' heen en zoekt voor overeenkomsten van de kenmerken van verschillende soorten hoofdpijn. Wanneer een kenmerk overeenkomt krijgt die soort hoofdpijn een punt. Vervolgens wordt de conclusie gegeven op basis van het aantal punten. De 'cData' variabele bevat de kenmerken die de gebruiker aan heeft gegeven. De kenmerken van de verschillende soorten hoofdpijn staan in een variabele genaamd 'conclusions'. Deze variabele is een JSON bestand met de kenmerken.

```

"Spanningshoofdpijn": {
    "Geslacht": [
        "Man",
        "Vrouw"
    ],
    "PijnZijde": [
        "Beide"
    ],
    "PijnPunt": [
        "Slaap",
        "Voorhoofd"
    ],
    "PijnSoort": [
        "Drukkend",
        "Dof"
    ],
    "Hevigheid": [
        1,
        2
    ],
    "Aanwezigheid": [
        "Wisselend"
    ],
    "Kenmerken": [
        "Pijnlijke, stijve en gespannen nek",
        "Pijnlijke, stijve en gespannen schouders",
        "Misselijkheid bij toenemende hoofdpijn (niet overgeven)",
        "Depressiviteit als gevolg"
    ],
    "Advies": {
        "Matig": "Ontspannen( Goed slapen, regelmatig strekoefeningen, massage), Paracetamol, Ibuprofen"
    }
},

```

Figuur 71: Kenmerken van soorten hoofdpijn in JSON

In Figuur 71 is te zien hoe de kenmerken van een soort hoofdpijn, in dit geval spanningshoofdpijn, genoteerd zijn. Op deze manier is het makkelijk om eventuele andere soorten hoofdpijn toe te voegen zonder code aan te passen.

10.4. Conclusie

Na een vergelijking tussen de tree en graph is gekozen om een graph te implementeren. De graph past binnen de casus van dit onderzoek, omdat het mogelijk is om terug te gaan naar een eerder punt. Hierdoor is het mogelijk om een complexere chatbot te maken. Dit is voor dit onderzoek essentieel omdat hierdoor nog aanpassingen in de vragen gemaakt kunnen worden gedurende het gesprek en tussen dialogen gewisseld kan worden indien nodig.

Door gebruik te maken van de tools die zijn meegeleverd bij het MSBF is de flowchart (Figuur 57) geïmplementeerd in de bot. Door middel van de 'dialog' functionaliteit van het MSBF zijn verschillende dialogs gemaakt die ieder hun eigen vragen stellen. Door het gebruik van Prompts kan op een eenvoudige manier een vraag gesteld worden aan de gebruiker. Wanneer de klant hier een antwoord op geeft zal de chatbot het antwoord opslaan en gebruiken om later in de conversatie een conclusie te berekenen. Met deze conclusie wordt uiteindelijk gericht advies gegeven aan de gebruiker.

In de initial dialog wordt gebruik gemaakt van een NLP tool om de entites uit de gegeven zin te halen. Deze entites bestaan uit:

- Informatie van de gebruiker (geslacht, etcetera).
- symptomen van de klacht.

De chatbot hoeft vervolgens niet meer naar deze symptomen te vragen.

11. Meetup bij KLM digital studio

Op 17 mei is het projectteam langs geweest bij de Dutch chatbot user group. Dit is een community van chatbot enthousiasten die om de paar maanden met elkaar afspreken om kennis met elkaar te delen. Dit keer was de meetup op schiphol bij KLM digital studio.

Hieronder een link naar de meetup pagina:

<https://www.meetup.com/DCUGNL/events/248331592>

Hier zijn twee sprekers langs gekomen: Dhr. Sarkar van KLM Chatbot deep dive en Dhr. Baldwin als motivational speaker. Dhr. Sarkar werkt als lead developer aan de BlueBot van KLM. Door middel van deze chatbot kan een gebruiker via een messenger platform een vliegticket boeken. Binnen deze chatbot wordt gebruik gemaakt van de NLP tool Dialogflow.

11.1. Verwachting meetup

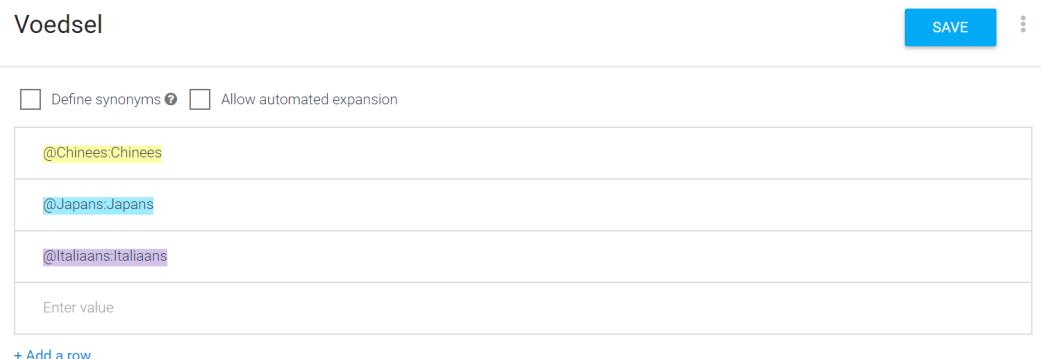
De verwachting die het projectteam had van de meetup bij KLM digital studio was voornamelijk het verkrijgen van informatie over hoe Dialogflow gebruikt wordt binnen KLM en inzicht krijgen over de opbouw van de BlueBot, ook wel BB genoemd.

11.2. Verkregen informatie

De volgende informatie heeft het projectteam verkregen bij de presentatie van Dhr. Sarkar.

