



Faculteit Bedrijf en Organisatie

Een chatbot automatisch laten bijleren op basis van het gedrag van gebruikers

Ivor Faingnaert

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

Promotor:
Leen Vuyge
Co-promotor:
Sander Goossens

Instelling: Endare

Academiejaar: 2018-2019

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Een chatbot automatisch laten bijleren op basis van het gedrag van gebruikers

Ivor Faingnaert

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

Promotor:
Leen Vuyge
Co-promotor:
Sander Goossens

Instelling: Endare

Academiejaar: 2018-2019

Tweede examenperiode

Woord vooraf

Samenvatting

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus.

Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Inhoudsopgave

1	Inleiding	15
1.1	Probleemstelling	15
1.2	Onderzoeksvraag	16
1.3	Onderzoeksdoelstelling	16
1.4	Opzet van deze bachelorproef	16
2	Stand van zaken	17
2.1	Chatbot	17
2.2	Soorten Chatbots	17
2.2.1	Rule-Bases Bots	18
2.2.2	AI-bot	18
2.2.3	Generalistische bot	18
2.2.4	Specialisatiebot	18

2.3	Natural Language Processing (NLP)	18
2.4	Natural Language Understanding (NLU)	19
2.5	Artificial intelligence	19
2.6	Intents en Entities	19
2.7	Feedback loops	19
2.8	Framework	19
2.9	Luis.ai	20
2.9.1	Verbeteren van de voorspellingen	20
2.10	Wit.ai	21
2.11	API	21
3	Methodologie	23
3.1	Mogelijkheden	23
3.1.1	Ontdekken	23
3.2	Frameworks	27
3.2.1	LUIS	27
3.2.2	Wit.ai	27
3.2.3	IBM Watson Assistant	29
3.2.4	Dialogflow	31
3.2.5	Andere frameworks	32
3.2.6	Conclusie frameworks	32
4	Proof of concept	35
4.1	Gekozen framework	35
4.2	Opzetten LUIS	36

4.3	Software	37
4.3.1	Node.js en TypeScript	37
4.3.2	Bot Framework Emulator	37
4.3.3	Microsoft Bot Framework	37
4.4	Uitwerking	39
5	Conclusie	43
A	Onderzoeksvoorstel	45
A.1	Introductie	45
A.2	State-of-the-art	45
A.2.1	Chatbots	45
A.2.2	NLP (Natural language processing)	46
A.2.3	Intents	46
A.2.4	Entiteiten	46
A.2.5	Trainen	46
A.2.6	Feedback loops	46
A.3	Methodologie	47
A.4	Verwachte resultaten	47
A.5	Verwachte conclusies	47
	Bibliografie	49

Lijst van figuren

2.1	Logo Luis.ai (Hsu, 2017)	20
2.2	Logo Wit.ai (Dimitrova, 2016)	21
3.1	Intent aanmaken Luis.ai	24
3.2	Entities aanmaken Luis.ai	24
3.3	Testen van de chatbot Luis.ai	25
3.4	Bot Framework Emulator	26
3.5	Review endpoint utterances Luis.ai	26
3.6	Intent aanmaken Wit.ai	27
3.7	Intent aanmaken Wit.ai	28
3.8	Intent aanmaken Wit.ai	28
3.9	Intent aanmaken IBM Watson Assistant	29
3.10	Utterances toevoegen IBM Watson Assistant	29
3.11	testen van de Skill IBM Watson Assistant	30
3.12	weak understandings IBM Watson Assistant	30
3.13	Intent aanmaken Dialogflow	31
3.14	Intent aanmaken Dialogflow	31
3.15	Intent aanmaken Dialogflow	32

Lijst van tabellen

1. Inleiding

De inleiding moet de lezer net genoeg informatie verschaffen om het onderwerp te begrijpen en in te zien waarom de onderzoeksvraag de moeite waard is om te onderzoeken. In de inleiding ga je literatuurverwijzingen beperken, zodat de tekst vlot leesbaar blijft. Je kan de inleiding verder onderverdelen in secties als dit de tekst verduidelijkt. Zaken die aan bod kunnen komen in de inleiding (**Polleffiet2011**):

- context, achtergrond
- afbakenen van het onderwerp
- verantwoording van het onderwerp, methodologie
- probleemstelling
- onderzoeksdoelstelling
- onderzoeksvraag
- ...

1.1 Probleemstelling

Uit je probleemstelling moet duidelijk zijn dat je onderzoek een meerwaarde heeft voor een concrete doelgroep. De doelgroep moet goed gedefinieerd en afgeleid zijn. Doelgroepen als “bedrijven,” “KMO’s,” systeembeheerders, enz. zijn nog te vaag. Als je een lijstje kan maken van de personen/organisaties die een meerwaarde zullen vinden in deze bachelorproef (dit is eigenlijk je steekproefkader), dan is dat een indicatie dat de doelgroep goed gedefinieerd is. Dit kan een enkel bedrijf zijn of zelfs één persoon (je co-promotor/opdrachtgever).

1.2 Onderzoeksvraag

Wees zo concreet mogelijk bij het formuleren van je onderzoeksvraag. Een onderzoeksvraag is trouwens iets waar nog niemand op dit moment een antwoord heeft (voor zover je kan nagaan). Het opzoeken van bestaande informatie (bv. “welke tools bestaan er voor deze toepassing?”) is dus geen onderzoeksvraag. Je kan de onderzoeksvraag verder specificeren in deelvragen. Bv. als je onderzoek gaat over performantiemetingen, dan

1.3 Onderzoeksdoelstelling

Wat is het beoogde resultaat van je bachelorproef? Wat zijn de criteria voor succes? Beschrijf die zo concreet mogelijk. Gaat het bv. om een proof-of-concept, een prototype, een verslag met aanbevelingen, een vergelijkende studie, enz.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

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

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

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

2. Stand van zaken

In dit onderdeel zal de werking worden uitgelegd van een chatbot. Er zullen ook enkele van de meest bekende frameworks vergeleken worden met mekaar.

2.1 Chatbot

Een chatbot is software die gemaakt is voor het automatiseren van een bepaalde taak. Er kan met de chatbot geconverseerd worden via een gebruikers interface. Deze chatbot heeft op zich toegang tot bepaalde data die toegankelijk is via een API zodat het deze kan leveren aan een bepaalde gebruiker die hier om vraagt.

De gebruikers interfaces kunnen zich bevinden op Messenger, Skype, Slack, WhatsApp, enz. Ook Siri en Alexa zijn bots, ze variëren van een chatfunctie tot een spraakassistent. Het is de bedoeling dat de gebruiker een vraag intypt en de chatbot geeft daar zo een gepast mogelijk antwoord op. Een chatbot kan zelf ook altijd een vraag stellen, dat kan zijn voor de naam van de persoon te weten te komen, zodat deze bot persoonlijker kan antwoorden. Het kan ook zijn voor een quiz of iets dergelijks.

Met NLP en machine learning is het mogelijk om mensentaal aan een chatbot aan te leren. Deze hebben een zeer grote rol bij het maken van zo een bot. (Elovic, 2017)

2.2 Soorten Chatbots

Chatbots kunnen onderverdeeld worden in verschillende categorieën.

2.2.1 Rule-Bases Bots

Een rule based bot is een geautomatiseerde bot die reageert op bepaalde acties en sleutelwoorden en worden vaak ingezet op gespecialiseerde taken. Eén van de talen van zo een bot is AIML (Artificial Intelligence Markup Language), dat is een taal gebaseerd op XML. Deze laat toe om ontwikkelaars regels op te stellen voor een bot die deze moet volgen. Het is onmogelijk om regels te schrijven voor elk mogelijk scenario. (Shridhar, 2017)

2.2.2 AI-bot

Een AI-bot simuleert het gedrag van een mens. Deze bot zal de bekende Turing-test moeten doorstaan. Deze test is ontwikkeld door Alan Turing in 1950. Hierin wordt gekeken of een mens onderscheid kan maken tussen een echte mens en een computer.

Deze bots zijn getraind aan de hand van een bepaalde hoeveelheid vragen. Voor elke vraag vindt de bot het meest relevante antwoord van alle mogelijke antwoorden. Zulke bots moeten ook rekening houden met de spelling en zinsbouw. Eenmaal deze ook goed getraind zijn daarop, zijn ze veel beter dan een rule-based bot.

2.2.3 Generalistische bot

Deze bots gebruiken vooral data uit databases en zoekmachines maar kunnen de gebruiker ook koppelen aan diensten. Deze heeft geen specifieke taak maar kan op een grote selectie van vragen antwoorden. Eénmaal de gebruiker iets specifieker in detail wil gaan over een bepaald onderwerp, kan deze bot overgaan naar een specialisatiebot.

2.2.4 Specialisatiebot

Een specialisatiebot biedt een gespecialiseerde taak aan, dit vaak in tegenstelling tot een generalistische bot. Deze bot is uitsluitend bedoeld voor een specifiekere taak.

Generalistische en specialisatie bots werken vaak samen. Meestal begint het bij een generalistische bot die nog niet specifiek op iets ingaat. Het eindigt dan met een specialisatiebot die dan dieper ingaat op wat je precies wil weten.

2.3 Natural Language Processing (NLP)

Vroeger typten we enkel commando's om te communiceren met een computer. Nu wordt er geprobeerd om met menselijke taal te communiceren met een computer. Natural Language Processing is de techniek die wordt gebruikt om menselijke taal te begrijpen.

NLP wordt al bij vele alledaagse dingen gebruikt. Bij de spellingscontroles in een e-mail of in de office pakketten. Bij een chatbot is dit al wat gecompliceerder, daar moet ook

gekeken worden naar wat de gebruiker exact bedoeld met zijn input, de juiste intentie eruit halen.

