HO
GENT

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

Het onderwerp van deze bachelorproef kwam ter sprake met nog een aantal andere onderwerpen die Endare, mijn stagebedrijf wou onderzoeken. Dit onderwerp gaande over chatbots sprak mij onmiddellijk aan door de enorme groei van AI en Machine learning. Ook chatbots is een onderdeel hiervan, dus leek het mij interessant om hier dieper op in te gaan en dit te onderzoeken. Chatbots zullen meer en meer worden gebruikt in de toekomst. De ontwikkeling van chatbots met AI staan nog niet zo super ver en daardoor wou ik mij daar ook wat meer in verdiepen en een meerwaarde bieden in dit domein.

Ik zou graag de personen willen bedanken die mij hebben geholpen bij het realiseren van deze bachelorproef. Allereerst mijn promotor Leen Vuyge voor mij zo goed te begeleiden tijdens dit onderzoek en tijd vrij te maken om samen te zitten. Ook voor alle duidelijke uitleg op de vragen die ik had

Ook zou ik mijn co-promoter Sander Goossens willen bedanken voor de hulp en de verduidelijking van dit onderzoek. Hij heeft mij geholpen met het zoeken van een onderwerp, het verduidelijken van vragen en problemen die ik had. Ook voor de tijd om samen te denken voor een oplossing op de onderzoeksvraag.

Als laatste zou ik mijn ouders en grootvader willen bedanken voor het nalezen en controleren van deze bachelorproef. Ook voor de grote steun tijdens het schrijven van deze.

Samenvatting

Dit werk is belangrijk voor het makkelijker maken van de ontwikkeling van een chatbot. Grotendeels zal dit onderzocht worden voor het besparen van tijd en het beter maken van de accuraatheid van deze bot.

Omdat het ontwikkelen van een chatbot zeer veel tijd in beslag neemt, vooral het op punt stellen van de accuraatheid van de bot. Er zal dus een efficiëntere manier gezocht worden om deze bot dingen aan te leren.

Er is een proof of concept werkende met LUIS opgesteld die de inputs van de gebruiker opvangt, deze controleerd aan de hand van de info die LUIS voorziet per input. Als de input na deze controle is goedgekeurd zal deze input automatisch worden gelinkt aan de juiste intent en zal de manuele controle achterwege kunnen worden gelaten.

In dit document bevindt zich een onderzoek van een aantal van de bekendste NLP-frameworks. Er wordt onderzocht hoe deze precies werken en of de API toegankelijk genoeg is voor het realiseren van deze proof of concept.

Het resultaat van dit onderzoek is positief uitgevallen. Het is effectief gelukt met LUIS om een proof of concept op te stellen die via de API LUIS gedeeltelijk automatisch zal aanleren.

Uit de testen hiervan is gebleken dat maar liefst 50 percent van de inputs niet meer in de 'Review endpoint utterances' tabblad komen maar automarisch worden gelinkt aan de correcte intent.

In een vervolgonderzoek kan dit onderzoek nog wat uitgebreid worden. Er kunnen met meerdere LUIS-apps gewerkt worden, met elk een totaal onder doel en onderwerp. Deze

kunnen getest worden met meerdere en verschillende testdata. Ook kan er een veel ingewikkeldere applicatie getest worden met moeilijkere intent die utterances hebben met veel meer entities en controleren of deze hier ook correct worden gelinkt.

Inhoudsopgave

1 Inleiding 15

1.1 Probleemstelling 15

1.2 Onderzoeksvraag 15

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 33

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 Resultaten 43

5.1 Testdata **43**

5.2 Uitwerking **44**

6 Conclusie 47

A Onderzoeksvoorstel 49

A.1 Introductie **49**

A.2 State-of-the-art **49**

A.2.1 Chatbots 49

A.2.2 NLP (Natural language processing) 50

A.2.3 Intents 50

A.2.4 Entiteiten 50

A.2.5 Trainen 50

A.2.6 Feedback loops 50

A.3 Methodologie **51**

A.4 Verwachte resultaten **51**

A.5 Verwachte conclusies **51**

B Testdata 53

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 28
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 32
3.15 Intent aanmaken Dialogflow 32

5.1 Resultaat batch testen 45

Lijst van tabellen

1. Inleiding

In dit onderdeel zal er al een beter beeld worden geschept over het onderwerp zelf en of dit de moeite waard is om te onderzoeken.

1.1 Probleemstelling

Wanneer een bedrijf zoals Endare, de opdrachtgever van deze onderzoeksvraag een chatbot ontwikkeld voor een klant, zal er veel werkt kruipen in het op punt zetten van deze chatbot. Een deel van dit werk zal pas kunnen gedaan wanneer de chatbot effectief in gebruik wordt genomen. Al de input dat gebruikers ingeven waarvan LUIS zelf maar een beetje aan twijfelt zal moeten manueel gelinkt worden aan de juiste intent. Door een antwoord te zoeken op de onderzoeksvraag zal dit werkt worden verminderd door een gedeelte van deze inputs automatisch te koppelen aan de juiste intent.

1.2 Onderzoeksvraag

De onderzoeksvraag 'Kunnen we een chatbot automatisch laten bijleren op basis van het gedrag van gebruikers?' zal in deze bachelorproef onderzocht worden. Kan hier dus door middel van de input die gebruikers ingeven via de chatbot, deze input automatisch linken aan de correcte intent, dus de bot laten bijleren. Hierbij zal ook onderzocht worden in welke mate dit automatisch zal gebeuren en het werk voor het ontwikkelteam verminderen.

1.3 Onderzoeksdoelstelling

Dit werk zal een succes zijn als de API van minstens één van de onderzochte frameworks toelaat dat een input van een gebruiker kan opgehaald en gecontroleerd worden om dit dan te linken aan de correcte intent. Er zal een proof of concept worden opgesteld om dit te realiseren.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

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

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

In Hoofdstuk 4 wordt de gemaakt proof of concept besproken. Hierin wordt ook gezien hoe het gekozen framework zal worden opgezet.

In Hoofdstuk 5 zal de accuraatheid en de efficiëntie van de proof of concept worden getest.