11.2.1. Groeperen van entiteiten

Tijdens de presentatie van Dhr. Sarkar werd getoond hoe een groep van entiteiten gekoppeld kan worden aan overkoepelende entiteiten. Als voorbeeld zijn de entiteiten Chinees, Japans en Italiaans aangemaakt die als referentie Loempia, Sushi en Pizza mee kregen. Deze entiteiten kunnen weer gekoppeld worden aan een hoofd entiteit, in Figuur 72 is de hoofd entiteit "voedsel".



Voedsel

SAVE

Define synonyms ? Allow automated expansion

@Chinees:Chinees
@Japans:Japans
@Italiaans:Italiaans
Enter value

+ Add a row

Figuur 72: Entity voedsel

Door trainingsdata aan te maken met als voorbeeld "Mag ik pizza". Kan de entiteit pizza gekoppeld worden aan de intentie Voedsel. Zodra pizza genoemd wordt in een zin wordt als entiteit Voedsel meegeven die de waarde {"Italiaans":"Pizza"} meekrijgt (Zie Figuur 73).

Op deze manier kan bij de trainingsdata elke keer dezelfde entiteit meegegeven worden in plaats van elke intentie van hetzelfde type te linken aan een andere entiteit.

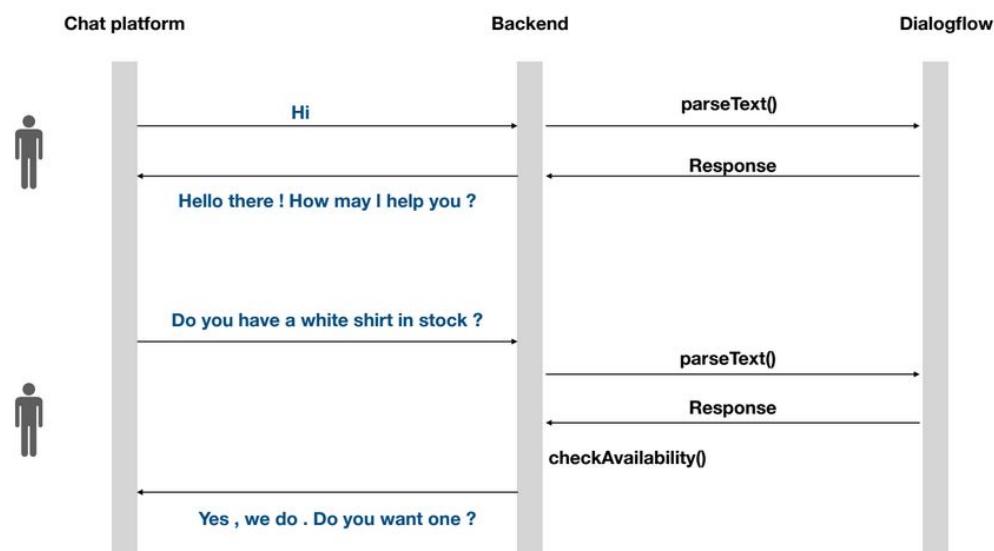
USER SAYS	COPY CURL
Wie heeft er pizza	
 DEFAULT RESPONSE	▼
oke	
INTENT	
Voedsel	
ACTION	
<i>Not available</i>	
PARAMETER	VALUE
Voedsel	{"Italiaans": "Pizza"}
date-period	

Figuur 73: Zin output

11.2.2. Opbouw chatbot

Tijdens de presentatie gaf Dhr. Sarkur aan dat KLM twee opties heeft overwogen voor het implementeren van Dialogflow in BB.

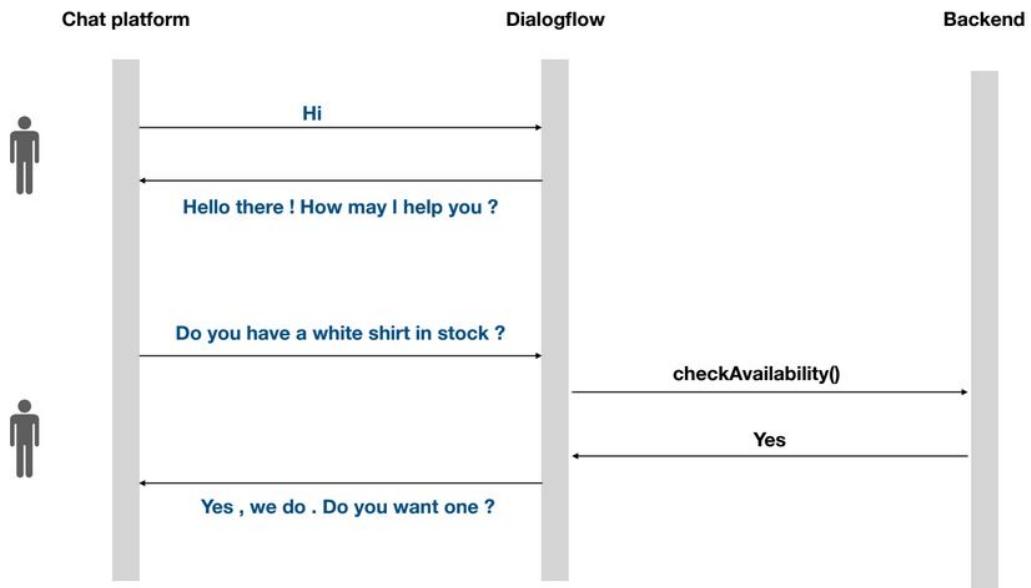
De eerste keuze was om alle berichten te ontvangen op de eigen backend. Hierdoor hebben de developers volledige controle over wat en wanneer informatie naar Dialogflow wordt gestuurd. Op dit moment worden de persoonlijke gegevens weg gefilterd voordat het bericht wordt doorgestuurd naar Dialogflow. In enkele gevallen wordt de inhoud überhaupt niet naar Dialogflow verstuurd. Dit in verband met de General Data Protection Regulation. Zoals in Figuur 74 is de huidige flow van de BB te zien.