NLP in de Nederlandse taal staat nog maar in zijn beginschoenen en zullen in vergelijking met de Engelstalige versie veel minder goed functioneren. (Doijer, 2018)

2.4 Natural Language Understanding (NLU)

Natural Language Understanding is een onderdeel van Natural Language Processing. Het is een vitaal onderdeel van een succesvol NLP. NLU focust zich primair op wat een bepaalde input betekent. Een input kan een zin zijn, gewone tekst of spraak. (Rouse, 2018)

2.5 Artificial intelligence

Machine learning werkt zonder directe input van een mens. Het probeert op zichzelf om algoritmes te verbeteren. Een bepaald algoritme heeft een doel. Machine learning zal trachten het algoritme te perfectioneren zodat de uitkomsten van nieuwe input dichter en dichter bij dat doel zullen liggen.

2.6 Intents en Entities

Als de gebruiker iets vraagt aan een chatbot is het doel van zijn vraag de intent, wat dus de intentie is van die vraag. Een intent kan ook entiteiten bevatten. Entiteiten zijn details van een intent. Als de gebruiker bijvoorbeeld vraagt aan de chatbot voor een vlucht te boeken, is de bestemming een entiteit.

2.7 Feedback loops

Wanneer een bepaalde input van een gebruiker nog niet is opgenomen in het machine learning algoritme van de chatbot is het de bedoeling dat aan de hand van deze feedback loops deze input op een automatische manier wordt gelinkt met de juiste intent en deze ongetrainde input te gebruiken voor de chatbot verder te trainen en deze slimmer te maken.

2.8 Framework

Vandaag de dag bestaan er al veel API's die ontwikkelaars kunnen gebruiken voor AI/NLP services. Zelf zonder veel kennis van deze kun je deze makkelijk gebruiken. Luis.ai, Wit.ai zijn enkele voorbeelden van deze frameworks.

2.9 Luis.ai



Figuur 2.1: Logo Luis.ai (Hsu, 2017)

Language Understanding (LUIS) is een op cloud gebaseerde API service van microsoft voor het bepalen van de intentie van een gebruiker. De input van een bepaalde gebruiker wordt via een Luis endpoint API verwerkt en krijgt daarop een gepast resultaat terug. De REST API van Luis kan gebruikt worden bij elk product, framework of service dat gebruik maakt van een HTTP request. (Berry, 2019)

2.9.1 Verbeteren van de voorspellingen

Nadat een Luis model is gepubliceerd en input ontvangt, zijn er enkele methodes voorzien om het model te verbeteren.

De eerste methode is aan de hand van patronen. Deze zorgen voor betere accuraatheid zonder veel meer voorbeeldzinnen te hoeven toevoegen. In de voorbeeldzinnen staan veel synoniemen, verschillende lengtes van zinnen, verschillende woordvolgordes, enz. Een patroon kan de woordvolgorde veel beter begrijpen.

De volgende methode werkt met woordengroeplijsten. Zo een lijst somt de woorden of zinnen op die verwant zijn met de applicatie. Wat Luis dan leert over 1 van die woorden in een lijst, wordt automatisch toegepast op die andere woorden of zinnen in die bepaalde lijst. Het doel is voor het verbeteren van het identificeren van de juiste intent en entiteiten.

Er zijn 2 soorten types van lijsten. Een Interchangeable lijst is voor waarden die synoniemen zijn van mekaar. Een Non-interchangeable lijst is voor waarden die belangrijker zijn dan normale woorden. Dit helpt veel voor het bepalen van de juiste intent. Een Non-interchangeable lijst wordt ook gebruikt voor woorden die niet veel gebruikt worden of voor woorden of zinnen uit een andere taal, zo leert Luis rekening te houden met deze. Hierin bevinden zich meer de termen specifiek aan de applicatie. Bijvoorbeeld als de applicatie zal gaan over winkelen zullen er bepaalde termen gaande over winkelen zeer belangrijk zijn voor de app maar geen synoniemen zijn van mekaar.

De laatste methode voor het verbeteren van de voorspellingen is het actieve bijleren van de bot. Luis selecteert hier inputs die hij niet zeker van is wat de juiste intent is en zet deze in een lijst. De bedoeling is dat deze gevalideerd en gekoppeld worden aan de juiste intent met de juiste entiteiten. Dit moet echter manueel gebeuren door de eigenaar van de applicatie. Als deze inputs allemaal gelinkt zijn aan de juiste intent kan het model opnieuw getraind worden.

Luis plaats inputs op de lijst waarvan hij helemaal niet zeker is waarbij deze behoort of

waarbij hij twijfelt tussen 2 of meerdere intents.

Het doel van dit onderzoek is om een manier te vinden om de inputs op deze lijst niet 100% handmatig te moeten linken aan de juiste intent.

2.10 Wit.ai



Figuur 2.2: Logo Wit.ai (Dimitrova, 2016)

Dit botframework is ontwikkeld door facebook. Wit helpt de gebruikers die iets zeggen tegen de chatbot van de app te verstaan. Wit dient ingesteld te worden door voorbeelden van zinnen toe te voegen net zoals Luis. Hoe meer voorbeelden er worden gegeven, hoe beter de bot de gebruikers zal begrijpen. De verzameling van voorbeeldzinnen voor 1 bepaalde intentie koppelt je dan aan de juiste intent. Er kan gemakkelijk gepraat worden met wit via de endpoint API.

Net zoals bij Luis heb je een inbox met zinnen die zijn ingegeven door een gebruiker in je chatbot zelf die je manueel moet linken aan de juiste intent. Wit stelt zelf al voor bij welke intent hij een bepaalde zin zou plaatsen. Je kan deze aanpassen indien deze fout is en de zinnen valideren. Deze worden dan opgenomen in het model als deze opnieuw wordt getraind. Ook dit deel zoals in Luis zou moeten kunnen gedeeltelijk geautomatiseerd worden. (Wit.ai, 2019)

2.11 API

Via een application programming interface (API) kan een programma communiceren met een ander programma of onderdeel hiervan. Bij een chatbot is het de bedoeling dat we in de applicatie die we ontwikkelen toegang krijgen tot de componenten van de chatbot. Zo kun je via het endpoint API van Luis of Wit de juiste intent uit een zin halen door de API aan te spreken.

Naar mate van toegankelijkheid van de API van deze platformen zal er moeten gekeken worden of het mogelijk om de inputs op te halen waar de bot aan twijfelt en deze automatisch te linken aan de juiste intent en deze inputs weer door te sturen naar het platform en zo linken aan de juiste intent zodat het model opnieuw kan worden getraind. (van Tuil, 2011)

3. Methodologie

In dit onderdeel zal de onderzoeksvraag 'Kunnen we een chatbot automatisch laten bijleren op basis van het gedrag van gebruikers?' onderzocht worden. Hierin zal er gekeken worden naar enkele van de meest bekendste frameworks voor chatbots te kunnen trainen. Het is zeer belangrijk dat hierbij de API van deze frameworks toegankelijk zijn en de juiste methoden bevat zodat de input van de gebruiker kan verwerkt worden op de juiste manier.

3.1 Mogelijkheden

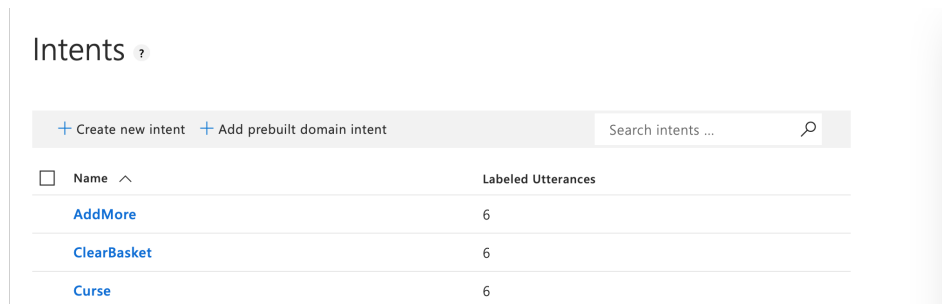
Eerst en vooral was er voorbereiding nodig want er was nog niet echt ervaring op vlak van chatbots. Wat zijn de mogelijkheden van zo een chatbot framework? Hoe worden intents aangemaakt en hoe werken die? Hoe worden entities aangemaakt en hoe voeg je die toe aan een bepaalde intent? Hoe wordt de bot getraind? Hoe wordt er dan effectief omgegaan met de input van de gebruiker waarbij de bot niet goed weet bij welke intent die hoort en hoe koppel je deze dan aan de juiste intent? Wat kun je nog allemaal met zo een framework doen? Hoe gebruik ik de API? Op al deze vragen zocht ik een antwoord.

3.1.1 Ontdekken

De keuze van het eerste framework voor de eerste ervaringen op te doen was niet zo moeilijk. Hierbij kwam LUIS.ai al snel naar boven doordat er op het stagebedrijf al gewerkt wordt hiermee en er een voorbeeld ter beschikking was gesteld waarop er kon worden verder gebouwd.

Het trainen van een chatbot via LUIS is zeer gebruiksvriendelijk. Eerst en vooral dient er

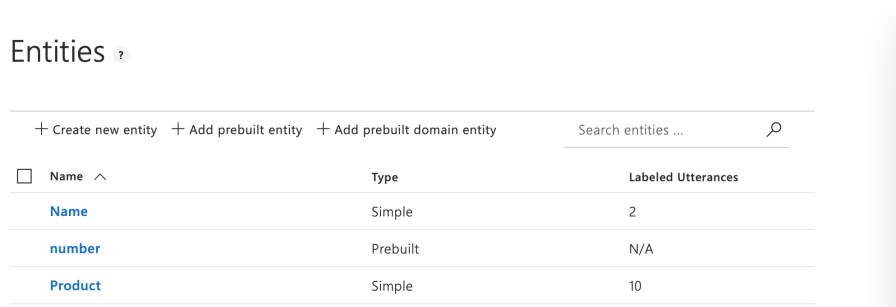
een nieuwe app te worden aangemaakt. Dit is zeer eenvoudig door de vereiste naam, de taal die er zal gesproken worden tegen de bot en eventueel een beschrijving toe te voegen. Eénmaal de app is gecreëerd dienen er Intents aangemaakt te worden. Een voorbeeld van een intent kan zijn "Greeting", hierin vallen dus alle begroetingen. Per intent dien je dan utterances toe te voegen. Utterances zijn voorbeeldzinnen die worden gelinkt met deze gemaakte intent. Door deze voorbeeldzinnen weet LUIS wat een "Greeting" is. Een voorbeeld die zou passen bij een begroeting is "Hi, I'm Ivor". Hoe meer zinnen je toevoegt, liefst allemaal met een verschillende zinsbouw, hoe beter LUIS de input van de gebruikers zal herkennen en aan deze intent zal linken.



