



Faculteit Bedrijf en Organisatie

Interactive Visualisations for IBM Watson Assistant Chatbot Flows

Thomas Schuddinck

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

Promotor:
Johan Van Schoor
Co-promotor:
Jürgen De Commer

Instelling: Zoovu

Academiejaar: 2019-2020

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Interactive Visualisations for IBM Watson Assistant Chatbot Flows

Thomas Schuddinck

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

Promotor:
Johan Van Schoor
Co-promotor:
Jürgen De Commer

Instelling: Zoovu

Academiejaar: 2019-2020

Tweede examenperiode

Woord vooraf

Deze bachelorproef werd geschreven in functie van het succesvol afronden van de opleiding Toegepaste Informatica, met afstudeerrichting Mobile Applications, aan de Hogeschool van Gent (HOGENT). Ik ben geïnteresseerd in alle nieuwe trends gaande van IoT en Augmented Reality, tot Artificiële Intelligentie en chatbots. Dit onderzoek was een opportuniteit om hierin verder te gaan.

Zoals in elk onderzoek stond ook ik er niet alleen voor. Ik heb hulp gekregen van verschillende mensen en zou hen hierbij oprecht willen bedanken.

In eerste instantie wil ik graag mijn co-promotor, Jürgen De Commer, willen bedanken om dit onderwerp aan te bieden en voor alle ondersteuning voor en tijdens het uitvoeren van de bachelorproef. Ik kon steeds rekenen op een snelle respons wanneer ik met vragen zat en kreeg steeds nuttige feedback, zowel over de inhoud van de bachelorproef, als over de structuur en taalgebruik.

Daarnaast wil ik ook zeker mijn promotor, Johan Van Schoor, bedanken voor de alle steun en feedback die ik kreeg bij het schrijven van de bachelorproef. Ik kon ook steeds bij hem terecht als ik vragen had over zaken zoals het gebruik van bronnen en de correctheid van mijn bronvermelding.

Ook wil ik graag mijn ouders bedanken voor de financiële en mentale steun gedurende mijn opleiding. Zonder hen was dit nooit gelukt.

Graag zou ik ook mijn vriendin willen bedanken voor de mentale steun gedurende de zware periode die niet enkel te wijten viel aan de opleiding, maar ook aan de quarantaine, ten gevolge van de uitbraak van COVID-19.

Als laatste wil ik ook mijn goede vrienden en collega-studenten Aleandro Delys en Matthias De Roo bedanken voor alle mentale steun en het collectief meedenken wanneer ik vast zat bij het schrijven van het prototype.

Samenvatting

De aanleiding tot het onderzoek in deze paper is een bestaand probleem, bij softwarebedrijf Clever (inmiddels Zoovu), rond het visueel vormgeven van een chatbotflow. Dit kan aanleiding geven tot misverstanden en communicatieproblemen met de klant. In deze paper wordt een uitgebreide analyse gedaan van de verschillende onderdelen een chatbot en welke daarvan relevant zijn voor visualisatie. Er wordt de focus gelegd op IBM Watson Assistant, de chatbot van het IBM Watson framework, dat door Zoovu gebruikt wordt voor hun Natural Language Processing-gedeelte. Deze onderdelen zijn vereist voor het selecteren van de correcte visualisaties. Daarnaast is niet enkel het weergeven van deze chatbots van belang maar moet er ook interactie mogelijk zijn. Deze bewerkingen zullen ook een indicator zijn voor het kiezen tussen diverse visualisaties.

Na een uitgebreide literatuurstudie werd een exportbestand van een Watson Assistant chatbot geanalyseerd en werden alle velden gelabeld naar relevantie. De volgende hoofdstukken gaan visualisatie types en visualisaties filteren en kijken welke operaties moeten en kunnen gebruikt worden.

Uit het onderzoek komt naar boven dat dendrogrammen een ideale visualisatie zijn om een chatbotflow weer te geven. Er zijn ook andere mogelijkheden zoals treemaps en sunburst diagrammen maar deze vereisen meer operaties om correct met deze visualisatie te werken en zijn veel moeilijker te gebruiken wanneer er “jump-to”-relaties moeten weergegeven worden. Zoomen is een operatie die verplicht aanwezig zou moeten zijn om een zeer grote visualisatie te kunnen lezen. De operaties *filter* en *extract* kunnen hier goed bij helpen door de hoeveelheid aan informatie dat weergegeven wordt te reduceren.