In Hoofdstuk 6, 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 gebruikersinterface. 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 gebruikersinterfaces 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 machinelearning 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 specialisatiebots 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 er precies wil geweten worden.

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 officepakketten. 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 kan deze makkelijk worden gebruikt. 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 plaatst 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 wordt dan gekoppeld aan de juiste intent. Er kan gemakkelijk gepraat worden met wit via de endpoint API.

Net zoals bij Luis is er een inbox met zinnen die zijn ingegeven door een gebruiker in de chatbot zelf die manueel moeten gelinkt worden aan de juiste intent. Wit stelt zelf al voor bij welke intent hij een bepaalde zin zou plaatsen. Deze kan aangepast worden 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 kan er via het endpoint API van Luis of Wit de juiste intent uit een zin worden gehaald door de API aan te spreken.

Naar mate van toegankelijkheid van de API van deze platformen zal er moeten gekeken worden of het mogelijk is 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.

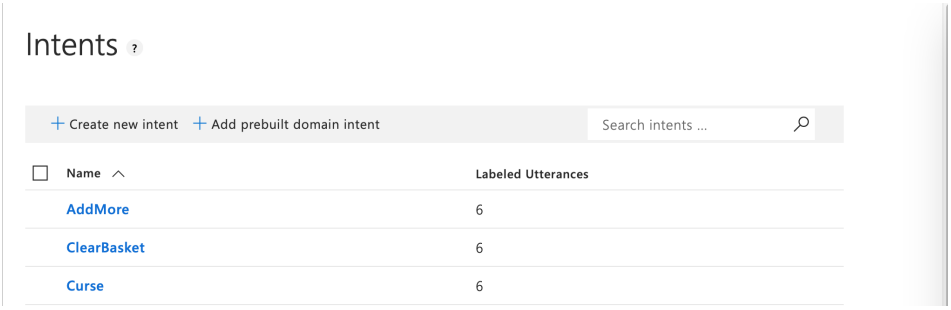
3.1 Mogelijkheden

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 worden deze toegevoegd 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 wordt deze dan gekoppeld aan de juiste intent? Wat kan er nog allemaal met zo een framework gedaan worden? Hoe wordt de de API gebruikt? Op al deze vragen wordt een antwoord gezocht.

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.

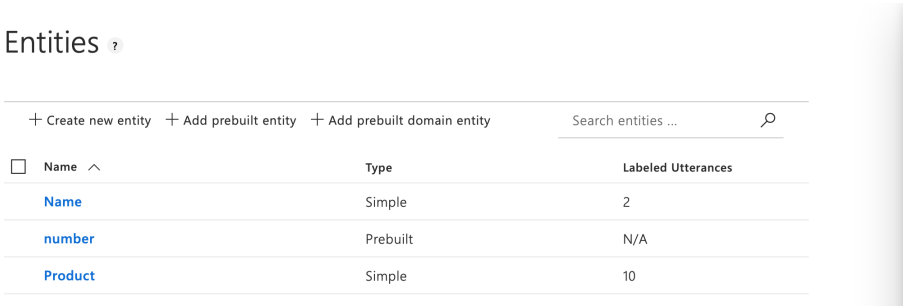
Het trainen van een chatbot via LUIS is zeer gebruiksvriendelijk. Eerst en vooral dient er een nieuwe app te worden aangemaakt. Dit is zeer eenvoudig door de vereiste naam, de taal die er zal gesproken worden tegen de bot en eventueel een beschrijving toe te voegen. Eénmaal de app is gecreëerd dienen er Intents aangemaakt te worden. Een voorbeeld van een intent kan zijn 'Greeting', hierin vallen dus alle begroetingen. Per intent dienen er dan utterances toegevoegd te worden. 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 er toegevoegd worden, 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 voorgaan 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 er zelf een entity aangemaakt worden en deze makkelijk linken aan het juiste woord in een utterance.



Figuur 3.2: Entities aanmaken Luis.ai

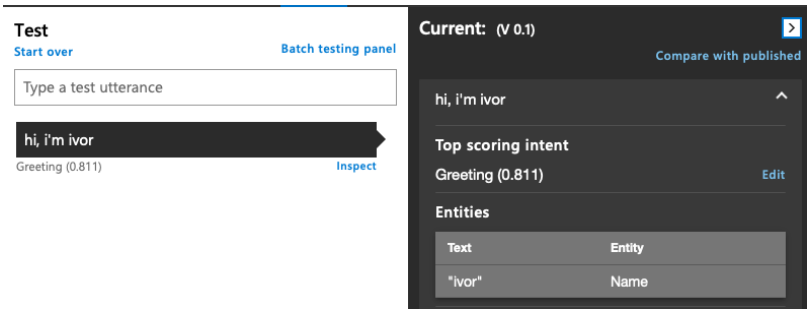
Eénmaal er enkele intents aangemaakt zijn en bij elke intent enkele utterances zijn toegevoegd (het is aangeraden van er minimum 5 toe te voegen voor een betere accuraatheid)

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 kan er gezien worden 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.

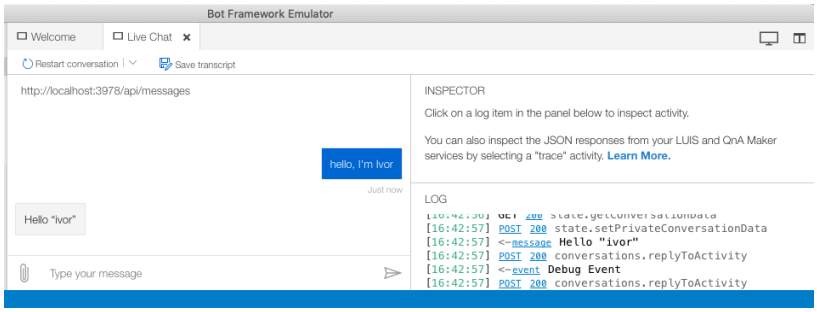


Figuur 3.3: Testen van de chatbot Luis.ai

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

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 er makkelijk gezien worden als er iets is misgelopen.