Figuur 3.1: Intent aanmaken Luis.ai

In de meeste gevallen is een utterance toevoegen niet zomaar eenvoudigweg een zin toevoegen. Meestal zal er in een utterance één of meerdere entities terug te vinden zijn. Bijvoorbeeld in voorgaand voorbeeld bij de intent "Greeting" konden we een utterance toevoegen "Hello, I'm Ivor". In deze utterance is Ivor een entity, namelijk een naam. Het voordeel van deze entities uit de zinnen te halen is dat bijvoorbeeld de chatbot dan al direct weet dat "Ivor" de naam is van de persoon die communiceert met de bot en kan de bot deze persoon aanspreken met zijn naam. Dit maakt het gesprek al persoonlijker.

LUIS heeft zelf al een redelijk aantal ingebouwde entities, zoals nummers, leeftijden, e-mail adressen, ... en kan deze makkelijk zelf al herkennen. Indien er geen ingebouwde entity bestaat kan je zelf een entity maken en deze makkelijk linken aan het juiste woord in een utterance.



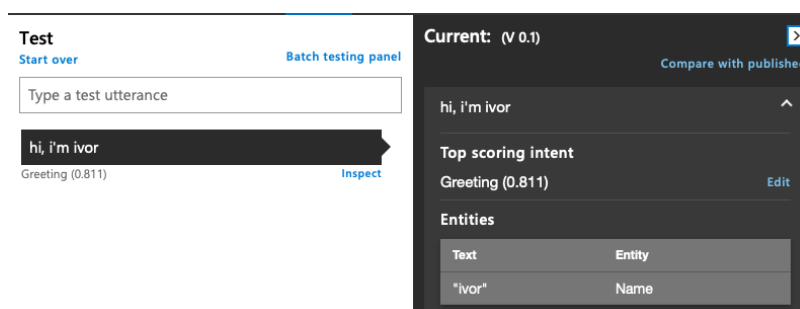
Figuur 3.2: Entities aanmaken Luis.ai

Eénmaal je enkele intents aangemaakt hebt en bij elke intent enkele utterances hebt toegevoegd (het is aangeraden van er minimum 5 toe te voegen voor een betere accuraatheid) met daarbij eventueel de gepaste entities kan de chatbot als eens getest worden.

Het enige dat nog moet gebeuren vooraleer er kan getest worden is LUIS trainen. Na het toevoegen van nieuwe intents, entities en utterances dient LUIS opnieuw getraind te worden met de nieuwe data. Na enkele seconden trainen staat alles klaar om getest te worden.

Op de site van LUIS zelf kan er gepraat worden met de chatbot via een simpele interface. Hierin kan er simpelweg gepraat worden met de bot door utterances in te geven. Als voorbeeld kan er weer gebruik worden gemaakt van de utterance: "Hi, I'm Ivor". Zo kun je zien of deze de juiste intent herkent en de entity er juist uithaalt met daarbij het percentage van de zekerheid dat LUIS juist is. Bij dit voorbeeld is LUIS er 81 percent zeker van dat deze utterance hoort bij de intent "Greeting" met de entity "Ivor" als "name".

Indien de utterance niet hoort bij de door LUIS geselecteerde intent kan deze handmatig aangepast worden naar de correcte en kan LUIS opnieuw getraind worden zodat dit de volgende keer correct is.

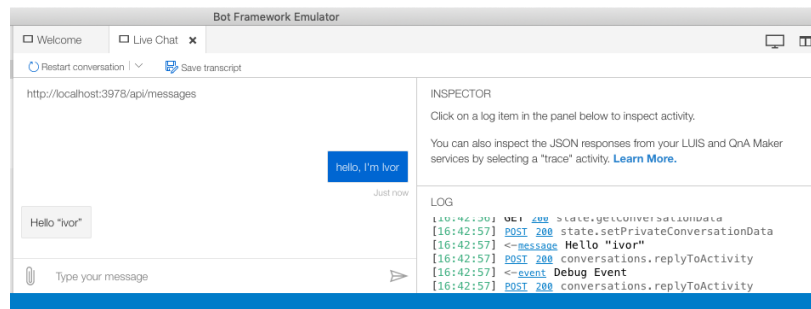


Figuur 3.3: Testen van de chatbot Luis.ai

Als alles goed verloopt en bij het testen zijn er niet veel fouten, kan de applicatie gepubliceerd worden. Dit zorgt ervoor dat je vanuit je eigen applicatie de chatbot kan integreren.

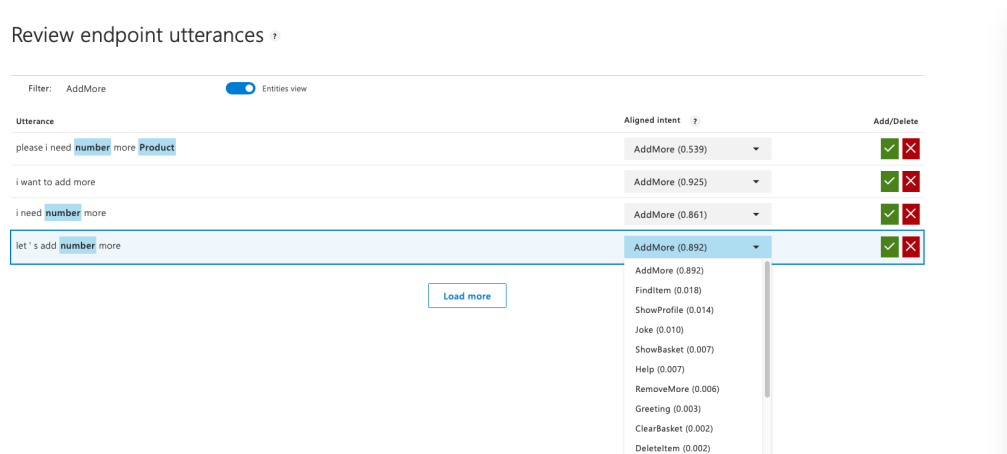
Eénmaal deze geïntegreerd is in een eigen applicatie kan er lokaal getest worden op wat de chatbot reageert op bepaalde utterances. Dit kan makkelijk getest worden via de "Bot Framework Emulator". Dit is een zeer handig applicatie voor het testen van de bot. Er is een interface zodat het direct lijkt op een echt chatvenster. Hiernaast is er ook een inspector. Hierin komen alle GET en POST boodschappen die worden gestuurd en ontvangen. Zo kan je makkelijk zien als er iets is misgelopen.

Als je LUIS applicatie gepubliceerd is en deze ontvangt input van gebruikers door bijvoorbeeld via de Bot Framework Emulator, analyseert LUIS elk van deze utterances. LUIS probeert die dan te linken aan de juiste intent. LUIS zal nooit 100 percent zeker zijn dat een bepaalde utterance behoort tot één specifieke intent. LUIS zal de utterances die de gebruiker invoert waar hij niet zo zeker van is verzamelen onder "Review endpoint utterances". Onder deze pagina bevinden zich alle utterances waar LUIS minder zeker van is zodat je deze manueel kan linken aan de juiste intent, utterance per utterance. Bij elke utterance geeft LUIS zelf al een schatting bij welke intent elke deze hoort. Afbeelding 3.5 toont de schattingen van LUIS aan de hand van het percentage dat hij zeker is dat de utterance bij een bepaalde intent hoort. Als deze juist is kan deze toegevoegd worden.



Figuur 3.4: Bot Framework Emulator

Indien dit niet juist is, kan er via een dropdown menu de juiste intent geselecteerd worden en deze dan handmatig toevoegen. Bij deze dropdown staan alle intents in volgorde van zekerheid met het percentage erbij. Bij dit voorbeeld "Let's add 3 more" is de gesuggereerde intent van LUIS AddMore met een percentage van 89.2 percent. De tweede suggestie is FindItem met een percentage van 1.8 percent. Door het grote verschil aan percentages wordt er niet getwijfeld dat deze utterance hoort bij de intent "AddMore". De entities van elk van deze utterances worden door LUIS automatisch gedetecteerd. Moest het zijn dat LUIS dat niet gedaan heeft kan je de entities handmatig aanduiden.



Figuur 3.5: Review endpoint utterances Luis.ai

Dit was een zeer goed begin om te begrijpen wat de concepten zijn van een framework voor het ontwikkelen van chatbots. Door het op voorhand bestuderen van één bepaald framework voor het maken van chatbots worden al veel aspecten zeer duidelijk en deze zijn meestal wel hetzelfde voor de andere frameworks ook. Door LUIS al eens te bestuderen is het doel van deze bachelorproef direct veel duidelijk geworden. De review endpoint utterances uit afbeelding 3.5 dient nu allemaal handmatig te worden goedgekeurd en nagekeken. Zoals er kan gezien worden is het niet nodig om bij elke utterance handmatige controle te steken. Soms is de score van de eerste en de tweede gesuggereerde intent zo groot dat er geen twijfel mogelijk is bij welke intent deze behoort. Bij alle utterances waarbij dit het geval is zou deze handmatige controle achterwege kunnen worden gelaten.

In het volgende deel worden verschillende van de bekendste frameworks voor het maken

van bot vergeleken. Hierbij zal gekeken worden of deze een API hebben en of deze toegankelijk genoeg is voor de 'review endpoint utterances' op te halen en deze via een API-call op te halen in een eigen applicatie, deze automatisch te controleren en goedkeuren indien dit kan. Zo zullen uiteindelijk de utterances die nog handmatig zullen moeten gecontroleerd worden veel minder zijn.