Daarnaast wordt er een stuk software geschreven die één of meerdere visualisaties gegenereerd. Op deze visualisaties kunnen er één of meerdere bewerkingen op gedaan worden.

De voorkeur voor deze bewerkingen gaat naar het in- en uitzoomen en het reduceren van informatie door meer in te zoomen op relevante zaken of irrelevante informatie te filteren van de componenten van zulke chatbotflows.

Toekomstig onderzoek kan verder gaan op welke visualisaties gebruikt kunnen worden voor analytisch onderzoek. Visualisaties zoals een heatmap en Sankey diagram zouden hier van pas kunnen komen.

Inhoudsopgave

1	Inleiding	17
1.1	Probleemstelling	17
1.2	Onderzoeksvraag	18
1.2.1	Hoofdonderzoekvragen	18
1.2.2	Deelonderzoekvragen	18
1.3	Onderzoeksdoelstelling	18
1.4	Hypothese	19
1.5	Opzet van deze bachelorproef	19
2	Stand van zaken	21
2.1	Terminologie	21
2.1.1	Datavisualisatie	21
2.1.2	Artificiële Intelligentie	21

2.1.3 Chatbot	22
2.1.4 Virtual Assistant	24
2.1.5 Natural Language Processing	25
2.1.6 Natural Language Understanding	26
2.1.7 Natural Language Generation	26
2.1.8 Conversational Search	27
2.2 Datavisualisatie in het dagelijkse leven	28
2.2.1 Wetenschappelijke versus Informatie-Visualisatie	28
2.2.2 Indeling van Visualisaties	28
2.3 Chatbots in het dagelijkse leven	32
2.3.1 IBM Watson Assistant	32
2.3.2 Dialogflow	35
2.3.3 Amazon Lex	36
2.3.4 Microsoft Bot Framework	40
2.3.5 Rasa (Open Source)	42
2.3.6 Samenvatting	43
2.4 Bestandsformaten voor Afbeeldingen	43
2.4.1 Vectorafbeeldingen	44
2.4.2 Rasterafbeeldingen	44
2.4.3 Vector versus Raster	46
3 Methodologie	47
3.1 Definitie en Requirements Chatbotflow	48
3.2 Evaluatie Chatbot Data	48
3.3 Voorwaarden Visualisatietypes	49

4	Opbouw Watson Assistant Chatbot	51
4.1	Algemene structuur van de chatbot	51
4.2	Chatbot Informatie	52
4.3	Intents	52
4.4	Entities	53
4.5	Dialog Nodes	54
4.5.1	Dialog Nodes Types	54
4.5.2	Standard (en Frame) Dialog Nodes	55
4.5.3	Slot Nodes	56
4.5.4	Event Handler Nodes	56
4.5.5	Response Condition Nodes	58
4.5.6	Overzicht datavelden chatbot	58
5	Datavisualisatie Types	61
5.1	Types van Datavisualisatie	61
5.2	Lineaire Datavisualisaties	62
5.3	Planaire Visualisaties	63
5.4	Volumetrische Datavisualisaties	63
5.5	Tijdsgebonden Datavisualisaties	64
5.6	Multidimensionale Datavisualisaties	65
5.7	Boom Datavisualisaties	67
5.8	Netwerk Datavisualisaties	67
5.9	Conclusie Datavisualisaties types	68

6	Datavisualisaties	69
6.1	Netwerk Visualisaties	69
6.2	Network Diagram	70
6.3	Chord Diagram	70
6.4	Arc Diagram	73
6.5	Sankey Diagram	77
6.6	Heatmap	77
6.7	Hive	77
6.8	Dendrogram	78
6.9	Treemap	78
6.10	Circular Packing	79
6.11	Sunburst	79
6.12	Hierarchical Edge Bundling	81
6.13	Conclusie	81
7	Interactie bij Visualisaties	85
7.1	Algemeen	85
7.2	Network	86
7.3	Arc Diagram	87
7.4	Dendrogram	87
7.5	Treemap en Circular Packing	88
7.6	Sunburst	88
7.7	Conclusie	89