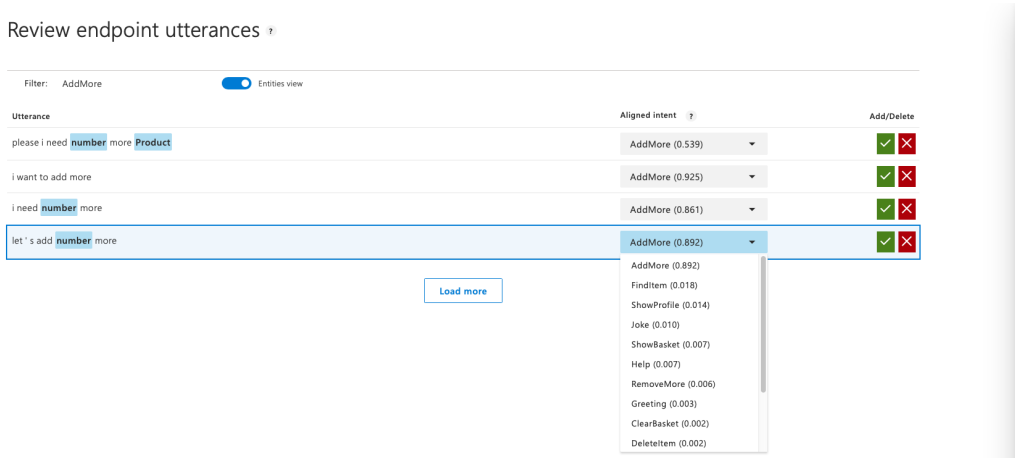
Als de LUIS applicatie gepubliceerd is en deze ontvangt input van gebruikers door bijvoorbeeld via de Bot Framework Emulator, analyseert LUIS elk van deze utterances. LUIS probeert die dan te linken aan de juiste intent. LUIS zal nooit 100 percent zeker zijn dat een bepaalde utterance behoort tot één 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 deze manueel kunnen gelinkt worden aan de juiste intent, utterance per utterance. Bij elke utterance geeft LUIS zelf al een schatting bij welke intent deze hoort. Afbeelding



Figuur 3.4: Bot Framework Emulator

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.

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 FintItem 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, kunnen de entities handmatig aangeduid worden.



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 duidelijker 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.

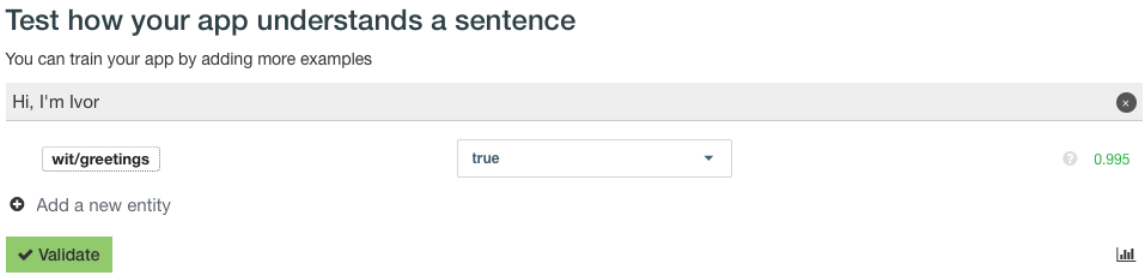
3.2.2 Wit.ai

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.

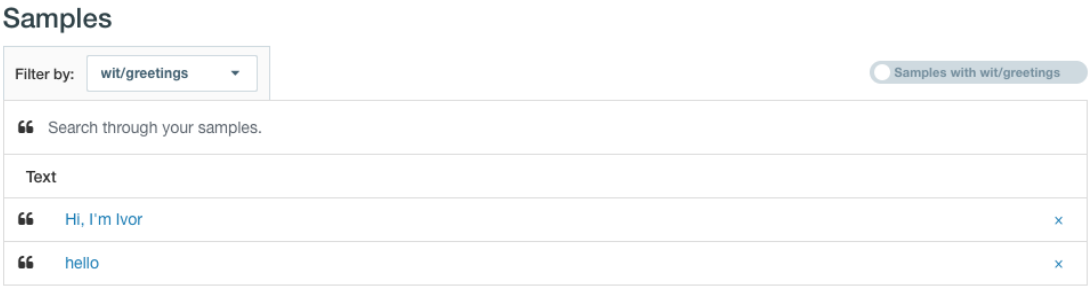
Eerst en vooral moet er een applicatie worden aangemaakt met de naam, de taal waarin de bot zal functioneren en eventueel een omschrijving. Het toevoegen van intents en utterances werkt op een verschillende manier maar komt uiteindelijk op hetzelfde neer. Afbeelding 3.6 toont hoe dit in zijn werk gaat.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

Wit herkent dit al automatisch als een 'Greeting', hier wordt ook het percentage van 99 percent gezien. 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 weergeven van de al ingevoerde utterances verschilt ook van LUIS. In Wit is er een tabblad 'Samples' waarbij alle utterances worden getoond onder elkaar. Hier kunnen de utterances gefilterd worden per intent.



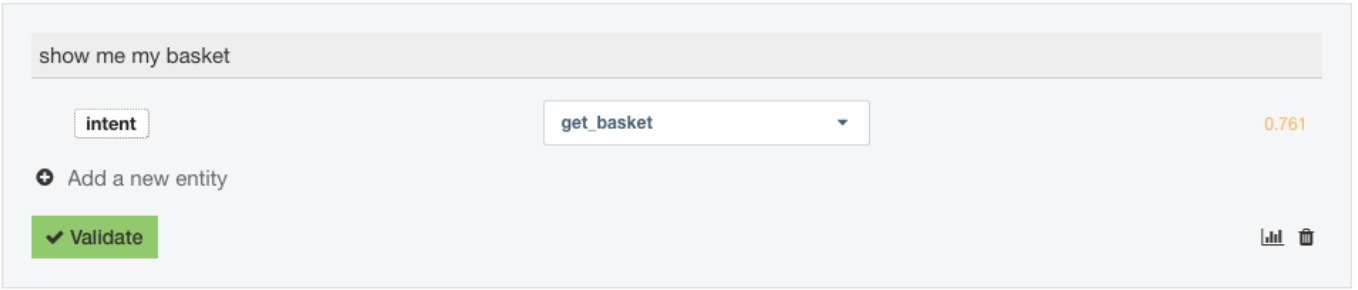
Figuur 3.6: Intent aanmaken Wit.ai



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 kunnen dan ook nagekeken worden of alles klopt. Indien dit niet juist is, kan deze veranderd worden via de dropdown en deze dan gevalideerd worden. 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.