3.2 Frameworks

3.2.1 LUIS

Nu dat de basisconcepten van LUIS duidelijk zijn door het vorige hoofdstuk, kan er worden gekeken of LUIS een API heeft en of deze toegankelijk genoeg is voor de onderzoeksvraag te kunnen oplossen. Al snel is duidelijk dat er effectief een API bestaat genaamd 'LUIS Programmatic APIs v2.0'. Hierin is er een grote variatie, van het toevoegen van intents en entities naar het toevoegen van een nieuwe utterance en meer. Voor het ophalen van utterances is er één API-call 'Review labeled examples'. Deze call heeft enkel de utterances weer die al zijn toegevoegd aan een bepaalde intent dus jammer genoeg niet de 'unlabeled examples'. De utterances waarvan LUIS niet exact weet bij welke intent deze hoort, die op de site van LUIS zelf onder 'review endpoint utterances' staan kunnen niet worden opgehaald via een API-call en kunnen dus niet automatisch gecontroleerd worden.

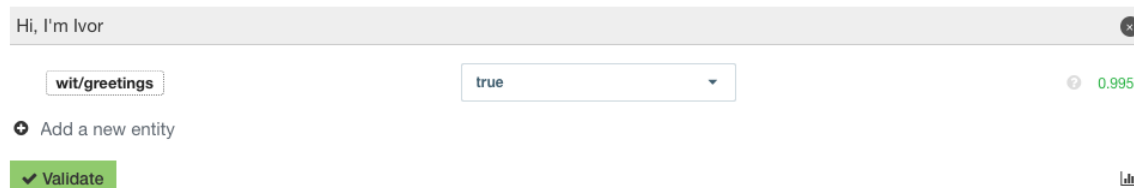
3.2.2 Wit.ai

Wit.ai hoort ook bij één van de bekendste frameworks voor het ontwikkelen van chatbots. De grote lijnen van dit framework zijn wat hetzelfde als die van LUIS.

Eerst en vooral moet er een applicatie worden aangemaakt met de naam, de taal waarin de bot zal functioneren en eventueel een omschrijving. Het toevoegen van intents en utterances werkt op een verschillende manier maar komt uiteindelijk op hetzelfde neer. Afbeelding 3.6 toont hoe dit in zijn werk gaat. Hier moet je beginnen met het invoeren van een utterance en dan kan deze gelinkt worden met een intent. Hier kunnen we weer het voorbeeld gebruiken van 'Hi, I'm Ivor'.

Test how your app understands a sentence

You can train your app by adding more examples



The screenshot shows the Wit.ai interface for testing a sentence. At the top, it says "Test how your app understands a sentence" and "You can train your app by adding more examples". Below this is a text input field containing "Hi, I'm Ivor". Under the input field, there are two buttons: "wit/greetings" and "true". To the right of these buttons, there is a green status indicator showing a question mark and the value "0.995". At the bottom left, there is a green button labeled "Validate". At the bottom right, there is a small icon representing a bar chart.

Figuur 3.6: Intent aanmaken Wit.ai

Wit herkent dit al automatisch als een 'Greeting', hier zie je ook het percentage van 99

percent. Dit percentage is zoals bij LUIS de zekerheid dat Wit denkt dat een intent bij een bepaalde utterance hoort. Indien Wit deze niet herkent of de intent bestaat simpelweg nog niet, kan deze hier aangemaakt en toegevoegd worden. Dit werkt juist op dezelfde manier bij het toevoegen van een entity.

Het weergegeven van de al ingevoerde utterances verschilt ook van LUIS. In Wit is er een tabblad 'Samples' waarbij alle utterances worden getoond onder elkaar. Hier kan je de utterances filteren per intent.

Samples

Filter by: wit/greetings

Samples with wit/greetings

“ Search through your samples.

Text
“ Hi, I'm Ivor x
“ hello x

Figuur 3.7: Intent aanmaken Wit.ai

Ook heeft Wit net zoals LUIS een pagina waar de utterances inkomen van de gebruikers die praten met de bot waarvan Wit niet exact weet bij welke intent deze horen. Hierin suggereert wit ook aan de hand van een percentage bij welke intent hij denkt dat een bepaalde utterance hoort. Deze kun je dan ook nakijken of dit klopt, indien dit niet juist is kan je deze veranderen via de dropdown en deze dan valideren. De gevalideerde utterance wordt daarna toegevoegd aan de lijst van de samples met zijn intent.

Ook bij Wit zouden deze reviews moeten worden geautomatiseerd. Nu de basiswerking van Wit duidelijk is geworden, kan er worden gekeken naar de toegankelijkheid van de API.

show me my basket

intent get_basket 0.761

+ Add a new entity

✓ Validate 📊 🗑

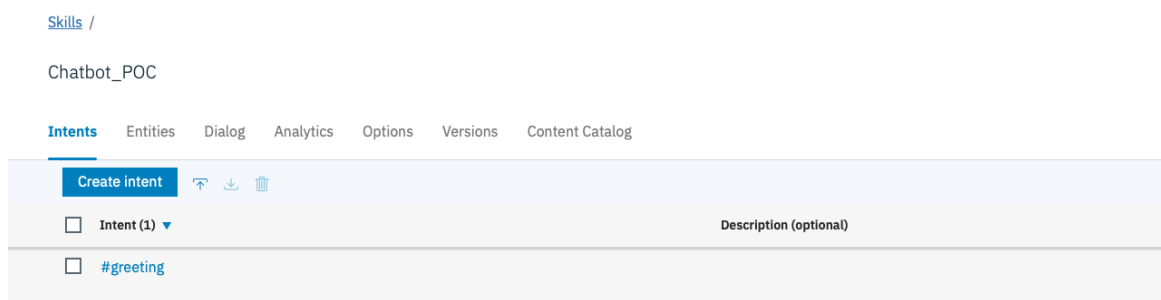
Figuur 3.8: Intent aanmaken Wit.ai

Wit heeft ook wel degelijk een API ter beschikking. De API lijkt zeer sterk op deze van LUIS, er kunnen nieuwe utterances aan worden toegevoegd, samen met nieuwe intent en entities. Ook kunnen hier de utterances worden opgehaald die al gelinkt zijn aan een bepaalde intent. Net zoals bij LUIS zijn de utterances waarvan Wit niet zeker weet bij welke intent ze horen hier niet bij. Deze zullen dus via de webpagina zelf moeten gevalideerd worden en zal niet automatisch kunnen worden gecontroleerd.

3.2.3 IBM Watson Assistant

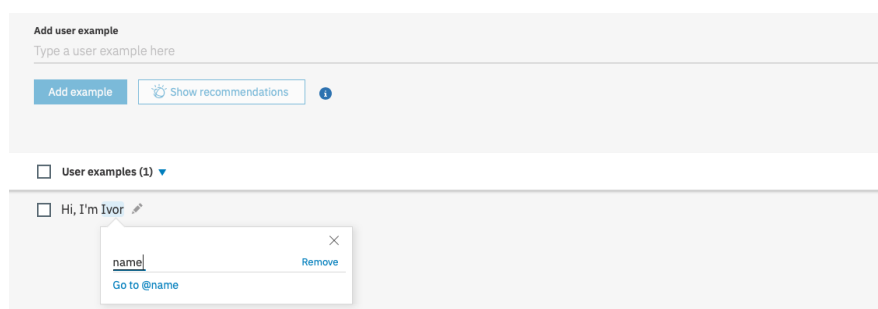
Een derde framework voor het bouwen van chatbots is IBM Watson Assistant. Dit framework werkt ook gedeeltelijk op een andere manier dan de vorige onderzochte frameworks. Voor het ontwikkelen van een chatbot in IBM Watson Assistant zijn er drie grote stappen.

De eerste stap is het maken van een 'Skill'. Onder de skill bevindt zich alle logica van de bot, de bot wordt hier getraind. Een skill kan hier vergeleken worden met een App dat je eerst moet maken bij de vorige frameworks. Eerst en vooral dien je een skill aan te maken. Een skill heeft een naam, de taal waarin tegen de bot zal gesproken worden en eventueel een omschrijving. Eénmaal deze skill is aangemaakt kan je zeer eenvoudig intents maken en aan elke intent utterances toevoegen.



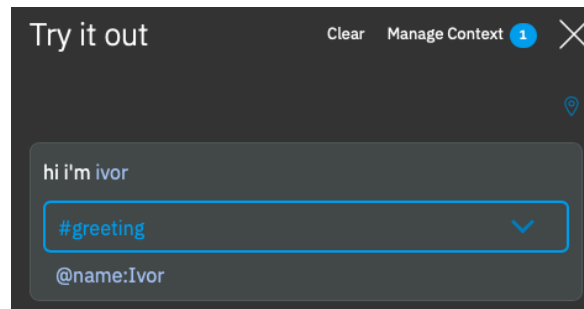
Figuur 3.9: Intent aanmaken IBM Watson Assistant

Na het aanmaken van een bepaalde intent wordt er naar de detailpagina van deze intent genavigeerd. Op deze pagina kan je utterances toevoegen aan deze intent. De werking van dit toevoegen lijkt zeer sterk op die van LUIS. Een entity kan hier ook zeer makkelijk aangeduid worden door de gewenste woorden te selecteren en de juiste entity aan te klikken. Indien de entity nog niet bestaat kan je deze hier even snel direct aanmaken en toewijzen.



Figuur 3.10: Utterances toevoegen IBM Watson Assistant

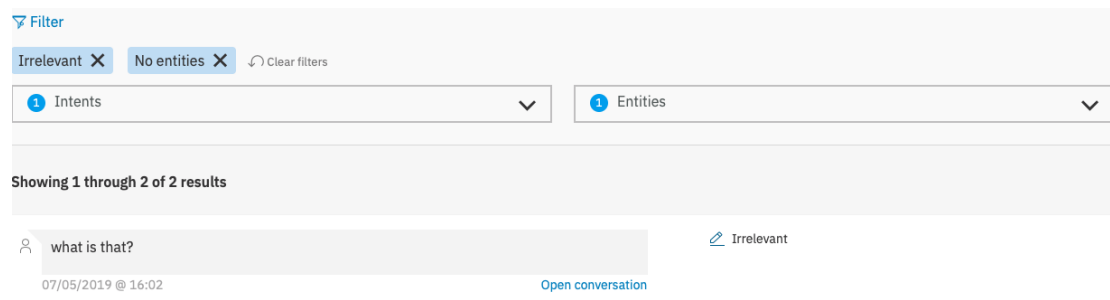
Het trainen van de Skill gaat hier vanzelf. Daarna kan de bot al eens getest worden met de al toegevoegde intents en entiteiten. Als voorbeeld kan er weer worden ingegeven 'Hi, I'm Ivor'. Er kan al direct worden gezien aan welke intent de utterance wordt gelinkt. In vergelijking met LUIS en Wit zijn er hier bij het testen geen percentages. Er is dus niet geweten hoeveel percent de Skill zeker is van een bepaalde intent. Ook kan er direct gezien worden welke entiteiten er worden herkend. Dit wordt getoond in figuur 3.11.