8	Van Chatbot naar Visualisatie	91
8.1	Specificaties van de Demo	91
8.2	Layout van de Demo	94
8.3	Chatbotflow met een Dendrogram	94
8.4	Chatbotflow met een Sunburst-Diagram	95
8.5	Evaluatie van de Visualisaties	95
9	Conclusie	97
9.1	Hoofdonderzoeksvraag Visualisaties	97
9.2	Hoofdonderzoeksvraag Bewerkingen/Operaties	99
9.3	Prototype	99
A	Onderzoeksvoorstel	101
A.1	Terminologie	101
A.1.1	Artificiële Intelligentie	101
A.1.2	Chatbot	102
A.1.3	Natural Language Processing	103
A.2	Introductie	103
A.3	State-of-the-art	104
A.4	Methodologie	105
A.5	Verwachte resultaten	107
A.6	Verwachte conclusies	107
B	Exportbestand Watson Assistant	109
B.1	Basis Structuur	109

B.2	Chatbot Informatie	110
B.3	Chatbot Intents	111
B.4	Chatbot Entities	115
B.5	Chatbot Nodes	120
C	JSON-Transformaties	135
C.1	Overzicht van Transformatiescript	135
C.2	Verdeelfunctie	136
C.3	Creatiefunctie voor Hiërarchische Structuur	137
C.4	Hulpfunctie voor Ondersteunende Nodes	138
C.5	Hulpfunctie voor Output	139
C.6	Exportfunctie voor Visualisatiescripts	140
	Bibliografie	141

Lijst van figuren

2.1	Indeling van definities van artificiële intelligentie	22
2.2	Werking van een dialog node	24
2.3	Een schakeling van meerdere dialog nodes	25
2.4	De werking van een IBM Watson Assistant chatbot	26
2.5	Structuur Natural Language Processing	27
2.6	Het proces voor de creatie van informatie visualisaties	30
2.7	Samenvattend model voor informatie visualisatie	31
2.8	IBM Watson Assistant gebruikersinterface	34
2.9	Dialogflow gebruikersinterface	37
2.10	Amazon Lex gebruikersinterface	39
2.11	Overzicht Microsoft Bot Framework	41
2.12	Het verschil tussen vector- en rasterafbeelding na inzoomen	45
4.1	Werking van event handler dialog nodes	57
5.1	Voorbeeld van een voorstelling van een flow met een planaire visualisatie	64
5.2	Voorbeeld van een voorstelling van een flow met een volumetrische visualisatie	65

5.3	Voorstelling van 4-dimensionele data in een 3-dimensionale ruimte	66
6.1	Overzicht van “Netwerk”-visualisaties	71
6.2	Voorbeeld chatbotflow met een netwerk diagram	72
6.3	Voorbeeld chatbotflow met chord diagram	74
6.4	Probleem met volgorde bij een arc diagram	75
6.5	Probleem met “Jump-To”-referenties bij een arc diagram	76
6.6	Voorbeeld chatbotflow met een dendrogram	79
6.7	Voorbeeld chatbotflow met een sunburst	80
6.8	Voorbeeld van hierarchical edge bundling	82
6.9	Voorbeeld procentuele spreiding van een DialogFlow chatbotflow	83
8.1	Visualisatie van een chatbotflow met een dendrogram	92
8.2	Visualisatie van een chatbotflow met een sunburst-diagram	93
8.3	Het detailpaneel van het demoproject	94
8.4	Overzicht chatbotflow met een dendrogram	95
8.5	Extraheren van het gewenste pad met een sunburst-diagram	96
A.1	Indeling van definities van artificiële intelligentie	102
A.2	De werking van een IBM Watson Assistant chatbot	103
A.3	De sessie flow van een Dialogflow chatbot visueel weergegeven	105
A.4	Een mogelijke voorstelling van een interactieve graaf	106

Lijst van tabellen

2.1	Overzicht van chatbot frameworks	43
2.2	Overzicht van vector- en rasterafbeeldingen	46
4.1	Overzicht Watson Assistant exportdata	59

1. Inleiding

The greatest value of a picture is when it forces us to notice what we never expected to see. — John Wilder Tukey (1977), statisticus

Bovenstaande uitspraak van John W. Tukey, wiskundige en staticus, geeft concreet de essentie van datavisualisatie. In 2017 werd er gemiddeld elke dag 2.5 triljoen bytes aan data geproduceerd (Domo, 2017). Dat cijfer ligt ondertussen veel hoger en zal blijven groeien. Deze data hebben nood aan visualisaties om verstaanbaar en dus ook nuttig te zijn (the TOM agency, 2020). Dit is het kernidee achter deze paper.