Figuur 3.8: Intent aanmaken Wit.ai

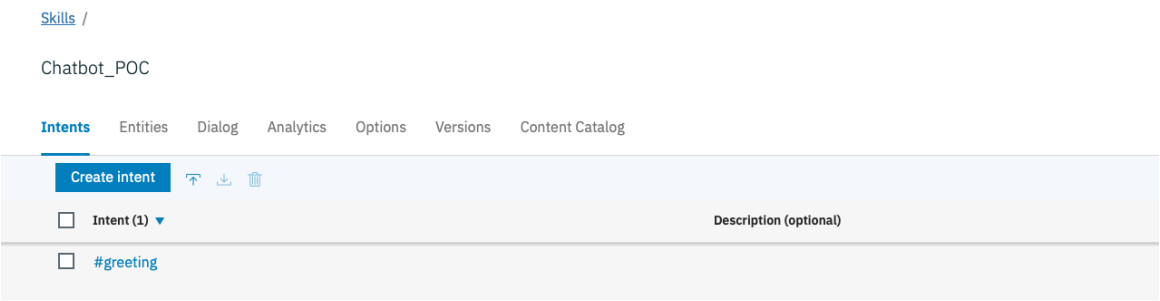
Wit heeft ook wel degelijk een API ter beschikking. De API lijkt zeer sterk op deze van LUIS, er kunnen nieuwe utterances aan worden toegevoegd, samen met nieuwe intent en entities. Ook kunnen hier de utterances worden opgehaald die al gelinkt zijn aan

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

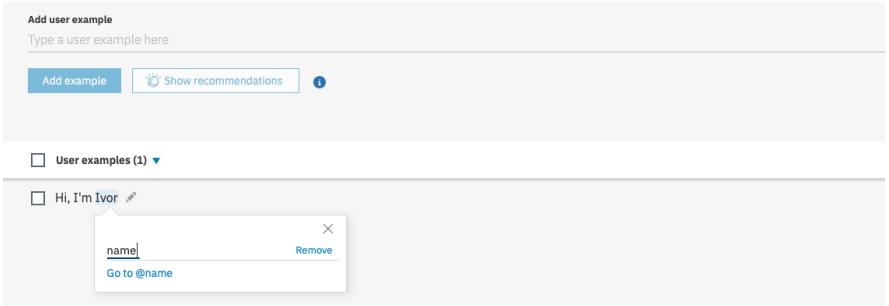
Een derde framework voor het bouwen van chatbots is IBM Watson Assistant. Dit framework werkt ook gedeeltelijk op een andere manier dan de vorige onderzochte frameworks. Voor het ontwikkelen van een chatbot in IMB Watson Assistant zijn er drie grote stappen.

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 eerst moet worden gemaakt bij de vorige frameworks. Eerst en vooral dient een skill te worden aangemaakt. Een skill heeft een naam, de taal waarin tegen de bot zal gesproken worden en eventueel een omschrijving. Eenmaal deze skill is aangemaakt, kunnen er zeer eenvoudig intents worden aangemaakt en aan elke intent utterances worden toegevoegd.



Figuur 3.9: Intent aanmaken IBM Watson Assistant

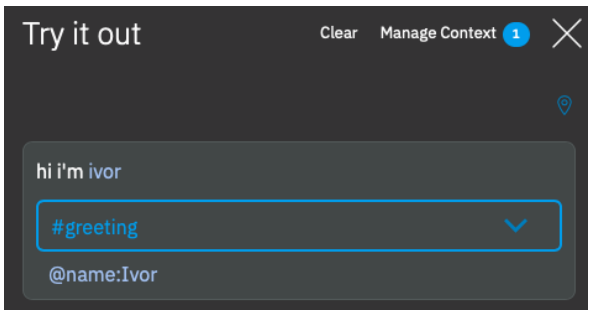
Na het aanmaken van een bepaalde intent wordt er naar de detailpagina van deze intent genavigeerd. Op deze pagina kunnen utterances worden toegevoegd aan deze intent. De werking van dit toevoegen lijkt zeer sterk op die van LUIS. Een entity kan hier ook zeer makkelijk aangeduid worden door de gewenste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



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

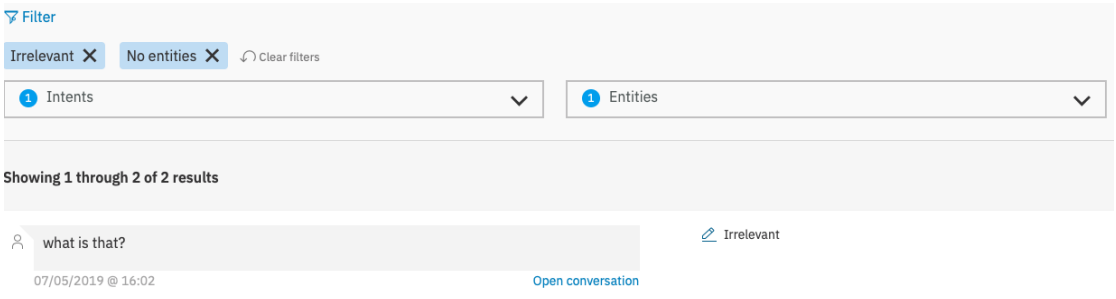
de al toegevoegde intents en entiteiten. Als voorbeeld kan er weer worden ingegeven 'Hi, I'm Ivor'. Er kan al direct worden gezien aan welke intent de utterance wordt gelinkt. In vergelijking met LUIS en Wit zijn er hier bij het testen geen percentages. Er is dus niet geweten hoeveel percent de Skill zeker is van een bepaalde intent. Ook kan er direct gezien worden welke entities 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 er een kanaal worden gekozen waarop de bot kan worden gedeployed. 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 er worden doorgeklikt naar de 'Weak understandings'. Hierin kunnen de utterances worden gelinkt 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