Figuur 74: Huidige opzet BB

De andere optie was om alle berichten direct naar Dialogflow te sturen en in enkele gevallen het resultaat naar de eigen backend te sturen. Hierdoor kon KLM echter de

privacy van de klant niet meer waarborgen. Daarnaast had KLM minder controle over wat met alle data gebeurt. Een diagram van deze opzet is zichtbaar in Figuur 75.



Figuur 75: Alternatief waarbij KLM geen controle had

Gezien KLM zelf controle wilt over de data en zich moet kunnen verantwoorden met betrekking tot de GDPR is gekozen voor de eerste optie.

11.2.3. Context

Omdat KLM volledige controle heeft over hoe en waar Dialogflow wordt ingezet bepaalt KLM ook zelf de context. Dit gebeurt door API calls te sturen naar Dialogflow. Om te bepalen waar in de flow de gebruiker zich bevindt maakt KLM gebruik van wat Dialogflow output context noemt. De output context kan per intentie ingesteld worden, daarnaast kunnen ook meerdere output contexts toegevoegd worden. Nu KLM weet waar in de flow de gebruiker zich bevindt is een term als "Ja" niet ambigu meer, omdat KLM kan herleiden wat aan de gebruiker is gevraagd.

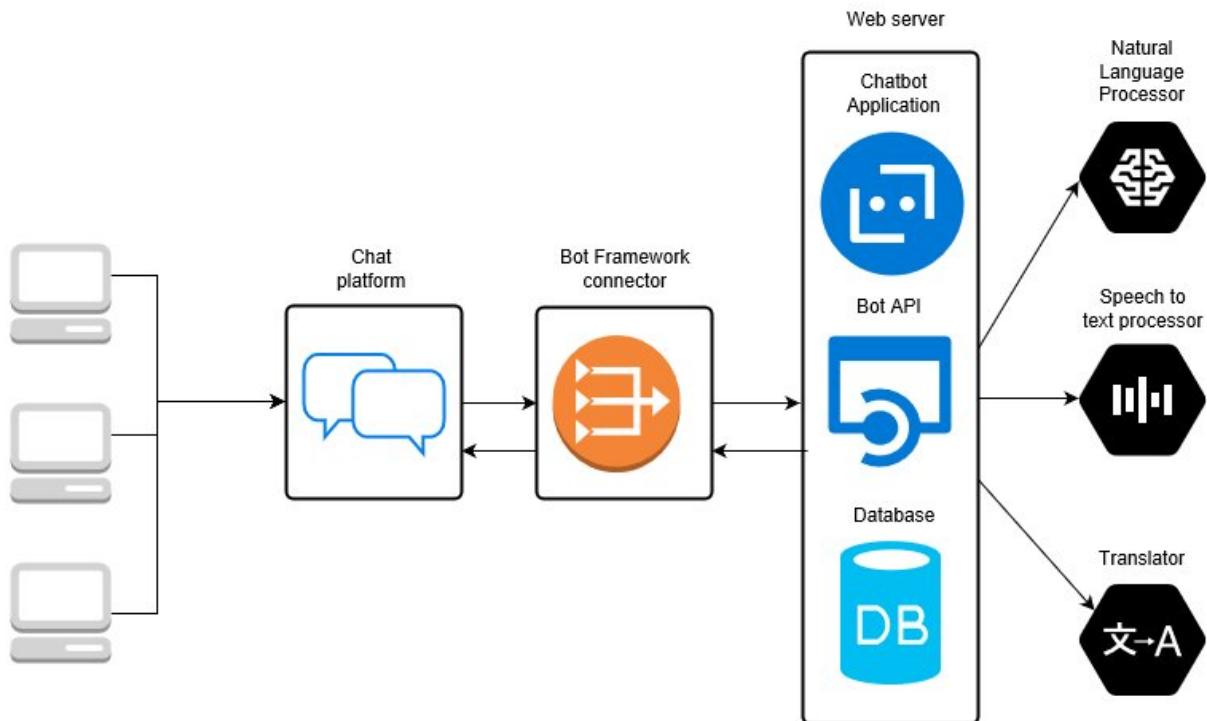
11.3. Conclusie

De meetup gaf nieuwe inzichten en toonde voor het projectteam nieuwe functionaliteiten van Dialogflow. Het nadeel is dat de meetup erg laat in de projecttijd viel. Voor de meeting is een andere opbouw bepaald en gebruikt waardoor de informatie irrelevant is geworden.

12. Conclusie

In dit onderzoek is gezocht naar een antwoord op de hoofdvraag: "Hoe kan een chatbot de ernst van een medische klacht classificeren?"

Uit onderzoek is gebleken dat een chatbot als volgt is opgebouwd, zie Figuur 76:



Figuur 76: Opbouw van een chatbot

De relevante onderdelen voor dit onderzoek zijn de Chatbot Application en de Natural Language Processing (NLP). Na onderzoek is voor de Chatbot Application het Microsoft Bot Framework gekozen en voor NLP, Dialogflow.