1.1 Probleemstelling

Clever, ondertussen onderdeel van Zoovu¹, is een bedrijf dat chatbots ontwikkelt en afstemt op de wensen van hun klanten. Ze specialiseren zich op chatbots met conversational search. Deze chatbots gebruiken IBM Watson voor het NLP-gedeelte. Een veel voorkomend probleem is de miscommunicatie tussen hun ontwerpers en klanten. Voor de klant is het zeer moeilijk om zo een chatbot, meer bepaald de flow ervan, voor te stellen. Om de communicatie te verbeteren hebben ze niet enkel nood aan visualisaties van de chatbot-flows, maar moet het ook mogelijk zijn om enkele bewerkingen hierop te doen. Deze bewerkingen zijn nodig om letterlijk in te kunnen zoomen op de gewenste onderdelen van de gegenereerde figuur.

¹vanaf hier zal er steeds Zoovu gebruikt worden om naar de opdrachtgever te verwijzen

1.2 Onderzoeksvraag

1.2.1 Hoofdonderzoeksvragen

Het onderzoek streeft er naar de volgende twee hoofdonderzoeksvragen te beantwoorden:

- Welke visualisaties zijn er toepasselijk om een duidelijk beeld te geven van de chatbotflows voor IBM Watson Assistant chatbots?
- Welke bewerkingen zijn er noodzakelijk om op deze visualisaties uit te kunnen voeren en waarom?

1.2.2 Deelonderzoeksvragen

Om een concreet antwoord te kunnen geven op bovenstaande hoofdonderzoeksvragen, moeten volgende vragen reeds beantwoord zijn:

- Uit welke elementen is een chatbotflow opgebouwd?
- Welke elementen zijn noodzakelijk voor de flow te visualiseren?
- Hoe kunnen deze elementen visueel worden voorgesteld?
- Welke bewerkingen kunnen er toegepast worden op deze visualisaties?

Volgende vragen moeten gesteld worden voor er een prototype kan opgesteld worden:

- Welke bestandsformaten zullen er worden gebruikt worden?
- Welke softwaretalen worden er gebruikt?
- Welke softwarelibraries kunnen gebruikt worden voor visuele voorstelling van de chatbotflow?
- Welke transformaties moeten er op de datainput gebeuren om deze visueel voor te kunnen stellen?

1.3 Onderzoeksdoelstelling

Het onderzoek heeft een tweevoudig doel. Enerzijds wordt er gekeken naar welke opties er zijn voor het visualiseren en het interacteren ervan. Dit zal dus het resultaat zijn van het onderzoek. Na de analyse wordt ook een concreet voorbeeld uitgewerkt waarbij er gestart wordt met een exportbestand van een IBM Watson Assistant chatbot. Het resultaat is een visualisatie van deze chatbot die rekening houdt met de resultaten en adviezen die uit het onderzoek voortkomen. Dit prototype focust zich eerst en vooral op een Watson Assistant chatbot maar zou idealiter ook met een Zoovu chatbot moeten kunnen werken.

1.4 Hypothese

Na een eerste literatuurstudie wordt de volgende hypothese bekomen: een chatbot bestaat uit knopen die onderling met elkaar verbonden zijn en die bereikbaar zijn door middel van de gebruikersinvoer. Relaties zijn dus van essentieel belang bij het voorstellen van een chatbotflow. Op basis hiervan lijkt het gebruik van een boom of graaf het meest interessant voor het visualiseren. Afhankelijk van de hoeveelheid data en wat de structuur van de knopen is, zullen het zoomen en knippen in deze structuur de belangrijkste basis bewerkingen zijn, die op deze visualisaties toegepast kunnen worden.

1.5 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 onderzoeksdomain, 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 het exportbestand van een IBM Watson Assistant chatbot onderzocht en worden de alle datavelden gelabeld.

In Hoofdstukken 5 en 6 worden de mogelijke visualisaties opgesomd en met elkaar vergeleken. Ongeldige visualisaties worden gefilterd.

In Hoofdstuk 7 worden de interacties met visualisaties besproken.

In Hoofdstuk 8 wordt kort een prototype toegelicht waarbij er vertrokken wordt van het exportbestand dat besproken werd in Hoofdstuk refch:opbouwwa.

In Hoofdstuk 9, 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

2.1 Terminologie

In deze sectie worden enkele fundamentele begrippen uitgelegd. Deze zijn noodzakelijk voor het begrijpen van deze paper.