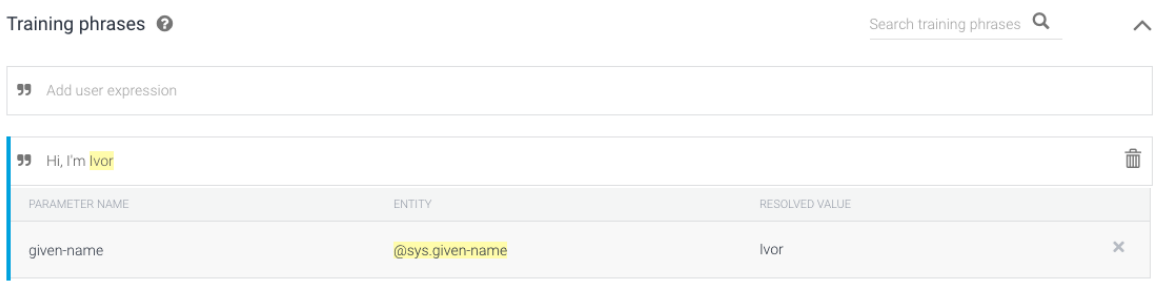
IBM Watson Assistant heeft ook een API. Deze API is zeer uitgebreid en bevat zo goed als alle aspecten van de site zelf. Van het creëren van een intent tot het toevoegen van nieuwe utterances en veel meer. In vergelijking met vorige frameworks heeft de assistant van IBM ook een endpoint voor het ophalen 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 is er 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 diede er zal worden gesproken tegen de bot 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.

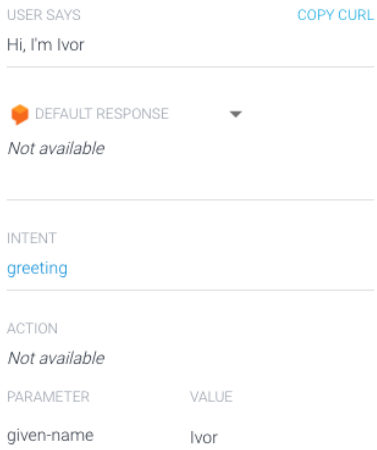


Figuur 3.13: Intent aanmaken Dialogflow

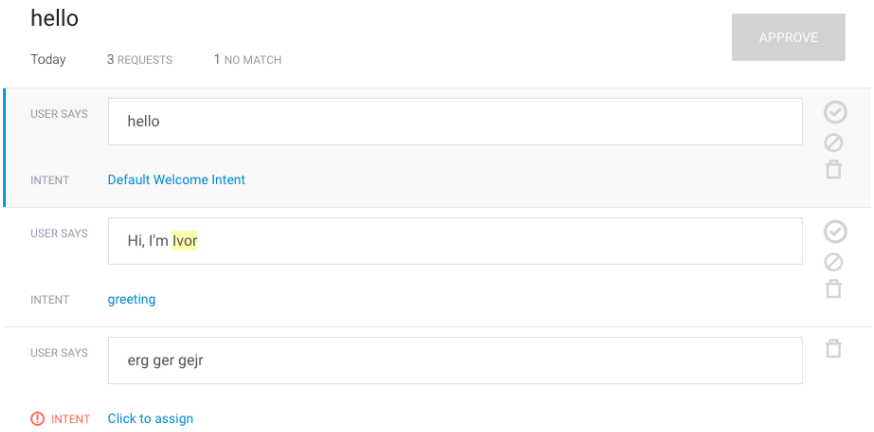
Deze gemaakte intents met de toepasselijk toegevoegde utterances kunnen al direct getest worden door de console die is voorzien. Er kan als voorbeeld 'Hi, I'm Ivor' gebruikt worden. Hierin kan gezien worden op afbeelding 3.14 dat deze input wordt geanalyseerd door de bot. Deze ziet dat de voorbeeldzin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

Bij dialogflow is er 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 failback intent' komen. Bij het training tabblad dient er dan zelf te worden gekeken welke intent hier dan exact bij hoort en dialogflow zal niet suggereren wie welke deze hoort. Er zijn hier ook geen percentages, dus er kan niet worden gezien waaraan dialogflow een bepaalde utterance zou linken die niet direct herkend wordt.

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



Figuur 3.14: Intent aanmaken Dialogflow



Figuur 3.15: Intent aanmaken Dialogflow

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 deze frameworks dezelfde problemen met zich mee brachten 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. Eenmaal 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 wordt gezien 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 percentages 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 onderzoekvraag. 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 gebruikgemaakt 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 te kunnen 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 instaat om deze zaken te garanderen. Wit.ai kwam in de buurt maar wit voorziet niet

genoeg informatie bij elke utterance. Er wordt wel gebruikgemaakt van scores hier maar bij elke utterance is er maar 1 intent met score voorzien in vergelijking met LUIS waarbij alle intents worden gezien 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 het model niet van nul zou moeten worden begonnen. Vanaf dit voorbeeld kan er verder gewerkt worden door nieuwe intents toe te voegen zodat deze alsmaar 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 inroepen, 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 entities die zich in LUIS bevinden zijn name, number en product waarvan number een ingebouwde entity is en de andere twee zelf gemaakte entities zijn. De naam dient als aanspreking bij de 'Greeting' intent. De entity 'number' is voor een hoeveelheid mee te geven bij bijvoorbeeld de intent 'AddMore', zodat er meerdere producten kunnen worden toegevoegd. De entity 'product' is voor te weten over welk product er gesproken wordt.

LUIS is volledig in het Engels geschreven omdat Engels de taal is dat 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 kan er een andere LUIS applicatie aangemaakt worden volledig in het Nederlands ofwel kan er een middleware worden tussengestoken die de Nederlandse tekst omzet naar het Engels zodat de Engelstalige 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 worden bekeken.

Het is zeer eenvoudig om met de bot te verbinden. Als deze lokaal aan het runnen is op een 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 veel gebruikt 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. Dit wordt hier gebruikt voor de webserver 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, is dat er een mogelijkheid bestaat 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 er onmiddellijk aan de slag worden gegaan met de botbuilder.