Met de gekozen tools is het prototype gebouwd. In dit prototype is het classificeren van een medische klacht succesvol geïmplementeerd. Dit is mogelijk gemaakt met een combinatie van twee conversatie algoritmes. Dit zijn de flow- en intent based algoritmes. Om de flow van de conversatie te realiseren is gebruik gemaakt van een decision graph. Dankzij het gebruik van de bovenstaande onderdelen is het classificeren van een medische klacht door een chatbot mogelijk gemaakt. Met het geslaagde prototype is de hoofdvraag beantwoord. Om het prototype te bekijken zie Bijlage B.

13. Adviesrapport

In dit hoofdstuk wordt een advies gegeven over hoe een chatbot gerealiseerd kan worden binnen de casus van de gebruiker.

13.1. Bepaal het doel van de chatbot

Chatbots kunnen veel toevoegingen bieden, maar hierbij moet overwogen worden of de chatbot voldoende aanvulling voor het doeleind biedt. Dit tegenover de moeite die het de klant kost om een conversatie te voeren met de chatbot. Initieel moet bepaald worden wat het resultaat zal zijn van een succesvolle conversatie met de chatbot.

Voor een gebruiker kost het aanspreken van een chatbot moeite en tijd. Wanneer het resultaat niet dusdanig voldoet aan de verwachtingen zullen gebruikers niet de tijd nemen om hier een chatbot voor aan te spreken, of kan de chatbot een negatieve invloed hebben.

Bepaal met deze informatie of een chatbot een toegevoegde waarde heeft bij het behalen van het doel.

13.2. Maak de nodige onderdelen duidelijk

Nadat het doel van de chatbot is geformuleerd is het belangrijk dat de benodigde onderdelen van de chatbot worden bepaald. Om de juiste onderdelen van een chatbot te bepalen is het verstandig om eisen op te stellen per onderdeel.

Op het moment dat eisen bekend zijn moet duidelijk zijn hoe de chatbot benaderd kan worden. Een chatbot kan benaderbaar zijn via tekst en spraak. Aan de hand van deze keuze moet gekeken worden naar welke services gebruikt moeten worden. Deze services helpen de chatbot met het begrijpen van de gebruiker.

Wanneer gebruik wordt gemaakt van tekst input van een gebruiker kan de chatbot kijken of de zin overeenkomt met een zin die de chatbot kent. Echter is de gebruiker dan gelimiteerd aan specifieke teksten. Een service die dit probleem kan oplossen is een Natural Language Processor.

Bij het gebruik van spraak, moet de opgenomen audio worden vertaald naar tekst voordat de chatbot hier wat van kan begrijpen. Hiervoor moet een speech-to-text service gebruikt worden.

Als de chatbot internationaal beschikbaar moet zijn moet de chatbot uitgerust zijn met een vertalende functionaliteit. Hier zijn ook services voor beschikbaar.

Een groot aantal services zijn beschikbaar om de chatbot aan te koppelen. Aan de hand van de eisen moet nagegaan worden welke services relevant zijn. De onderzochte services zijn ieder voor zich een unieke oplossing voor een probleem, echter willen niet alle services samenwerken met een mogelijk gekozen framework. De aanbevolen services zijn:

- Wit.Ai van Facebook
- Azure Bot Services van Microsoft
- Dialogflow van Google

Bij het gebruik van het Microsoft Bot Framework is te adviseren om Dialogflow te gebruiken als Natural Language Processor, omdat het Microsoft Bot Framework en Dialogflow gebruik maken van dezelfde intent en entity structuur. Bij het gebruik van het

Hubot framework kunnen de Natural Language Processors DialogFlow en Wit.Ai gebruikt worden. Hubot vraagt en geeft weinig tot geen structuur en is daarom goed te combineren.

Na het bepalen van de chatbot onderdelen zal moeten worden gekeken waar de chatbot benaderbaar is. Door te kiezen voor een bestaand platform, zoals bijvoorbeeld:

- Facebook Messenger
- Slack
- Skype

Deze chat platforms zijn van derde partijen waar de chatbot aan gekoppeld kan worden. Bij het gebruik van externe platforms zal de gebruiker dit platform moeten downloaden, installeren of naar toe browsen. Naast platforms van derde partijen is het ook mogelijk om een eigen platform te realiseren. Per casus zou gekeken moeten worden wat de beste chatbot platform is.

Een chatbot kan als vriendelijker en menselijker worden ervaren wanneer de chatbot onthoudt wat op voorgaande vragen is beantwoord. Deze data kan in-memory worden opgeslagen. Als het van belang is dat de chatbot de gebruiker herkent wanneer de gebruiker terugkeert, is het belangrijk dat de data extern wordt opgeslagen. Hiervoor kan bijvoorbeeld een externe database gebruikt worden. Een andere mogelijkheid is om de data per gebruiker in een bestand op te slaan binnen de codebase.

13.3. Kies de conversatie methode

Een chatbot zal moeten omgaan met input van gebruikers en hierop kunnen reageren. Een structuur hebben in de conversatie is een van de belangrijkste onderdelen van de chatbot. Wanneer deze structuur ontbreekt of onduidelijk is, is de kans groot dat de chatbot niet op de juiste manier zal reageren. Deze structuur wordt over het algemeen visueel gemaakt in een flowchart, dit is een overzicht van alle mogelijke vragen, antwoorden en berichten die de chatbot verstuurde.

Voordat deze flowchart opgezet wordt, moet duidelijk zijn wat het eindpunt is van een conversatie met de chatbot. Bijvoorbeeld: Een chatbot die een reservering plaatst bij een restaurant zal als eindpunt hebben dat alle benodigde informatie is verzameld om de reservering te maken. Om te achterhalen welke methode het beste past bij de chatbot, moet één succesflow bepaald worden, de reeks van berichten die leiden tot het eindpunt. Vanuit hier wordt bepaald hoeveel vertakkingen en mogelijke uitgangspunten nodig zijn.