2.1.1 Datavisualisatie

Datavisualisatie is een voorstelling van data/gegevens in een pictografische of grafische vorm. Een datavisualisatietool zorgt voor het genereren van zulke visualisaties. Deze visualisaties zorgen voor intuïtieve mogelijkheden om de data te onderzoeken en te analyseren om zo patronen, verbanden en oorzaak-gevolg situaties terug te vinden en op basis daarvan beslissingen te kunnen nemen. (Bikakis, 2018)

2.1.2 Artificiële Intelligentie

Artificiële intelligentie, ook kunstmatige intelligentie of kortweg AI, heeft geen éénduidige definitie. Er zijn doorheen de tijd heel wat definities aan toegekend. In deze paper wordt er gekozen om het model van Russel & Norvig te volgen (Figuur A.1).

Het model beschrijft artificiële intelligentie als een combinatie van vier concepten: menselijk denken, menselijk handelen, rationeel denken en rationeel handelen.

Menselijk denken gaat dieper in op het modelleren van het menselijk brein. A.d.h.v. van dit model probeert men de denkwijze van mensen te achterhalen. (Kent, 2019)

<p>Thinking Humanly</p> <p>“The exciting new effort to make computers think . . . <i>machines with minds</i>, in the full and literal sense.” (Haugeland, 1985)</p> <p>“[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . .” (Bellman, 1978)</p>	<p>Thinking Rationally</p> <p>“The study of mental faculties through the use of computational models.” (Charniak and McDermott, 1985)</p> <p>“The study of the computations that make it possible to perceive, reason, and act.” (Winston, 1992)</p>
<p>Acting Humanly</p> <p>“The art of creating machines that perform functions that require intelligence when performed by people.” (Kurzweil, 1990)</p> <p>“The study of how to make computers do things at which, at the moment, people are better.” (Rich and Knight, 1991)</p>	<p>Acting Rationally</p> <p>“Computational Intelligence is the study of the design of intelligent agents.” (Poole <i>et al.</i>, 1998)</p> <p>“AI . . . is concerned with intelligent behavior in artifacts.” (Nilsson, 1998)</p>

Figuur 2.1: Indeling van definities van artificiële intelligentie

Source: <https://eetn.eu/knowledge/detail/Evidence-Summary–Artificial-Intelligence-in-education>

Het concept “menselijk handelen” kwam tot stand toen Alan Turing de Turing test creëerde om te onderzoeken of een computer een menselijke ondervrager kan misleiden door zich voor te doen als een mens. De test slaagt als de ondervrager geen onderscheid kan maken tussen mens en computer. (Stuart & Peter, 2016)

Het derde gebied is rationeel denken, waar de redeneringsprocessen en de bijhorende conclusies worden beschreven. Bij rationeel handelen tracht men rationele agenten op te stellen: een agent die de beste uitkomst bekomt. Deze uitkomst is niet per se de beste mogelijke uitkomst maar de beste uitkomst volgens de kennis die de agent op dat moment heeft. Een agent kan simpelweg beschreven worden als een entiteit die zijn omgeving kan waarnemen door gebruik te maken van sensoren en invloed kan uitoefenen binnen diezelfde omgeving. (Lievens, 2019)

2.1.3 Chatbot

Een chatbot kan gedefinieerd worden als een programma, dat op basis van menselijke interactie¹, conversaties simuleert. (Rouse, 2019)

Een IBM Watson² chatbot bestaat uit meerdere onderdelen. Enkele belangrijke onderdelen

¹door middel van tekst of spraak

²dit is het chatbot framework dat hetzelfde NLP gebruikt als Zoovu en is dus het meest relevante voor het onderzoek

zijn:

- intents
- entities
- channels
- webhooks
- dialogs (en dialog nodes)
- slots

(McGrath, 2017)

Een *intent*, simpelweg vertaald intentie of bedoeling, is het doel dat een gebruiker wil bereiken. Het bevat een verzameling van mogelijke gebruikersinput die bedoeld is om de chatbot aan te geven wat het gebruikersdoel is. Een voorbeeld van een intent kan het samenstellen van een cupcake zijn. (IBM, 2020c)