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

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

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 LUIS gekoppeld worden aan deze connectie. Dit wordt gedaan aan de hand van het endpoint dat verkregen wordt wanneer LUIS gedeployed wordt op de site van LUIS zelf. Dan ontstaat er een link naar het endpoint. Deze link wordt best in de environment variables gestoken zodat deze veilig blijft naar de buitenwereld toe. Daarna kan geluisterd worden naar inkomende boodschappen die bijvoorbeeld worden ingegeven in de bot framework emulator.

```
// connection \\
var conn = new builder.ChatConnector({
  appId: process.env.MICROSOFT_APP_ID,
  appPassword: process.env.MICROSOFT_APP_PASSWORD
});

// link LUIS to botbuilder \\
var bot = new builder.UniversalBot(conn);
bot.recognizer(new builder.LuisRecognizer(process.env.
   LUIS_MODEL_URL));

// listen for incoming requests \\
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 teruggekregen wordt 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 worden teruggegeven.

Een dialog werkt volgens de watervalmethode. 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, enz. 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 watervalmethode 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 verdergegaan 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      else {
8        next();
9      }
10   },
11   function (sess, result) {
12     sess.userData.name = result.response;
13     sess.send("Hello, " + sess.userData.name);
14   }
15 ]);
```

Listing 4.3: Watervalmethode

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 worden begonnen 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 terugkrijgen 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 terugkrijgen en zal deze willen linken aan de juiste dialog. Dit wordt gedaan 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: Dialog