Is het mogelijk vanaf meerdere punten te beginnen? Maak dan gebruik van de intent based methode. Bij één begin- en eindpunt, maak gebruik van de waterval methode. Wanneer de flow bestaat uit één beginpunt met meerdere eindpunten is het gebruikt van de flow based methode aan te raden.

Wanneer de chatbot een beslissing maakt, moet duidelijk worden welk pad ingeslagen moet worden. Op het moment dat een flow te complex of onoverzichtelijk wordt zal de standaard flow based methode vervangen moeten worden door een complexere methode, zoals de combinatie van de flow based en de intent based methode. Hiervoor kunnen de volgende algoritmes worden gebruikt:

- Decision-graph
- Decision-tree.

Het gehele verhaal wordt volledig duidelijk gemaakt in een flowchart. Wanneer deze flowchart gemaakt is kan de implementatie beginnen.

Bibliografie

- Akkulka. (2017, april). *api-ai-recognizer*. Opgeroepen op april 17, 2018, van NPM: <https://www.npmjs.com/package/api-ai-recognizer>
- Alok, R. P., & Diganta, S. (2015, Juli). *Word sense disambiguation: A survey*. Opgeroepen op Mei 7, 2018, van arxiv: <https://arxiv.org/ftp/arxiv/papers/1508/1508.01346.pdf>
- Aspect. (Z.D.). *Aspect CXP Pro*. Opgeroepen op Maart 13, 2018, van Aspect: <https://www.aspect.com/solutions/self-service/aspect-cxp-pro>
- AVRO Tros. (2017, mei 2). *Huisartsen willen minder patiënten: wachtkamers overvol*. Opgehaald van Zorg Nu AVRO Tros: <https://zorgnu.avrotros.nl/nieuws/detail/huisartsen-willen-minder-patiënten-wachtkamers-overvol/>
- Botkit. (Z.D.). *Building Blocks for Building Bots*. Opgeroepen op Maart 13, 2018, van Botkit: <https://botkit.ai>
- Botmaster. (Z.D.). *Botmaster Documentation*. Opgeroepen op Maart 13, 2018, van Botmaster: <http://botmasterai.com/documentation/latest/>
- Botpress. (2017, juni 13). *What is a bot - on the inside?* Opgehaald van Botpress: <https://botpress.io/blog/what-is-a-bot-on-the-inside/>
- Botpress. (Z.D.). *The ultimate bot framework*. Opgeroepen op Maart 13, 2018, van Botpress: <https://botpress.io>
- Buskoop, D. (2017, mei 2). *Wachtkamer van de huisarts zit te vol*. Opgehaald van BNR: <https://www.bnr.nl/nieuws/zorg/10322060/wachtkamer-van-de-huisarts-zit-te-vol>
- Chowdhury, G. G. (2005, Januari 31). *Language and Representation*. Opgehaald van Wiley Online Library: <https://onlinelibrary.wiley.com/doi/full/10.1002/aris.1440370103>
- Chowdhury, G. G. (2005, januari 31). *Natural language processing*. Opgehaald van Wiley Online Library: <https://onlinelibrary.wiley.com/doi/full/10.1002/aris.1440370103>
- Dorash, M. (2017, December 19). *Machine Learning vs. Rule Based Systems in NLP*. Opgeroepen op Mei 8, 2018, van friendlydata.io: <https://friendlydata.io/blog/machine-learning-vs-rule-based-systems>
- Expllosion AI. (Z.D.). *Industrial-Strength Natural Language Processing*. Opgeroepen op Maart 28, 2018, van SpaCy: <https://spacy.io/>
- Etsel. (2015, juli 9). *Taal: stopzinnen/stopwoorden - zal ik maar zeggen*. Opgeroepen op april 25, 2018, van Educatie en School: Taal: stopzinnen/stopwoorden - zal ik maar zeggen
- Facebook. (Z.D.). *Natural Language for Developers*. Opgeroepen op Maart 28, 2018, van Wit ai: <https://wi.ao>
- Flow Xo. (Z.D.). *Smart chatbots made simple*. Opgeroepen op Maart 13, 2018, van Flow Xo: <https://flowxo.com/>
- Freed, A. R. (2017, Januari 25). *Comparing rules and machine learning natural language processing approaches*. Opgeroepen op Mei 8, 2018, van Freedville Blog: <http://freedville.com/blog/2017/01/25/comparing-rules-and-machine-learning-natural-language-processing-approaches/>