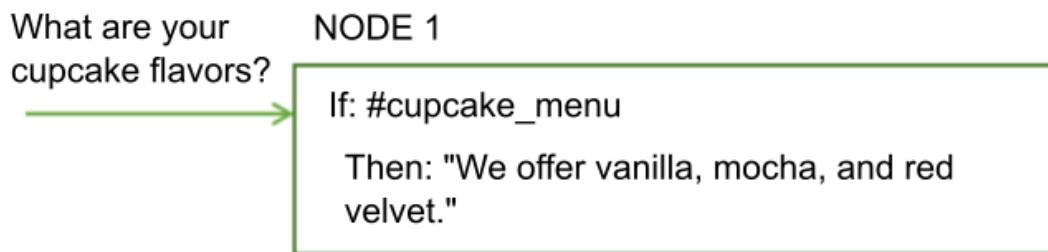
Waar intents gezien kunnen worden als taken die een gebruiker wil uitvoeren, kunnen *entities* gezien worden als parameters die noodzakelijk zijn voor het uitvoeren van deze taken. In het voorbeeld van de cupcakes, zijn de entities zaken zoals de toppings, het soort deeg, het aantal cupcakes... Entities worden herkend als een onderdeel van de gebruikersinput. Wanneer bijvoorbeeld de intent voor de cupcakes wordt getriggered, zal de chatbot op zoek gaan naar toppings die hij kan herkennen aan de hand van de gedefinieerde entites. Entities kunnen ook gebruikt worden om patronen te vinden door gebruik te maken van reguliere expressies.

Entities hebben niet enkel waarden maar kunnen ook synoniemen hebben. Dit is gelijk-aardig aan hoe intents trainingszinnen hebben. Deze synoniemen kunnen bijvoorbeeld afkortingen zijn, veel gebruikte anderstalige benamingen of letterlijk synoniemen van die waarde. Voor het geval van toppings zijn “koekjes” en “vanille crème” goede voorbeelden van waarden, met synoniemen respectievelijk “koeken”, “cookies”, “koek”, “biscuit”... en “crème vanille”, “vancrème”, “vanilla”, “vanille”... (IBM, 2020b)

Channels zijn het medium waarmee er met de chatbot geconverseerd kan worden. Populaire channels zijn Messenger, Skype en Slack, maar ook email valt hieronder. In deze paper zal er soms de term *integrations* gebruikt worden in plaats van channels. Er wordt steeds het kanaal bedoeld waarover er met de chatbot gecommuniceerd wordt, ongeacht welke term er gebruikt wordt.

Chatbots kunnen naast het voeren van conversaties ook programmatisch acties uitvoeren. Een chatbot om reservaties in een restaurant te maken zou geen nut hebben als er op het eind van de conversatie geen reservatie wordt gemaakt. Om zulke functionaliteiten mogelijk te maken is er nood aan webhooks. *Webhooks* gaan door middel van een POST-request een oproep doen naar het achterliggend systeem om wijzigingen aan te brengen. De werking van webhooks is niet relevant en valt buiten de scope van het onderzoek. (IBM, 2020f)

Een *dialog* kan gezien worden als de flow van interacties met een gebruiker. Een dialog is opgebouwd uit één of meerdere dialog nodes. Waar een dialog een reeks aan interacties is,



Figuur 2.2: Werking van een dialog node

Source: <https://cloud.ibm.com/docs/services/assistant?topic=assistant-dialog-overview>

kan een dialog node gezien worden als één enkele interactie ofwel een vraag en antwoord.

Een *dialog node* wordt geactiveerd wanneer er aan een bepaalde conditie wordt voldaan. Dit kan bijvoorbeeld een intent zijn die de chatbot herkent, of een entity... zoals in figuur 2.2. De node in dit voorbeeld, wordt getriggered wanneer de intent “cupcake_menu” wordt herkend. Elke node heeft steeds één ouder³ tenzij deze zelf de root⁴ is. (IBM, 2020d)

Figuur 2.3 geeft een voorbeeld van hoe deze nodes aaneengeschakeld zijn. Een dialog kan dus als een boomstructuur gezien worden die opgesteld wordt op basis van zijn dialog nodes. Deze “boomstructuur” is van essentieel belang voor het creëren van datavisualisaties van een chatbotflow. (IBM, 2020d)