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 er wordt ingegeven als input 'Add 5 more' te verkrijgen, kan er simpelweg volgende lijn code gebruikt worden.

```
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 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 eerst 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 worden bekeken. 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 efficiënt 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 efficiënt 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 worden bekomen.

5. Resultaten

In dit onderdeel zal de accuraatheid en de efficiëntie van het ontwikkelde proof of concept (POC) worden getest. Ook zal hier worden bekeken hoeveel utterances er bij de 'Review endpoint' utterances komen met en zonder de implementatie van de proof of concept.

Eerst en vooral zullen er per intent een 6-tal voorbeeld utterances worden opgesteld. Dat zal gedaan worden voor de volgende intents: AddMore, ClearBasket, DeleteItem, FindItem, Greeting, Help, RemoveMove, ShowBasket, ShowProfile, Stop en UpdateProfile. Aan de hand van deze voorbeelden zal er getest worden.

Deze testen zullen verschillende keren worden uitgevoerd. Er zijn een aantal parameters in de POC die kunnen worden aangepast. Uit deze testen zullen uiteindelijk blijken wat de beste parameters zijn voor de optimale accuraatheid.

5.1 Testdata

De testdata van deze opzet bevindt zich in bijlage. De eerste 4 voorbeeld utterances van elke intent zullen als input doorgestuurd worden naar de bot, dus naar LUIS zelf. Dit zal gedaan worden met en zonder het uitgewerkte proof of concept. Dan zal er gekeken worden hoeveel van deze utterances er bij 'Review endpoint' komen wanneer er wordt gewerkt op de gewone manier zonder het uitgewerkte POC en daarna zal gekeken worden hoeveel er bijkomen als er wordt gewerkt met het gemaakte POC. De laatste 2 voorbeeld utterances van elke intent zullen gebruikt worden om te controleren hoe hard de accuraatheid per utterance is veranderd voor en na het ingeven van de 4 eerste voorbeeld utterances. Dit zal ook gedaan worden eerst op de gewone manier en dan met het POC.

5.2 Uitwerking

Na het opzetten van de testdata kan er uiteindelijk getest worden. Op de site van LUIS zelf is er een tool ter beschikking om meerdere utterances uit te voeren en daar de resultaten kan bekijken genaamd 'batch testing'. De bedoeling is dat alle inputs die getest moeten worden in een json formaat moeten worden geüpload. In deze json dient bij elke input de intent meegegeven te worden en eventueel de entities. De entities zijn hier niet noodzakelijk omdat LUIS deze er automatisch uithaalt. Eenmaal de json is opgesteld, kan deze worden geïmporteerd in de 'batch testing panel'. LUIS leest deze json dan in en als ze verwerkt is, kan de test worden uitgevoerd.

Aan de hand van hoe groot de test is, zal deze redelijk snel worden uitgevoerd en kan er al direct naar de resultaten worden gekeken. Hier kan er gezien worden hoeveel van de utterances geslaagd zijn en kan er per utterance dan nog eens in detail bekeken worden met een percentage aan welke intent deze wordt gelinkt.

Bij de testen die zullen uitgevoerd worden, zal er telkens dezelfde set aan input gebruikt worden. Dit is een test die éénmaal zal worden uitgevoerd voor al de andere testen beginnen en éénmaal na alle andere testen. Aan de hand van deze test zal er bekeken worden in welke mate de POC invloed heeft op de scores die LUIS heeft aan elke utterance samen met de intent. Idealiter zou zijn dat de scores stijgen. Als deze zouden dalen zou dat betekenen dat er utterances gelinkt zijn aan de verkeerde intent. Dit is iets dat zeker moet worden vermeden.

Bij het uitvoeren van deze test zijn er 22 inputs die getest worden. Hier is gebleken dat LUIS 2 van die 22 utterances aan de foute intent heeft gelinkt. Dit kan komen door de beperkte aantal voorbeeld utterances die LUIS bevat. Het zou pas een probleem zijn moest uit de laatste test blijken dat dit aantal zou stijgen.

Nu dat deze test heeft gelopen, kan er aan de slag worden gegaan met de andere testdata, de 4 andere voorbeeld inputs per intent, in totaal zijn dit 44 inputs. Deze voorbeeldzinnen zullen via de 'Bot Framework Emulator' worden ingegeven, zo zullen de zinnen ook binnenkomen in LUIS als er echte gebruikers met de bot zouden praten via messenger bijvoorbeeld.

Omdat dit 44 zinnen zijn en dit is vrij veel om deze meerdere keren handmatig te moeten ingeven, zal er een manier moeten gezocht worden om dit te automatiseren. De oplossing die hiervoor is gevonden, is met een programma genaamd 'MurGaa auto talker'. In dit programma kunnen al de inputs éénmaal worden ingegeven en dient er verder niets meer gedaan te worden met deze. Er kan ingesteld worden dat dit programma de berichten verstuurd naar de emulator. Elke 5 seconden wordt er zo automatisch een bericht gestuurd naar de chatbot tot als alle inputs zijn ingegeven. Op deze manier dient niet elke zin apart te worden ingegeven en zal dit veel tijd besparen.

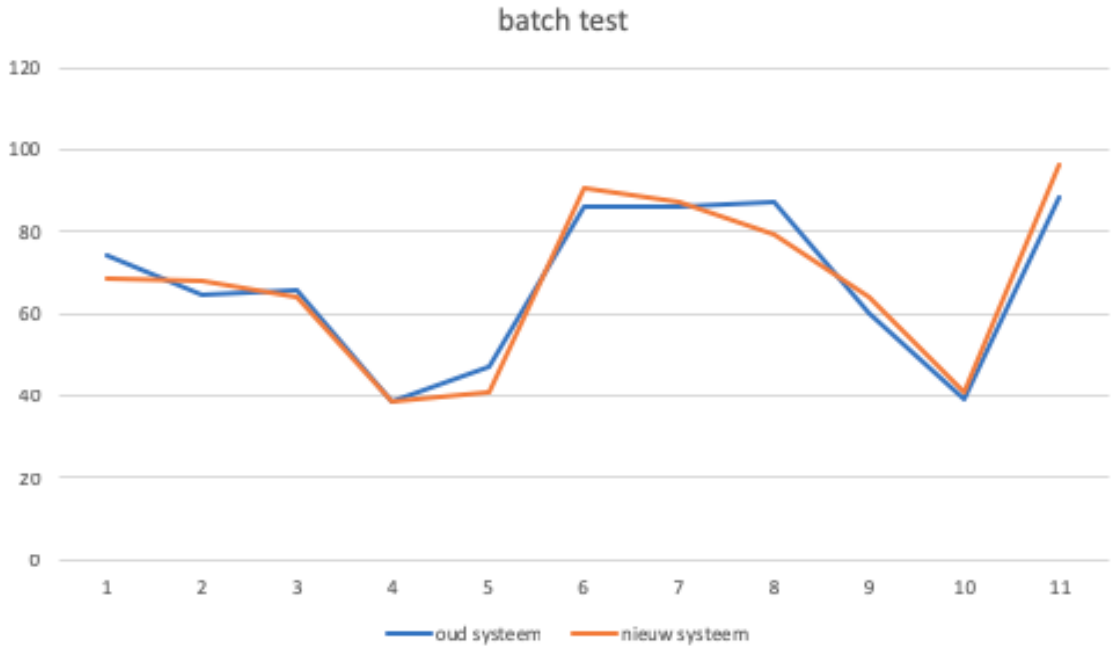
Al deze 44 inputs worden nu automatisch ingegeven in de emulator. Deze emulator stuurt deze dan naar de POC die dan LUIS zal aanroepen. Er zullen hier ook 2 testen worden uitgevoerd. Bij de eerste test zal de POC de utterances gewoon doorsturen naar LUIS

zonder verder iets te doen, zoals het altijd al gebeurde dus. Bij de tweede test zal het systeem worden aangezet die is ontwikkeld voor het controleren van elke utterance en deze dan zal toevoegen aan LUIS indien dat kan.

Door hier op deze 2 manieren te testen zal er in LUIS worden gekeken na elke test hoeveel van deze 44 inputs er komen in de 'Review endpoint utterances' tab en zal zo berekend kunnen worden hoeveel percent minder utterances in deze tab komen als het systeem is ingeschakeld.

Na het doorlopen van de eerste test is gebleken dat zo goed als alle inputs komen bij 'Review endpoint utterances' wat dus zeer inefficiënt is. Hier is er dus zeer veel werk nodig om elke utterance te bekijken en deze te valideren. Na de eerste test is het systeem ingeschakeld en zal een tweede test worden uitgevoerd. De 'MurGaa auto talker' zal weer de 44 inputs doorsturen naar de emulator. Uit de tweede test is gebleken dat in plaats van een 40-tal utterances die komen bij 'Review endpoint utterances' er nu maar een 20-tal meer inkomen. Dit betekent dat zo een 50 percent van de utterances niet meer bekeken zal worden want deze zijn al automatisch gelinkt aan de juiste intent. Na het controleren of deze wel effectief correct zijn toegevoegd, zijn er geen fouten gevonden. Het kan wel zijn dat LUIS soms een entity niet vindt of fout benoemt. Dit komt door het beperkt aantal utterances per intent.

Nu dit allemaal is getest en de resultaten positief zijn, kan nu de tweede batchtest worden gedaan. LUIS is nu veranderd doordat de POC automatisch nieuwe utterances heeft toegevoegd een de intent. Hier kan er dan gezien worden of dit wel effectief niet is verslechterd en eventueel beter is geworden.



Figuur 5.1: Resultaat batch testen

In afbeelding 5.1 is er duidelijk te zien dat door de implementatie van het nieuwe systeem