- Freed, A. R. (2017, Januari 13). *Demo of natural language processing with rules and machine-learning based approaches*. Opgeroepen op mei 8, 2018, van freedville: <https://freedville.com/blog/2017/01/13/demo-of-natural-language-processing-with-rules-and-machine-learning-based-approaches/>
- Freed, A. R. (2017, Januari 20). *Improving simple natural language processing models with rules or machine learning*. Opgeroepen op Mei 8, 2018, van Freedville Blog: Improving simple natural language processing models with rules or machine learning
- Gazarov, P. (2016, Augustus 13). *What is an API? In English, please*. Opgehaald van freeCodeCamp: <https://medium.freecodecamp.org/what-is-an-api-in-english-please-b880a3214a82>
- Geeks for Geeks. (sd). *Removing stop words with NLTK in Python*. Opgeroepen op mei 28, 2018, van Geeks for Geeks: <https://www.geeksforgeeks.org/removing-stop-words-nltk-python/>
- Github Inc. (Z.D.). *Learn Git and GitHub without any code!* Opgeroepen op April 9, 2018, van Github: <https://github.com>
- Github. (Z.D.). *Hubot*. Opgeroepen op Maart 13, 2018, van Hubot: <https://hubot.github.com>
- Google. (2018). *CLOUD SPEECH API*. Opgehaald van Google cloud: <https://cloud.google.com/speech/>
- Google. (Z.D.). *Build natural and rich conversational experiences*. Opgeroepen op April 3, 2018, van Dialogflow.
- Gunthercox. (2018, Februari 27). *ChatterBot*. Opgeroepen op Maart 13, 2018, van Github: <https://github.com/gunthercox/ChatterBot>
- Haeseryn, W. K. (1997). *Part-of-speech tagging*. Opgeroepen op mei 14, 2018, van ru: http://lands.let.ru.nl/cgn/doc_Dutch/topics/version_1.0/annot/pos_tagging/info.htm#de
- HBO-i. (2018, januari 18). *Methoden Toolkit HBO-i*. Opgehaald van HBO-i Onderzoek: http://onderzoek.hbo-i.nl/index.php/Methoden_Toolkit_HBO-i
- Hofmann, T., & Ganea, O.-E. (2017, Juli 31). *Deep Joint Entity Disambiguation with Local Neural Attention*. Opgeroepen op April 25, 2018, van Arxiv: <https://arxiv.org/pdf/1704.04920.pdf>
- Hongzhao, H., Larry, H., & Heng, J. (2015, April 28). *Leveraging Deep Neural Networks and Knowledge Graphs for Entity Disambiguation*. Opgeroepen op APril 25, 2018, van Arxiv: <https://arxiv.org/pdf/1504.07678.pdf>
- IBM. (2013, Januari 23). *The Art of Tokenization*. Opgehaald van <https://www.ibm.com/developerworks/community/blogs/nlp/entry/tokenization?lang=en>
- Ina. (2017, maart 21). *How do Chatbots Work?* Opgehaald van Onlim: <https://onlim.com/en/how-do-chatbots-work/>
- infoNu. (2015, Juli 9). *Taal: stopzinnen/stopwoorden - zal ik maar zeggen*. Opgeroepen op April 25, 2018, van infoNu.nl: <https://educatie-en-school.infonu.nl/taal/115529-taal-stopzinnenstopwoorden-zal-ik-maar-zeggen.html>

- Iqbal, K., Standefer, R., & Mak, D. (2017, december 13). *Handle user actions*. Opgeroepen op april 24, 2018, van Microsoft: <https://docs.microsoft.com/en-us/azure/bot-service/nodejs/bot-builder-nodejs-dialog-actions?view=azure-bot-service-3.0>
- Iqbal, K., Standefer, R., Mak, D., & Vo, D. C. (2017, december 13). *Dialogs in the Bot Builder SDK for Node.js*. Opgeroepen op april 24, 2018, van Microsoft: <https://docs.microsoft.com/en-us/azure/bot-service/nodejs/bot-builder-nodejs-dialog-overview?view=azure-bot-service-3.0>
- James Campbell. (Z.D.). *The World's Simplest Bot Framework*. Opgeroepen op Maart 13, 2018, van Bottr: <https://bottr.co>
- Jashkenas. (Z.D.). *CoffeeScript*. Opgeroepen op Maart 28, 2018, van CoffeeScript: <http://coffeescript.org/>
- Jivani, A. G. (2011, December). *A Comparative Study of Stemming Algorithms*. Opgeroepen op mei 14, 2018, van kenbenoit.net: http://kenbenoit.net/assets/courses/tcd2014qta/readings/Jivani_ijcta2011020632.pdf
- Joffreybvn. (2017, juli). *botbuilder-wit-remade*. Opgeroepen op april 17, 2018, van NPM: <https://www.npmjs.com/package/botbuilder-wit-remade>
- Jurafsky, D., & Martin, J. H. (2018, augustus 7). *Part-of-Speech Tagging*. Opgeroepen op mei 14, 2018, van Stanford: <https://web.stanford.edu/~jurafsky/slp3/10.pdf>
- KDNuggets. (2015, December). *Everything You Need to Know about Natural Language Processing*. Opgeroepen op Mei 15, 2018, van KDNuggets: <https://www.kdnuggets.com/2015/12/natural-language-processing-101.html>
- Kiser, M. (2016, augustus 11). *Introduction to Natural Language Processing (NLP)*. Opgeroepen op Mei 15, 2018, van Algorithmia: <https://blog.algorithmia.com/introduction-natural-language-processing-nlp/>
- Kolalapudi, S. (2016, November 14). *Pluralsight*. Opgeroepen op April 25, 2018, van Pluralsight: <https://app.pluralsight.com/player?course=python-natural-language-processing&author=swetha-kolalapudi&name=python-natural-language-processing-m0&clip=0&mode=live>
- Kompella, R. (2018, februari 9). *Conversational AI chat-bot—Architecture overview*. Opgehaald van Towards Dara Science: <https://towardsdatascience.com/architecture-overview-of-a-conversational-ai-chat-bot-4ef3dfef52e>
- Kraaij, W., & Pohlmann, R. (sd). *Porter's stemming algorithm for Dutch*. Opgeroepen op mei 28, 2018, van Radboud Universiteit: <http://www.cs.ru.nl/~kraaijw/pubs/Biblio/papers/kraaij94porters.pdf>
- Lambert, M. (2018, april 21). *Chatbot Decision Trees*. Opgeroepen op april 25, 2018, van Medium: <https://chatbotsmagazine.com/chatbot-decision-trees-a42ed8b8cf32>
- Liddy, E. D. (2001). *Natural Language Processing*. Opgehaald van Surface: https://surface.syr.edu/cgi/viewcontent.cgi?referer=https%253A%252F%252Fscholar.google.nl%252Fscholar%253Fq%253Dnatural%2520language%2520processing&l=&hl=nl&as_sdt=0&as_vis=1&oi=scholart&sa=X&ved=0ahUKEwjtvqXH8PrZAhXQJ1AKHYFRA34
- Liddy, E. D. (2001). *NAtural Language Processing*. Opgehaald van SU Surface: https://surface.syr.edu/cgi/viewcontent.cgi?referer=https%253A%252F%252Fscholar.google.nl%252Fscholar%253Fq%253Dnatural%2520language%2520processing&l=&hl=nl&as_sdt=0&as_vis=1&oi=scholart&sa=X&ved=0ahUKEwjtvqXH8PrZAhXQJ1AKHYFRA34