Figuur 3.11: testen van de Skill IBM Watson Assistant

De tweede stap in dit proces is het deployen van de skill met een assistant. In deze stap wordt de skill aan een assistant gelinkt. Hierin is het eerst nodig om een nieuw assistant te maken. Alleen een naam is vereist, een beschrijving is optioneel. Eénmaal de assistant is gecreëerd, kan er een skill worden aan toegevoegd. Daarna kan je een kanaal kiezen waarop je de bot kan deployen. Hierbij is er ook een preview waarop kan worden getest.

De utterances waarvan de assistant niet zeker is bij welke intent deze horen worden bijgehouden en kunnen teruggevonden worden onder het tabblad 'Analytics'. Hier kan je doorklikken naar de 'Weak understandings'. Hierin kan je de utterances linken aan de juiste intent. Wat verschillend is met vorige frameworks is dat de assistant geen enkele intent aanbeveelt aan de hand van scores. Hier moet er 100 percent zelf worden nagedacht bij welke intent deze hoort.



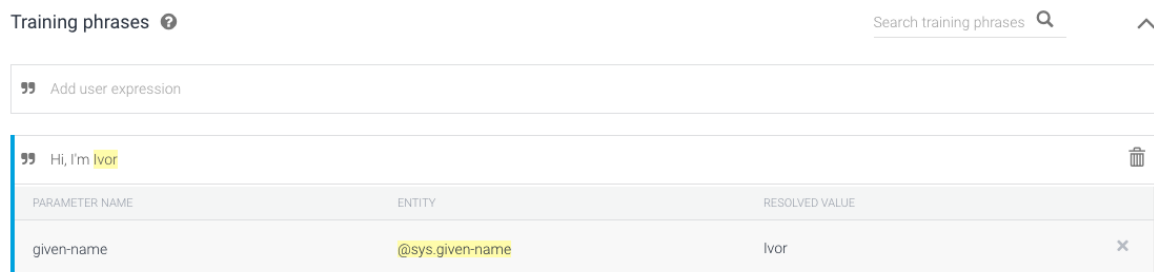
Figuur 3.12: weak understandings IBM Watson Assistant

IBM Watson Assistant heeft ook een API. Deze API is zeer uitgebreid en bevat zo goed als alle aspecten van de site zelf. Van het creëren van een intent tot het toevoegen van nieuwe utterances en veel meer. In vergelijking met vorige frameworks heeft de assistant van IBM ook een endpoint voor het ophalen van utterances waarvan de assistant niet zeker is. Het enige probleem hier is dat IBM geen scores voorziet bij een utterance dus zal er altijd manuele controle nodig zijn. De niet gelinkte utterances kunnen hier wel opgehaald worden met een eigen applicatie maar de scores ontbreken dus zal er niet automatisch kunnen gecontroleerd worden of deze utterance is gelinkt met de juiste intent.

3.2.4 Dialogflow

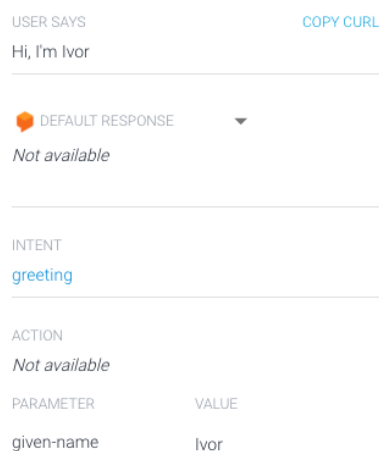
Dialogflow is een chatbot framework gemaakt door Google. Net zoals de vorige frameworks heb je hier de mogelijkheid om op een relatief eenvoudige manier een chatbot op te zetten. Hierbij is de eerste stap om een 'Agent' te maken. Dit wordt gedaan door de naam van de Agent, de taal die je tegen de bot zal praten en de tijdzone in te geven.

Een intent aanmaken is hier ook zeer eenvoudig, dit kan door de naam van de intent in te geven. Daarna kunnen er al direct utterances aan toegevoegd worden bij 'Training phrases' zoals er wordt getoond in afbeelding 3.13. De entities kunnen ook toegevoegd worden aan de intent door deze te selecteren en de juiste entity te kiezen. Hierbij zijn er al een aantal ingebouwde entities voorzien. Indien de entity niet in de lijst staat kan er een nieuwe worden aangemaakt in het 'Entities' tabblad, dit kan eenvoudigweg door de naam van de entity in te vullen.



Figuur 3.13: Intent aanmaken Dialogflow

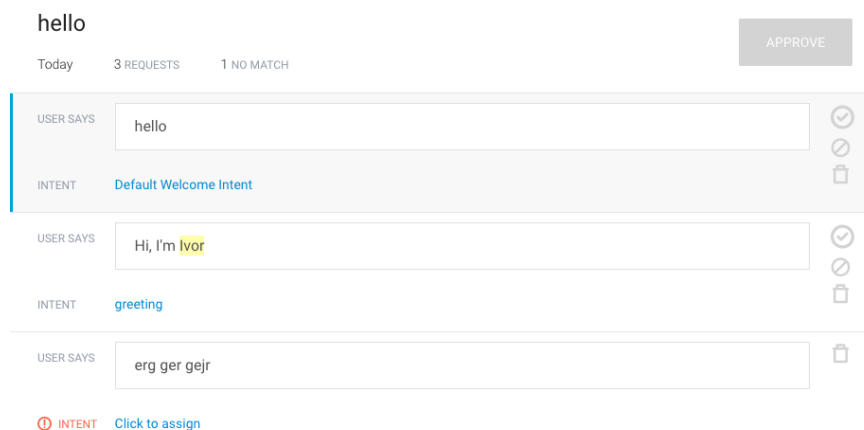
Deze gemaakte intents met de toepasselijk toegevoegde utterances kunnen al direct getest worden door de console die is voorzien. Er kan al voorbeeld 'Hi, I'm Ivor' gebruikt worden. Hierin kan gezien worden op afbeelding 3.14 dat deze input wordt geanalyseerd door de bot. Deze ziet dat de voorbeeldzin past bij de juiste intent 'Greeting' en dat de waarde 'Ivor' een entity is.



Figuur 3.14: Intent aanmaken Dialogflow

Bij dialogflow heb je ook de kans om de utterances die de gebruiker ingeeft waarvan de

bot niet zeker is bij welke intent deze horen te bekijken en te valideren. Hier is er zowat hetzelfde probleem als bij de IBM Watson Assistant, als dialogflow echt niet zeker is van de intent zal er in de testconsole de intent 'default fallback intent' komen. Bij het training tabblad dien je dan zelf te kijken welke intent hier dan exact bij hoort en dialogflow zal niet suggereren. Er zijn hier ook geen percentages, dus je kan niet zien waaraan dialogflow een bepaalde utterance zou linken die niet direct herkend wordt.



Figuur 3.15: Intent aanmaken Dialogflow

Dialogflow heeft ook een API. Deze kan zogoed als enkel het noodzakelijkste, zoals intent en entities aanmaken en utterances toevoegen en nog enkele zaken. Via de API kunnen de utterances van de gebruikers waarvan dialogflow niet weet bij welke intent die past niet opgevraagd worden. Deze kunnen dus niet worden geïntegreerd in een eigen applicatie om die automatisch bij de juiste intent toe te wijzen. Ook zou hier nog een ander probleem zijn zoals bij IBM Watson Assistant dat er geen percentages worden weergegeven. Moest er zelf een API voor zijn om deze op te halen zouden ze niet automatisch kunnen toegewezen worden door het gebrek aan deze percentages van zekerheid.

3.2.5 Andere frameworks

Verder zijn er nog een aantal frameworks bekeken en vergeleken zoals Amazon Lex, BotKit, BotPress, BotMan, Rasa Stack, SAP Conversational AI, enz. Al snel was er ingezien dat ofwel brachten deze frameworks dezelfde problemen met zich mee zoals de al vergeleken frameworks ofwel hadden deze frameworks geen API die kon worden gebruikt.

3.2.6 Conclusie frameworks

Bij geen enkel van de onderzochte frameworks is het mogelijk om via de API de utterances op te halen waarvan het framework niet zo goed weet of hij ze gelinkt heeft aan de correcte intent. Om dan zo via deze API call al deze utterances op te halen en te verwerken in een eigen applicatie die er dan voor zorgt dat deze automatisch worden gecontroleerd en gelinkt worden aan de juiste intent enkel als de zekerheid groot genoeg is. Als de zekerheid niet

goed is, met andere woorden als de applicatie twijfelt tussen 2 intents omdat de percentages van deze veel te dicht bij elkaar liggen zal er niets gedaan worden en zal deze handmatig moeten worden gecontroleerd. Eénmaal gecontroleerd zou de utterance via een andere API call kunnen toegevoegd worden aan de juiste intent in het framework zelf. Zo zouden maar een beperkt aantal van de utterances handmatig moeten gecontroleerd worden doordat er al veel automatisch worden verwerkt.

Dit alles is niet mogelijk door de beperkingen van de API's van deze frameworks of doordat er geen scores (percentages) worden meegeleverd met de API call.