Tenslotte zijn er nog *slots*. Dit zijn onderdelen van een dialog node met als doel het ophalen van informatie. Stel dat je in het cupcake voorbeeld een bestelling wilt plaatsen kan de chatbot vragen om hoe laat je de cupcakes wilt komen ophalen. Het uur dat je doorgeeft is de informatie dat het slot binnenhaalt en doorgeeft voor verdere verwerking zoals bijvoorbeeld een bevestingsmail versturen met je opgegeven uur. Vaak worden er entites gebruikt voor het herkennen van de gevraagde data. Zo kan er een entity zijn dat alles van datum en tijd bijhoudt en kan zo op basis van je geschreven bericht het uur eruit halen. In sommige gevallen worden er ook intents gebruikt. (IBM, 2020e)

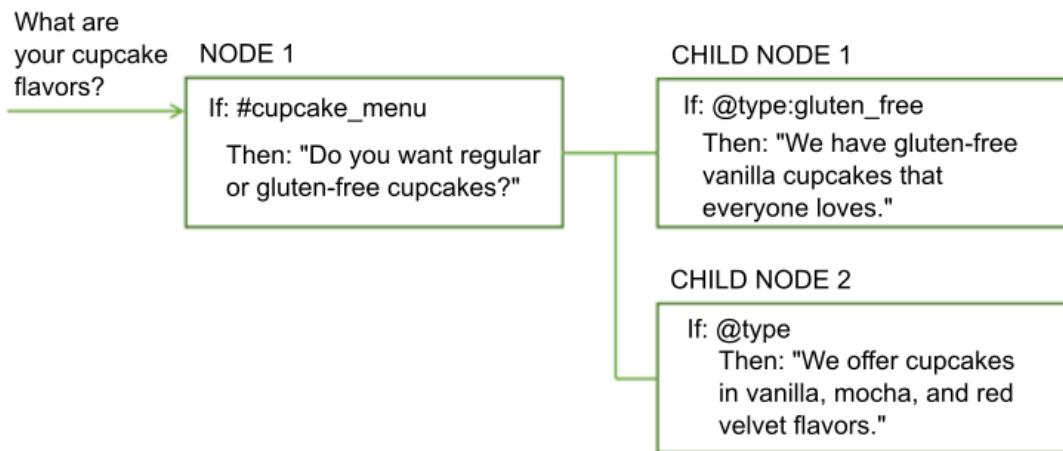
Dit zijn niet de enige componenten van een chatbot maar kennis van de andere onderdelen is niet noodzakelijk voor het begrijpen van de paper en zullen daarom ook niet aan bod komen.

2.1.4 Virtual Assistant

Er is veel discussie over wat nu juist het verschil is tussen een chatbot en een virtual assistant. Deze paper maakt volgende onderscheid:

³of parent. In deze paper zal er vaker de Engelse term gebruikt worden

⁴letterlijk vertaald “wortel”: verwijzing naar de wortels van een boom



Figuur 2.3: Een schakeling van meerdere dialog nodes

Source: <https://cloud.ibm.com/docs/services/assistant?topic=assistant-dialog-overview>

Een virtuele assistent (VA) is een programma dat zich richt op het dagelijks leven van de gebruiker en dus letterlijk als een assistent zaken regelt. Deze taken omvatten het zetten van alarmen en het inplannen van afspraken in de digitale agenda. Een goed voorbeeld van een VA is Siri.

Chatbots daarentegen focussen zich eerder op business level zoals bijvoorbeeld klantenservices. In tegenstelling tot virtual assistants, maken chatbots weinig of geen gebruik van emotie detectie en is de service gericht op het vervullen van business taken, in plaats van het ondersteunen van de alledaagse handelingen van de gebruiker.

Daarnaast is er veel debat over welke van de twee het intelligentste is, meer mogelijkheden biedt, beter de context onthoudt.... Dit is niet relevant voor dit onderzoek en er wordt niet verder op ingegaan. Er wordt vanuit gegaan dat dit verschilt van chatbot tot chatbot en van virtual assistant tot virtual assistant. (DeplerAI, 2019; Joshi, 2018)