geen verbetering maar ook geen verslechtering is van LUIS. Bij sommige van de 11 utterances hier is het resultaat een beetje beter en dat is wat we wouden zien. Bij een aantal utterances is er ook een lichte verslechtering te zien. Dit komt doordat er per intent een beperkt aantal utterances zijn voorzien en deze utterances lagen allemaal zeer dicht bij elkaar, met andere woorden was de woordkeuze niet zo verschillend. In de testdata is er al meer gewerkt met echte zinnen en andere woordkeuzes. Door deze nieuwe utterances zullen de utterances per intent meer uiteenliggen met het gevolg dat de utterances van één bepaalde intent dichter zullen liggen bij de utterances van een andere intent. Bijvoorbeeld 'Can you please show my basket' en 'Can you please show my profile' liggen veel dichter bij elkaar dan 'show my basket' en 'what's my name'.

Er kan hier ook worden gekeken om een aantal parameters aan te passen om te kijken of dit de resultaten zou beïnvloeden. Hoe de resultaten hier beschreven zijn is met parameters die goed geschikt zijn voor deze LUIS testset. Elke utterance kwam bij de juiste intent terecht en de resultaten van de batchtest waren zelf lichtjes beter dan ervoor. Er kan altijd nog wel wat lichtjes met gespeeld worden met deze parameters maar er moet ook rekening gehouden worden met nieuwe utterances die er eventueel aan kunnen worden toegevoegd. Een beetje speling houden kan altijd voordelig uitkomen.

6. Conclusie

Wat we uit dit onderzoek kunnen besluiten is dat er wel degelijk een manier is om te realiseren wat er wordt gevraagt in de onderzoeksvraag. Wel is dit op een andere manier dan dat verwacht was. Er was verwacht dat minstens één van de onderzochte frameworks een API zou hebben die toegankelijk genoeg zou zijn voor dit te realiseren. Wat bleek was dat dit niet het geval was, geen enkele van de frameworks had deze gewenste API-call of was er niet genoeg informatie in de respons van deze call voor dit allemaal te realiseren op deze manier.

Wat wel bleek uit dit onderzoek is dat er een andere manier was om dit te realiseren. Eén framework genaamd LUIS had andere API-calls met de benodigde informatie om de input van de gebruikers rechtsstreeks op te vangen en met de info die LUIS meegaf, zo de utterance te koppelen met de juiste intent. Enkel LUIS was in staat om dit op deze manier te doen omdat deze per utterances meer info gaf dan andere frameworks. Juist dit was noodzakelijk om te kunnen onderzoeken of LUIS zeker genoeg was dat een utterance bij een bepaalde intent past. Bij de andere frameworks kan deze controle niet worden gedaan omdat er niet genoeg info wordt weergegeven.

Uit de testen die zijn gedaan met de proof of concept is gebleken dat 50 percent van de inputs van de gebruiker niet meer in de 'Review endpoint utterances' kwam maar rechtstreeks werd gekoppeld aan de juiste intent. Hier is ook gecontroleerd geweest wat de invloed is op de accuraatheid van LUIS. De resultaten hiervan zijn positief uitgevallen. De accuraatheid is niet gedaald, deze is zelf lichtjes gestegen. Er was verwacht dat deze wat meer zou stijgen maar door de beperkte hoeveelheid utterances per intent is dit niet zo gestegen. Dit zal meer en meer beginnen stijgen vanaf er meer en meer verschillende utterances zullen komen per intent.

Wat er verwacht was van het onderzoek was een automatisatie van 10 tot 30 percent. Wat uit het onderzoek is gebleken, is dat automatisatie effectief mogelijk is met zelf een beter resutaat dat werd verwacht. Uit de opzet en de tests was maar liefst 50 percent automatisch toegewezen aan de intent. Er kunnen altijd nog aanpassingen gedaan worden aan de parameters voor de controle van de utterance voor dit eventueel nog beter te maken maar er moet altijd wat speling zijn voor eventuele fouten op te vangen zodat deze kunnen vermeden worden.

Wat er in een vervolgonderzoek nog kan gebeuren is het onderzoeken van intents met meer ingewikkeldere utterances, bijvoorbeeld utterances met meer dan 1 entity. Hier zou ook beter kunnen gecontroleerd worden of deze entities correct uit de utterance worden gehaald. Een intent met nog maar weinig utterances zal meer last kunnen hebben met deze er correct uit te halen dan intents met al een grote verscheidenheid aan utterances. Ook zouden er meerdere LUIS applicaties kunnen gestest worden, met elk een vershillend doen en verschillende testdata, eventueel een veel grotere testdataset. Ook bij deze controleren in welke mate de input van de gebruiker al automatisch gelinkt wordt met de correcte intent.

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 ordinaire 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 percent 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 uitgehaald.

Een paar van de bekendste NLP-platformen zijn Wit.ai en LUIS.ai. In elk van die platformen kunnen er intents manueel worden getraind 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. De intentis kunnen voornamelijk in twee categorieën ingedeeld worden.

- Casual intents
- Bedrijfsintents

Casual intenties 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 er wordt ontvangen uit echte gesprekken, hoe beter de NLP-service kan worden getraind voor intenties. Het proces van het opnieuw trainen van het systeem moet doorgaan totdat het foutenpercentage vermindert. Dit wordt gedaan wanneer er supplementaire input komt.

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 inplementeren. Normaal moeten de inputs manueel gelinkt worden aan de juiste intent. Dit komt doordat het systeem niet 100 percent 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 percent 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

B. Testdata

AddMore

- Can you please add 5 more?
- I want to add 3 more.
- Will you please add 4 more?
- Let's add 2 more.
- Add 8 more.
- I need 3 more.

ClearBasket

- I want to clear my basket.
- Can you remove my basket?
- I want my basket to be empty.
- Can you delete everything from my basket?
- Clear my basket.
- Empty my shopping list.

DeleteItem

- Can you remove this from my shopping list?
- I want to remove this item.
- Please, delete this from my shopping list.
- Will you remove this from my basket?
- Delete item.
- Remove this.

FindItem

- I want to add some salami.
- Can you find salami for me please?
- I would like to find 5 beers.
- I want to find some beers.
- I need 8 more beers please.
- I am looking for catfood.

Greeting

- Hi, my name is Ivor.
- Good morning, I'm Ivor.
- Hello, how are you?
- Good evening.
- Hello, what's up?
- Hello.

Help

- Please, I need help.
- Can you help me with this?
- What is this?
- I don't get it how this works.
- I want help.
- Please, help me.

RemoveMore

- Can you remove these items?
- I want to remove this 2 times.
- Can you please remove this a couple of times?
- Remove 5 pieces from this item.
- Remove 8 items.
- Remove items.

ShowBasket

- I want to see my basket.
- Can you show me my shopping list?
- What's in my cart?
- Please, show my cart.
- Show my shopping basket?
- Please, show me my basket.

ShowProfile

- I want to see my profile.

- Can you show me my information?
- What's my profile information?
- Show my profile.
- What's my name.
- Show my username.

Stop

- Can you stop?
- Please start over again.
- I want to exit.
- Let's quit.
- Can I stop?
- Make it stop.

UpdateProfile

- I want to update my name.
- Can you change my username?
- Update my profile please.
- Will you change my username?
- Change my name.
- Update my username.