Er is echter wel een manier om dit alles mogelijk te maken door het op een andere manier aan te pakken. Namelijk door alle input van gebruikers op te vangen in een eigen applicatie, deze input daarna door te sturen naar de bot zelf en de resultaten opvangen. Hierin krijg je te zien welke intents er aan deze utterance gelinkt worden met een percentage erbij. Bijvoorbeeld als er 'Hello' wordt gestuurd zullen er een paar intents worden teruggegeven. Hier bijvoorbeeld 'Greeting' met een percentage van 92 percent en 'Curse' met een percentage van 3 percent. Deze percentage verschillen heel veel. Op deze manier weet de applicatie dan direct dat hello bij de intent 'Greeting' hoort en zal deze utterance toevoegen aan deze intent. Indien de applicatie niet zeker is en de 2 percentages liggen te dicht bij elkaar of zijn beide veel te laag zal deze niet toegevoegd worden en zal deze utterance automatisch worden toegevoegd aan de lijst van de utterances waarvan de bot niet zeker is.

Het werken op deze manier zal uiteindelijk dezelfde resultaten bekomen. De utterances die dan worden geanalyseerd en uiteindelijk worden toegevoegd zullen niet meer in de lijst komen van utterances waarvan de bot niet zeker is. Dat is uiteindelijk de bedoeling van het uitwerken van de onderzoeksvraag. De input van de gebruiker zal gedeeltelijk automatisch worden toegevoegd aan de juiste intent maar niet allemaal. Er zijn altijd utterances waarvan de chatbot geen goede suggesties kan weergeven en die zullen dan handmatig moeten worden toegevoegd. Hoe meer utterances per intent, hoe beter die suggesties zullen zijn. Dus hoe langer de chatbot in productie zou zijn, hoe minder utterances er handmatig zouden moeten worden gecontroleerd.

4. Proof of concept

In dit onderdeel zal de proof of concept besproken worden. In deze proof of concept wordt er gebruik gemaakt van de Microsoft bot framework SDK en van LUIS.

4.1 Gekozen framework

Uit het onderzoek is gebleken dat LUIS de juiste keuze is uit alle vergeleken frameworks voor het maken van bots. Het is belangrijk dat de API een goede toegankelijkheid heeft en dat er per weergegeven utterance informatie is over de verschillende intents en daarbij een percentage.

Door de goede toegankelijkheid van de API is er een mogelijkheid om de interface van LUIS zelf achterwege te laten en LUIS bijna volledig in een eigen applicatie kan implementeren. Zoals het toevoegen van intents, entities, utterances. Het enige dat niet kan worden opgehaald zijn de 'review endpoint utterances'. Maar juist door de goede toegankelijkheid kan dit op een andere manier worden opgelost, besproken in de conclusie van vorige hoofdstuk.

Doordat LUIS bij elke utterance veel info teruggeeft over de intents waarbij deze kunnen behoren met daarbij dan nog eens de percentages van zekerheid dat een utterance bij een intent hoort kan er worden gekeken hoezeer LUIS zeker is dat een utterance bij die bepaalde intent hoort.

Dat zijn de belangrijkste zaken waarmee er rekening moet gehouden worden voor de onderzoeksvraag te kunnen uitwerken. In dit geval is er één van de frameworks daartoe in staat om deze zaken te garanderen. Wit.ai kwam in de buurt maar wit voorziet niet genoeg

informatie bij elke utterance. Er wordt wel gebruik gemaakt van scores hier maar bij elke utterance is er maar 1 intent met score voorzien in vergelijking met LUIS waarbij je alle intents ziet met de score daarbij.

Ook komt dit goed uit dat LUIS wordt gekozen als framework dat dit kan realiseren omdat de opdrachtgever Endare, zelf ook gebruik maakt van LUIS voor het ontwikkelen van chatbots en zijn daarmee het meest vertrouwd.

4.2 Opzetten LUIS

Voor de opzet van de LUIS intents en utterances is er mij een voorbeeld toegereikt van de opdrachtgever Endare, zodat ik mijn model niet van nul zou moeten beginnen. Vanaf dit voorbeeld kan er verder gewerkt worden door nieuwe intents toe te voegen zodat deze alsnog beter wordt.

LUIS zal hier worden getraind als een soort van hulp bij de aankoop van producten en onderhoud en weergave van het account en winkelmandje. Zo kan er bijvoorbeeld producten worden toegevoegd aan het mandje, producten kunnen verwijderen, producten zoeken, hulp invoeren, het mandje tonen en verwijderen, het profiel tonen en aanpassen, enz.

De intents die zijn toegevoegd zijn: AddMore, ClearBasket, Curse, DeleteItem, FindItem, Greeting, Help, Joke, RemoveMore, ShowBasket, ShowProfile, Stop en UpdateProfile. Bij elke intent zijn er ongeveer 5 tot 10 utterances toegevoegd wat al een goed begin is voor het correct trainen van de bot.

De entiteiten die zich in LUIS bevinden zijn name, number en product waarvan number een ingebouwde entiteit is en de andere twee zelf gemaakte entiteiten zijn. De naam dient als aanspreking bij de 'Greeting' intent. De entiteit 'number' is voor een hoeveelheid mee te geven bij bijvoorbeeld de intent 'AddMore', zodat er meerdere producten kunnen toegevoegd worden. De entiteit 'product' is voor te weten over welk product er gesproken wordt.

LUIS is volledig in het engels geschreven omdat engels de taal is die het beste wordt begrepen door LUIS. Dit zal dan ook resulteren in betere resultaten als een gebruiker praat tegen de bot. Indien de bot nederlands moet verstaan kunnen er twee dingen gebeuren. Ofwel maak je een andere LUIS applicatie die volledig in het nederlands ofwel steek je er een middleware tussen die de nederlandse tekst omzet naar het engels zodat de engelse versie van LUIS alles verstaat. De beste en gemakkelijkste optie is het tweede omdat er ten eerste minder werk zal zijn. Er zal maar 1 LUIS app moeten gemaakt worden, dus enkel in het engels. Ook zal luis beter werken omdat hij engels beter verstaat dan het nederlands.

4.3 Software

4.3.1 Node.js en TypeScript

De keuze om de applicatie te ontwikkelen met Node.js in TypeScript was niet zo moeilijk. Omdat er bij Endare, het bedrijf die met de onderzoeksvraag is gekomen wordt gewerkt met Node.js is de keuze snel gemaakt om het met Node.js te maken. Door dit te doen met de gebruikelijke tools van dat bedrijf zorgt er voor eventueel latere implementatie een gebruiksgemak doordat de taal hetzelfde is en niet zal moeten worden omgezet. Daardoor ook de keuze om het in TypeScript te maken in de plaats van JavaScript omdat Endare is overgestapt naar TypeScript.

4.3.2 Bot Framework Emulator

Dit is een applicatie gemaakt voor ontwikkelaars voor het te kunnen testen van een chatbot. Met de emulator kan er met de bot gepraat worden via de user interface. Er is ook een inspector en een log waar de verzonden en ontvangen berichten verder in detail kunnen bekeken worden.

Het is zeer eenvoudig om met de bot te verbinden. Als deze lokaal aan het runnen is op je eigen applicatie kan er geconnecteerd worden door enkel het localhost adres in te geven. Daarna is de emulator klaar om met de bot te praten.

4.3.3 Microsoft Bot Framework

Voor het uitwerken van de proof of concept is ervoor gekozen om het Bot Framework SDK te gebruiken van Microsoft. Dit framework is veelgebruikt bij het ontwikkelen van chatbots omdat deze zeer veel mogelijkheden heeft voor het maken van bots. Deze kan gebruikt worden voor zowel simpele console applicaties tot zeer uitgebreide interfaces. Het Bot Framework SDK gebruikt restify, dat is een bekend framework voor het maken van webservices voor het maken van de web server van de bot.

Dit SDK heeft een systeem voor het maken van dialogs, er zijn veel ingebouwde zaken zoals kaarten met daarin knoppen, foto's, enz. Wat heel belangrijk is hier is dat er een mogelijkheid is voor het implementeren van AI frameworks. Wat hier zeker een noodzaak is en gebruik van zal gemaakt worden door LUIS te implementeren in het bot framework SDK.

Het installeren van de bot framework SDK (botbuilder) kan eenvoudig worden gedaan door simpelweg een commando in te geven. Daarna kan je onmiddellijk aan de slag met de botbuilder.

Eerst en vooral dient er een HTTP server gecreëerd te worden. Hierop wordt de bot lokaal gehost zodat deze kan getest worden. De bot kan getest worden gebruik makende van de Bot Framework Emulator als deze HTTP server draait.

```
1 // create HTTP server \\
2 var server = restify.createServer();
3 server.listen(process.env.port || process.env.PORT || 3978, () => {
4   console.log('Listening... ${server.name}... ${server.url}');
5 });
```

Listing 4.1: creatie van de HTTP server

Met de HTTP server alleen is er nog geen connectie met de botbuilder. Een connectie maken kan ook eenvoudig in enkele lijnen code. Daarna kun je LUIS koppelen aan deze connectie. Dit doe je aan de hand van het endpoint dat je krijgt wanneer je LUIS deployed op de site van LUIS zelf. Dan krijg je een link naar het endpoint. Deze link steek je best in de environment variables zodat deze veilig blijft naar de buitenwereld toe. Daarna moet kunnen geluisterd worden naar inkomende boodschappen die je bijvoorbeeld ingeeft in de bot framework emulator.

```
1 // connection \\
2 var conn = new builder.ChatConnector({
3   appId: process.env.MICROSOFT_APP_ID,
4   appPassword: process.env.MICROSOFT_APP_PASSWORD
5 });
6
7 // link LUIS to botbuilder \\
8 var bot = new builder.UniversalBot(conn);
9 bot.recognizer(new builder.LuisRecognizer(process.env.
10   LUIS_MODEL_URL));
11
12 // listen for incoming requests \\
13 server.post("/api/messages", conn.listen());
```

Listing 4.2: connectie en linken van LUIS

Als al deze stappen gedaan zijn, is alles in gereedheid gebracht voor het kunnen verwerken van een bepaalde input. Dit wordt gedaan door dialogs. Aan de hand van de intent die je terug krijgt van luis voor een bepaalde input van een gebruiker, wordt telkens een andere dialog aangeroepen zodat de juiste actie kan worden ondernomen en het juiste resultaat kan teruggegeven worden.