google.nl%252Fscholar%253Fq%253Dnatural%2520language%2520processing&am
p=&hl=nl&as_sdt=0&as_vis=1&oi=scholart&sa=X&ved=0ahUKEwjtvqXH8PrZAhXQJ1AKHYFRA34

Maes, A. (2017, mei 14). *Steeds meer huisartsen krijgen burn-out door toenemende werkdruk*. Opgehaald van VP Huisartsen: www.vphuisartsen.nl/nieuws/steeds-meer-huisartsen-krijgen-burn-out-toenemende-werkdruk/

Majumder, P., Mitra, M., & Chaudhuri, B. (2002, November). *N-gram: a language independent*. Opgeroepen op Mei 7, 2018, van mt-archive: <http://www.mt-archive.info/ICUKL-2002-Majumder.pdf>

Maruti Techlabs. (Z.D.). *A guide to the chatbot architecture*. Opgehaald van Maruti Techlabs: <https://www.marutitech.com/chatbots-work-guide-chatbot-architecture/>

Microsoft. (2017, December 13). *Bot scenarios*. Opgeroepen op Maart 23, 2018, van Microsoft Azure: <https://docs.microsoft.com/en-us/azure/bot-service/bot-service-scenario-overview>

Microsoft. (2017, December 13). *Debug bots with the bot framework*. Opgeroepen op April 3, 2018, van Microsoft Azure: <https://docs.microsoft.com/en-us/azure/bot-service/bot-service-debug-emulator>

Microsoft. (2017, December 13). *Develop bots with Bot Builder*. Opgeroepen op Maart 23, 2018, van Microsoft Azure: <https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-overview-getstarted>

Microsoft. (2017, December 13). *Manage conversation flow with dialogs*. Opgeroepen op April 9, 2018, van Microsoft Azure: <https://docs.microsoft.com/en-us/azure/bot-service/nodejs/bot-builder-nodejs-dialog-manage-conversation-flow>

Microsoft. (2018, Maart 22). *BotBuilder*. Opgeroepen op Maart 23, 2018, van Github: <https://github.com/Microsoft/BotBuilder>

Microsoft. (sd). *Woordafbreking*. Opgeroepen op Mei 7, 2018, van Microsoft Azure: <https://azure.microsoft.com/nl-nl/services/cognitive-services/web-language-model/?from=http%3A%2F%2Fresearch.microsoft.com%2Fenus%2Fcollaboration%2Ffocus%2Fcs%2Fweb-ngram.aspx>

Microsoft. (z.d.). *Cognitive Services Directory*. Opgehaald van Microsoft Azure: <https://azure.microsoft.com/nl-nl/services/cognitive-services/directory/search/>

Microsoft. (Z.D.). *Tekst-API van Translator*. Opgehaald van Microsoft Azure: <https://azure.microsoft.com/nl-nl/services/cognitive-services/translator-text-api/>

Microsoft. (Z.D.). *Azure Bot Service*. Opgeroepen op Maart 23, 2018, van Microsoft Azure: <https://azure.microsoft.com/nl-nl/services/bot-service/>

Microsoft. (Z.D.). *Cognitive Services*. Opgehaald van Microsoft Azure.

Microsoft. (Z.D.). *Cognitive Services*. Opgeroepen op Maart 28, 2018, van Microsoft Luis: <https://www.luis.ai/home>

Microsoft. (Z.D.). *Microsoft Bot Framework*. Opgeroepen op Maart 13, 2018, van Microsoft: <https://dev.botframework.com>

- Mitra, M. (2002, November). *N-gram: a language independent approach to IR and NLP*.
Opgeroepen op 05 07, 2018, van mt-archive: <http://www.mt-archive.info/ICUKL-2002-Majumder.pdf>
- Mohanoor, A. (2017, januari 24). *Natural Language Processing Glossary for Programmers*.
Opgeroepen op mei 14, 2018, van miningbusinessdata:
<https://miningbusinessdata.com/natural-language-processing-glossary/>
- Mohanoor, A. (2018, mei 21). *Dialogflow Machine Learning Algorithm*. Opgeroepen op 5 28, 2018, van Mining business data: <https://miningbusinessdata.com/dialogflow-machine-learning-algorithm/>
- new gen apps. (2018, februari 7). *Intent vs Flow Based Chatbot Communication*.
Opgeroepen op april 4, 2018, van new gen apps:
<https://www.newgenapps.com/blog/intent-vs-flow-based-chatbot-communication>
- Nordstörm, B., & Ranta, A. (2008). Advances in Natural Language Processing. In B. Nordstörm, & A. Ranta, *Advances in Natural Language Processing*. Zweden.
- OpeNER. (Z.D.). *Getting started*. Opgeroepen op Maart 28, 2018, van Opener project:
<http://www.opener-project.eu/getting-started/>
- Radboud University. (Z.D.). *Frog*. Opgeroepen op Maart 28, 2018, van Frog:
<http://languagemachines.github.io/frog/>
- Robin. (2009, December 4). *Parts-Of-Speech Tagging*. Opgeroepen op Mei 14, 2018, van worldofcomputing.net: <http://language.worldofcomputing.net/pos-tagging/parts-of-speech-tagging.html>
- Rouse, M. (2017, Februari). *database (DB)*. Opgehaald van TechTarget:
<https://searchsqlserver.techtarget.com/definition/database>
- Schlicht, M. (2016, april 20). *The Complete Beginner's Guide To Chatbots*. Opgehaald van Chatbots Magazine: <https://chatbotsmagazine.com/the-complete-beginner-s-guide-to-chatbots-8280b7b906ca>
- Schlicht, M. (2016). *The Complete Beginner's Guide To Chatbots*. Opgehaald van Chatbots Magazine: <https://chatbotsmagazine.com/the-complete-beginner-s-guide-to-chatbots-8280b7b906ca>
- Schlicht, M. (2016, april 20). *The complete beginner's guide to chatbots*. Opgehaald van Chatbots Magazine: <https://chatbotsmagazine.com/the-complete-beginner-s-guide-to-chatbots-8280b7b906ca>
- Smooch Technologies. (Z.D.). *Power omnichannel conversations in you software*.
Opgeroepen op Maart 13, 2018, van Smooch: <https://smooch.io>
- Standford. (2009, April 7). *Stemming and lemmatization*. Opgeroepen op April 25, 2018, van standford.edu: <https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html>
- Standford. (2009, April 7). *Tokenization*. Opgeroepen op April 24, 2018, van <https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html>
- Standford. (2017, Augustus 7). *Part-of-Speech Tagging*. Opgeroepen op Mei 14, 2018, van standford: <https://web.stanford.edu/~jurafsky/slp3/10.pdf>

- Stanford. (sd). *Information Extraction and Named Entity Recognition*. Opgeroepen op mei 28, 2018, van Stanford:
https://web.stanford.edu/class/cs124/lec/Information_Extraction_and_Named_Entity_Recognition.pdf
- Swaen, B. (2012, november 19). *Formuleren van onderzoeksvragen voor je scriptie*. Opgehaald van Scribbr: <https://www.scribbr.nl/starten-met-je-scriptie/type-onderzoeksvragen/>
- Sylvester, S. (2018, maart). *botbuilder-wit*. Opgeroepen op april 17, 2018, van NPM:
<https://www.npmjs.com/package/botbuilder-wit>
- TensorFlow. (2018, januari 27). *Image Recognition*. Opgehaald van TensorFlow:
https://www.tensorflow.org/tutorials/image_recognition
- TextRazor Ltd. (Z.D.). *Exact meaning from your text*. Opgeroepen op Maart 28, 2018, van TextRazor: <https://www.textrazor.com/>
- The Apache Software Foundation. (Z.D.). *Welcome to Apache OpenNLP*. Opgeroepen op Maart 28, 2018, van OpenNLP: opennlp.apache.org/
- Tuil, K. v. (Z.D.). *Wat is een API?* Opgehaald van Computer World:
<http://computerworld.nl/development/74796-wat-is-een-api>
- Tunkelang, D. (2017, februari 6). *Stemming and Lemmatization*. Opgeroepen op mei 28, 2018, van Medium: <https://queryunderstanding.com/stemming-and-lemmatization-6c086742fe45>
- Turgman, A. (2017, september 17). *graph-bot-dialog*. Opgeroepen op april 23, 2018, van Github: <https://github.com/CatalystCode/bot-graph-dialog>
- Upadhyay, P. (sd). *Removing stop words with NLTK in python*. Opgeroepen op Mei 7, 2018, van Geeksforgeeks: <https://www.geeksforgeeks.org/removing-stop-words-nltk-python/>
- Vo, D. C., Iqbal, K., & Standefer, R. (2017, december 13). *Manage conversation flow with dialogs*. Opgeroepen op april 23, 2018, van Microsoft: <https://docs.microsoft.com/en-us/azure/bot-service/nodejs/bot-builder-nodejs-dialog-manage-conversation-flow?view=azure-bot-service-3.0>
- Vo, D. C., Iqbal, K., Standefer, R., Mak, D., & Delimarsky, D. (2017, december 12). *Prompt for user input*. Opgeroepen op april 24, 2018, van Microsoft:
<https://docs.microsoft.com/en-us/azure/bot-service/nodejs/bot-builder-nodejs-dialog-prompt?view=azure-bot-service-3.0#promptsattachment>
- Zadeh, R. (2016, augustus 17). *Twitter*. Opgeroepen op april 23, 2018, van Twitter:
https://twitter.com/Reza_Zadeh/status/765722701465948160/photo/1?ref_src=twsrc%5Etfw&ref_url=https%3A%2F%2Fadeshpande3.github.io%2FHow-I-Used-Deep-Learning-to-Train-a-Chatbot-to-Talk-Like-Me
- Zerobotlabs. (2016, April 15). *Nestorbot Programming Manual*. Opgeroepen op Maart 13, 2018, van Github: <https://github.com/zerobotlabs/nestorbot>

Bijlage A: Lijst van stopwoorden

De bijlage is in de map Bijlagen A te vinden. Hierin staat een document met Nederlandse stopwoorden.

Bijlage B: Prototype

In de map Prototype staat het prototype van het project.