Een dialog werk volgens de waterval methode. Dit betekent dat een dialog van functie naar functie kan gaan. Dit is zodat als eerst dit stukje code en dan dat stukje code als dat eerst gedaan is en zo voort. Zoals er kan gezien worden in Listing 4.3 wordt er eerst gevraagd naar de naam van de gebruiker. Als de gebruiker deze naam heeft ingegeven gaat er via de waterval methode naar de volgende functie gegaan worden waarbij de naam van de gebruiker wordt opgeslagen zodat de bot de naam blijft onthouden voor in de toekomst en wordt er een antwoord teruggestuurd naar de gebruiker.

Dit is een eenvoudig voorbeeld, er kan hierin veel verder gegaan worden, er kan bijvoorbeeld in een dialog een andere dialog aangeroepen worden. Als de naam bijvoorbeeld nog niet geweten is kan er een dialog worden aangeroepen voor de naam op te vragen maar indien deze wel al geweten is kan er gewoon verder worden gegaan met de volgende stap in de waterval.


```

1 // dialog \
2 bot.dialog("/", [
3     function (sess, args, next) {
4         if (!sess.userData.name) {
5             builder.Prompts.text(sess, "Hello, user! What is your name?")
6             ;
7         }
8         else {
9             next();
10        }
11    },
12    function (sess, result) {
13        sess.userData.name = result.response;
14        sess.send("Hello, " + sess.userData.name);
15    }
16 ]);

```

Listing 4.3: Waterval methode

Dit is zowat de basis van de werking van de botbuilder. Er zijn hier dus nog veel meer mogelijkheden maar die liggen uit de scope van deze bachelorproef. Met deze basis kan er begonnen worden aan de uitwerking van de proof of concept.

4.4 Uitwerking

Een eerste stap was het werkende krijgen van de botbuilder en daarna het linken van LUIS hieraan. Hoe LUIS wordt gelinkt met de botbuilder is uitgelegd in het vorige hoofdstuk. Hoe de verschillende intents van LUIS gelinkt zijn met de dialog zit nog een beetje anders in mekaar dan een simpele dialog voor het vragen en opslaan van een naam. per intent is er één dialog nodig en deze moet hieraan worden gelinkt. De code in Listing 4.4 is een basis dialog voor de intent 'AddMore'. Als deze intent wordt aangeroepen zal de gebruiker een bericht terug krijgen met de boodschap 'You reached AddMore intent'. Op deze manier kan er al gecontroleerd worden of de intents van LUIS juist gelinkt worden met de dialogs. Als er in de emulator bijvoorbeeld een zin wordt ingegeven 'Please add 5 more', dan zal de botbuilder het AddMore intent terug krijgen en zal deze willen linken aan de juiste dialog. Dit gebeurt door de 'triggerAction' te matchen aan de intent. De triggerAction zal bij dit voorbeeld zijn 'AddMore' en het antwoord van LUIS is ook 'AddMore'. Op deze manier zal de botbuilder kijken of dit ergens gelijk is, hier is dit het geval en zal deze bepaalde dialog aanroepen en de gepaste functies uitvoeren en uiteindelijk een boodschap terugsturen naar de gebruiker.

```

1 // dialog AddMore intent \
2 bot.dialog("/addMore", (sess, args) => {
3     sess.send('You reached AddMore intent');
4     }).triggerAction({
5         matches: "AddMore"
6     });

```

Listing 4.4: Waterval methode

De intent zelf en de entities kunnen makkelijk worden verkregen uit de args van de dialog. Bijvoorbeeld voor de volgende zin 'You need the AddMore met entity 5' als je ingeeft als input 'Add 5 more' te verkrijgen kan je simpelweg volgende lijn code gebruiken.

```
1 sess.send('You need the "${args.intent.intent}" intent ${sess.  
    dialogData.number === null ? '' : 'met entity ' + sess.  
    dialogData.number.entity}');
```

Listing 4.5: gebruiken van args

De 'sess.dialogData.number' komt uit de lokale opslag. Door elke intent te analyseren door de code in Listing 4.6 worden de entities uit de utterance gehaald en opgeslagen per dialog zodat deze gemakkelijk toegankelijk zijn voor verder te kunnen werken om uiteindelijk feedback te kunnen geven naar de gebruiker toe.

```
1 function normalizeAndStoreData(sess: builder.Session, args: any):  
    void {  
2     sess.dialogData.name = builder.EntityRecognizer.findEntity(args.  
        intent.entities, "Name");  
3     sess.dialogData.product = builder.EntityRecognizer.findEntity(  
        args.intent.entities, "Product");  
4     sess.dialogData.number = builder.EntityRecognizer.findEntity(args.  
        intent.entities, "builtin.number");  
5 }
```

Listing 4.6: Analyseren van de intent

Als alles correct werkt, LUIS die de input van de gebruiker aan de correcte intent linkt, de botbuilder die de intent van LUIS terugkrijgt die bij deze input past en dan de juiste dialog zal aanspreken met behulp van de intent van LUIS, de entities hier correct uithalen en een gepaste boodschap terugsturen naar de gebruiker zal er uiteindelijk kunnen overgegaan worden naar de kern voor de zaak. Dat is het automatisch aanleren van LUIS aan de hand van de input van de gebruikers zodat de 'Review endpoint utterances' op de site van LUIS zelf zo leeg mogelijk blijft.

Dit zal gebeuren door de boodschap zelf en de intents in de dialog door te geven naar een functie die zal controleren of deze bepaalde input kan worden toegevoegd in LUIS. Dat zal worden gedaan door de eerste en de tweede intent die LUIS meegeeft met een bepaalde input te controleren. Indien de eerste en de tweede intent zeer sterk verschillen van elkaar en als het percentage van de eerste intent hoog genoeg is zal deze input worden toegevoegd als utterance bij deze eerste intent.

Door dat te doen zullen voor van de inputs niet meer bij de 'review endpoint utterances' komen en zullen er veel minder utterances moeten gecontroleerd worden, wat uiteindelijk het doel is van dit onderzoek.

Deze functie controleert eerste de scores van de intents met elkaar en kijkt of deze voldoet aan de voorwaarden. De parameters kunnen altijd nog aangepast worden, deze worden onderzocht en geoptimaliseerd in het volgende hoofdstuk. Indien de voorwaarde voor het toevoegen van de input aan LUIS niet voldoen zal er niets gebeuren. Daardoor zal LUIS

deze input toevoegen aan de 'Review endpoint utterances' tabblad zodat deze manueel kan bekeken worden. Als deze wel voldoet zal de API van LUIS worden aangesproken voor het toevoegen van deze input in LUIS.

De code voor het aanspreken van het endpoint bevindt zich in Listing 4.7. De body die de API verwacht moet in een bepaald formaat staan zodat deze begrepen wordt en de API deze kan verwerken. Eerst en vooral wordt dan het endpoint aangesproken van de API call met de juiste link. Met dit endpoint moeten een aantal parameters worden meegegeven. Eerst en vooral de methode, hierbij is dat een POST, er wordt data meegestuurd. Ook zijn er headers nodig. De eerste header is voor het formaat aan te duiden, hier is dat json. De tweede header is de subscription key voor de authenticatie. Als laatste en belangrijkste moet er een body worden meegegeven. Die body bevat een text, dat is de utterance zelf. De intentName is de intent waarbij deze utterance hoort, deze intentName wordt uit de intent gehaald die wordt meegegeven naar de functie. Als laatste kunnen er entities worden meegegeven die horen bij de intent met de locatie waar deze zich bevinden in de utterance. Dit is niet noodzakelijk want LUIS zoekt vanzelf de entities in een zin dus dit kan eventueel achterwege gelaten worden.

```
1 fetch('https://westus.api.cognitive.microsoft.com/luis/api/v2.0/
  apps/3480e277-67b8-4cb9-af10-f7db6ce55d63/versions/0.1/example',
  {
2   method: 'POST',
3   body: JSON.stringify({
4     "text": text,
5     "intentName": intent.intents[0].intent,
6     "entityLabels": [
7       getEntities(intent.entities),
8     ]
9   }),
10  headers: {
11    'Content-type': 'application/json',
12    'Ocp-Apim-Subscription-Key': key
13  }
14 })
15 .then((res: Response) => {
16   return res.json();
17 })
18 .then((json) => {
19   console.log('Result fetch succes', json);
20 })
21 .catch(() => console.log('Calling POST example has crashed'));
```

Listing 4.7: Toevoegen van een input aan LUIS van een API call

Dit is zowat het grootste werk voor het realiseren en uitwerken van de onderzoeksvraag. Hoe efficient dit alles is zal worden getest in het volgende hoofdstuk. Er zullen een aantal zinnen opgesteld worden die zullen worden meegegeven met LUIS zonder en met dit uitgewerkte systeem en zo zal er gekeken worden hoe efficient dit systeem is. Ook zal er gekeken worden naar de effectieve utterances die bij 'Review endpoint utterances' komen met en zonder het uitgewerkte systeem. Het optimaliseren van dit systeem zal zich ook in dit hoofdstuk bevinden, dit zal gebeuren door bepaalde parameters aan te passen zodat er

een optimaal resultaat zal bekomen worden.

5. Conclusie

Curabitur nunc magna, posuere eget, venenatis eu, vehicula ac, velit. Aenean ornare, massa a accumsan pulvinar, quam lorem laoreet purus, eu sodales magna risus molestie lorem. Nunc erat velit, hendrerit quis, malesuada ut, aliquam vitae, wisi. Sed posuere. Suspendisse ipsum arcu, scelerisque nec, aliquam eu, molestie tincidunt, justo. Phasellus iaculis. Sed posuere lorem non ipsum. Pellentesque dapibus. Suspendisse quam libero, laoreet a, tincidunt eget, consequat at, est. Nullam ut lectus non enim consequat facilisis. Mauris leo. Quisque pede ligula, auctor vel, pellentesque vel, posuere id, turpis. Cras ipsum sem, cursus et, facilisis ut, tempus euismod, quam. Suspendisse tristique dolor eu orci. Mauris mattis. Aenean semper. Vivamus tortor magna, facilisis id, varius mattis, hendrerit in, justo. Integer purus.

Vivamus adipiscing. Curabitur imperdiet tempus turpis. Vivamus sapien dolor, congue venenatis, euismod eget, porta rhoncus, magna. Proin condimentum pretium enim. Fusce fringilla, libero et venenatis facilisis, eros enim cursus arcu, vitae facilisis odio augue vitae orci. Aliquam varius nibh ut odio. Sed condimentum condimentum nunc. Pellentesque eget massa. Pellentesque quis mauris. Donec ut ligula ac pede pulvinar lobortis. Pellentesque euismod. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent elit. Ut laoreet ornare est. Phasellus gravida vulputate nulla. Donec sit amet arcu ut sem tempor malesuada. Praesent hendrerit augue in urna. Proin enim ante, ornare vel, consequat ut, blandit in, justo. Donec felis elit, dignissim sed, sagittis ut, ullamcorper a, nulla. Aenean pharetra vulputate odio.

Quisque enim. Proin velit neque, tristique eu, eleifend eget, vestibulum nec, lacus. Vivamus odio. Duis odio urna, vehicula in, elementum aliquam, aliquet laoreet, tellus. Sed velit. Sed vel mi ac elit aliquet interdum. Etiam sapien neque, convallis et, aliquet vel, auctor non, arcu. Aliquam suscipit aliquam lectus. Proin tincidunt magna sed wisi. Integer blandit

lacus ut lorem. Sed luctus justo sed enim.

Morbi malesuada hendrerit dui. Nunc mauris leo, dapibus sit amet, vestibulum et, commodo id, est. Pellentesque purus. Pellentesque tristique, nunc ac pulvinar adipiscing, justo eros consequat lectus, sit amet posuere lectus neque vel augue. Cras consectetur libero ac eros. Ut eget massa. Fusce sit amet enim eleifend sem dictum auctor. In eget risus luctus wisi convallis pulvinar. Vivamus sapien risus, tempor in, viverra in, aliquet pellentesque, eros. Aliquam euismod libero a sem.

Nunc velit augue, scelerisque dignissim, lobortis et, aliquam in, risus. In eu eros. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur vulputate elit viverra augue. Mauris fringilla, tortor sit amet malesuada mollis, sapien mi dapibus odio, ac imperdiet ligula enim eget nisl. Quisque vitae pede a pede aliquet suscipit. Phasellus tellus pede, viverra vestibulum, gravida id, laoreet in, justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer commodo luctus lectus. Mauris justo. Duis varius eros. Sed quam. Cras lacus eros, rutrum eget, varius quis, convallis iaculis, velit. Mauris imperdiet, metus at tristique venenatis, purus neque pellentesque mauris, a ultrices elit lacus nec tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent malesuada. Nam lacus lectus, auctor sit amet, malesuada vel, elementum eget, metus. Duis neque pede, facilisis eget, egestas elementum, nonummy id, neque.

A. Onderzoeksvoorstel

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

A.1 Introductie

Door het massale gebruik van chats en chatbots op mobiele applicaties en websites hebben we steeds meer nood aan chatbots die automatisch kunnen bijleren. Doordat ordinare chats en het aanleren/bijleren van chatbots zeer veel tijd in beslag nemen zou het automatiseren een ideale oplossing zijn. Aan de hand van de volgende onderzoeksvraag gaan we antwoorden zoeken:

- Kunnen we een chatbot automatisch laten bijleren (zijnde de NLP bijsturen) op basis van het gedrag van gebruikers?

A.2 State-of-the-art

A.2.1 Chatbots

Ruim 90 procent van de chatbots is nog zonder AI. Bij chatbots met Ai wordt er gebruik gemaakt van een NLP programma, waarmee de chatbot kan getraind worden.

A.2.2 NLP (Natural language processing)

Bij NLP wordt de intentie van een zin achterhaald en daar wordt de relevante informatie uit gehaald.

Een paar de de bekendste NLP platformen zijn wit.ai en LUIS.ai. In elk van die platformen kan je je intents manueel trainen door zelf voorbeeldzinnen toe te voegen of door antwoorden van gebruikers te linken aan de juiste intent en zo dus extra voorbeeldzinnen als input te gebruiken.

A.2.3 Intents

Simpel gezegd, Intents zijn de intenties van de eindgebruiker, wat is het doel van het bericht van die gebruiker. Deze intents worden door de gebruiker naar de chatbot overgebracht. Je kunt je intenties voornamelijk in twee categorieën indelen

- Casual intents
- Bedrijfsintenties

Casual intents is zoals "small talk", het openen of sluiten van een gesprek en positieve of negatieve bedoelingen. Bedrijfsintenties zijn de intenties die verwijzen naar het specifieke doel van die chatbot.

A.2.4 Entiteiten

Entiteiten zijn gegevens uit een bericht zoals naam of e-mailadres, deze gegevens worden door de bot uit de input van de gebruiker gehaald.

Het is belangrijk om de juiste entiteiten uit de input te halen en deze te koppelen aan de intent.

A.2.5 Trainen

Hoe meer variantie van input die je ontvangt uit echte gesprekken, hoe beter je je NLP-service kunt trainen voor intenties. Het proces van het opnieuw trainen van het systeem moet doorgaan totdat het foutenpercentage vermindert. Dit doe je wanneer je supplementaire input krijgt.

A.2.6 Feedback loops

Met feedback loops wordt bedoeld om het mogelijk te maken om een nieuwe input van een gebruiker, die de chatbot nog niet herkent als een juiste intent, toch automatisch te gaan gebruiken.

A.3 Methodologie

In dit werk zal er onderzocht worden hoe men de chatbots gedeeltelijk kan automatiseren met behulp van een NLP platform. Ook zal onderzocht worden welk platform het meest geschikt zal zijn hiervoor. Er zal worden gekeken naar de toegankelijkheid van de API bij deze platformen. Een goede toehankelijkheid is essentieel voor dit makkelijk te kunnen onderzoeken en implementeren. Normaal moeten de inputs manueel gelinkt worden aan de juiste intent. Dit komt doordat het systeem niet 100 procent zeker kan weten of hij ze onder de juiste intent heeft geplaatst. Die manuele controle zouden we dan achterwege proberen laten. Dit zou kunnen gedaan worden door een threshold in te stellen. Wanneer de chatbot boven een bepaald percentage denkt dat de input bij een bepaalde intent past deze te zien als correct. Daardoor moet deze niet meer worden gecontroleerd door een persoon.

Er zullen ook testomgevingen worden opgezet. De verschillende NLP as a service platformen zullen worden vergeleken, zoals wit.ai, LUIS.ai,... om te experimenteren en de onderzoeksvraag te evalueren en op te lossen. Bij deze platformen zal gekeken worden waarbij de API het meest toegankelijk is voor eventueel te kunnen automatiseren.

A.4 Verwachte resultaten

Er zal een platform zijn die het meeste geschikt zal zijn om dit te onderzoeken. Deze zal merkbaar verschillen met andere platformen. Er wordt echter wel verwacht dat er een systeem wordt gevonden op een manier waarbij de input wordt gelinkt aan de juiste intent en daar de juiste entiteiten uithaalt.

A.5 Verwachte conclusies

De verwachtingen die uit dit onderzoek zullen blijken is dat er een methode bestaat waarbij men de input van de gebruiker automatisch kan linken aan een intent zonder manuele controle. Het streefdoel zou zijn om 10 tot 30 procent van ongekende inputs van een chatbot automatisch te laten koppelen aan de juiste intent.

Jepma, 2018 Singh, 2017 „What is Language Understanding (LUIS)?“, 2019 Giulia, 2017

Bibliografie

- Berry, D. (2019). What is Language Understanding (LUIS)? *microsoft azure*. Verkregen van <https://docs.microsoft.com/nl-be/azure/cognitive-services/luis/what-is-luis>
- Dimitrova, E. (2016). Bot Wars: Wit.ai vs Chatfuel. *chatbots magazine*. Verkregen van <https://chatbotsmagazine.com/bot-wars-wit-ai-vs-chatfuel-5e978070b9e>
- Doijer, D. (2018). Chatbots zijn de trend: wat weet jij al over deze bots? *kgom*. Verkregen van <https://kgom.nl/24/chatbots-zijn-de-trend-wat-weet-jij-al-over-deze-bots/>
- Elovic, A. (2017). Chatbots the beginners guide. *chatbots magazine*. Verkregen van <https://chatbotsmagazine.com/chatbots-the-beginners-guide-618e72599b55>
- Giulia. (2017). Wit.ai and how to use it. *medium*. Verkregen van <https://medium.com/@Giuul/wit-ai-and-how-to-use-it-72372b07d98b>
- Hsu, K. (2017). ChatWeb: Build Websites that understand users with full (free) Microsoft Stack. *hackernoon*. Verkregen van <https://hackernoon.com/chatweb-build-websites-that-understand-users-with-full-free-microsoft-stack-dc07ce18b19d>
- Jepma, L. (2018). In 7 stappen aan de slag met chatbots. *frankwatching*. Verkregen van <https://www.frankwatching.com/archive/2018/01/03/in-7-stappen-aan-de-slag-met-chatbots/>
- Rouse, M. (2018). natural language understanding (NLU). *Search Enterprise AI*. Verkregen van <https://searchenterpriseai.techtarget.com/definition/natural-language-understanding-NLU>
- Shridhar, K. (2017). Rule bases bots vs AI bots. *medium*. Verkregen van <https://medium.com/botsupply/rule-based-bots-vs-ai-bots-b60cdb786ffa>
- Singh, B. R. (2017, januari 29). Chat Bots Designing Intents and Entities for your NLP Models. *Designing Intents and Entities for your NLP Models*. Verkregen van <https://medium.com/@brijrajsingh/chat-bots-designing-intents-and-entities-for-your-nlp-models-35c385b7730d>

- van Tuil, K. (2011). Wat is een API? *computerworld*. Verkregen van <https://computerworld.nl/development/74796-wat-is-een-api>
- What is Language Understanding (LUIS)? (2019). *microsoft*. Verkregen van <https://docs.microsoft.com/en-us/azure/cognitive-services/luis/what-is-luis>
- Wit.ai. (2019). Natural Language for Developers. *wit.ai*. Verkregen van <https://wit.ai>