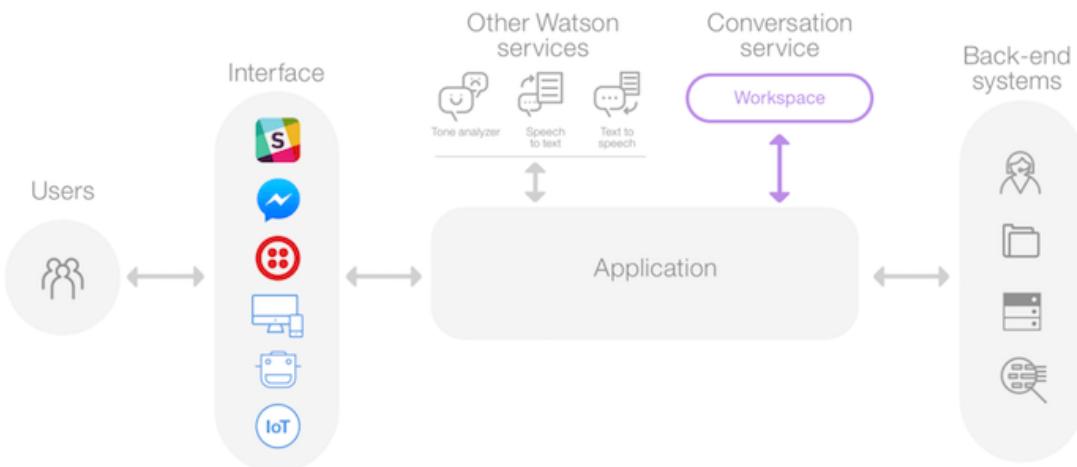
Voor deze paper is de structuur van een chatbot/virtuele assistent van belang en zullen deze termen beiden voorkomen maar dit zal geen invloed hebben op het resultaat van de visualisatie.

2.1.5 Natural Language Processing

NLP, of Natural Language Processing⁵, is een onderdeel van het onderzoeksgebied artificiële intelligentie dat zich bezighoudt met het verstaanbaar maken van mensentaal voor een computer. Het omvat zaken zoals language modeling, parsing en morphology. (Otter e.a., 2019).

NLP ligt aan de basis van elke chatbot en is noodzakelijk om gepast te kunnen reageren.

⁵In deze paper worden zowel de volledige term als de afkorting gebruikt



Figuur 2.4: De werking van een IBM Watson Assistant chatbot

Source: https://www.ibm.com/cloud/architecture/tutorials/watson_conversation_support

Wanneer een chatbot gebruikersinput ontvangt via een interface, zoals bijvoorbeeld Messenger, zal NLP de boodschap vertalen naar een probleem dat verstaanbaar is voor het programma. Wanneer de input voor het programma duidelijk is, kan de bot op zoek gaan naar een correct antwoord in het achterliggend systeem. (Figuur 2.4).

2.1.6 Natural Language Understanding

Natural Language Understanding(NLU)⁶, is een onderdeel van NLP. Waar Natural Language Processing zorgt dat de input verstaanbaar is voor de computer, zorgt NLU dat de computer het begrijpt. Waar NLP woord voor woord kijkt, zal NLU de volledige zinnen beschouwen en naar de grammatica kijken, de context waarin de conversatie loopt en op zoek gaan naar intents en entities. (Lola.com, 2016; Rasa, 2019).

2.1.7 Natural Language Generation

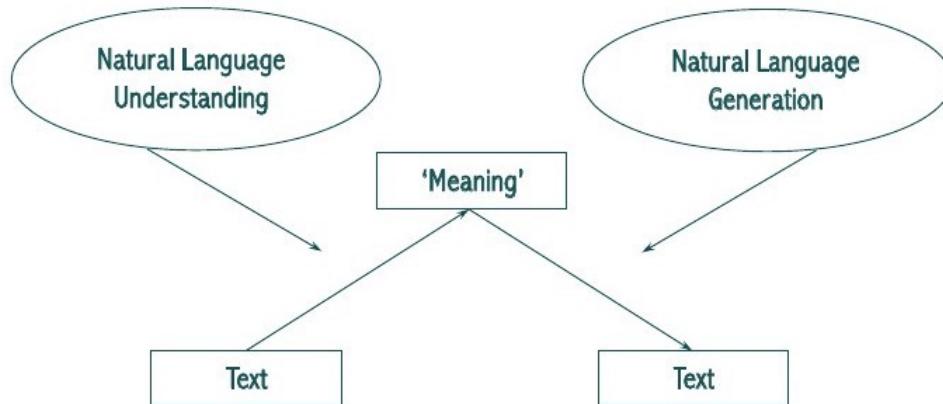
De tegenpool van NLU is Natural Language Generation(NLG)⁷. Het is ook een onderdeel van NLP en tracht de vertaalslag te maken van computer naar menselijke taal. NLG maakt dus de responses van een chatbot die verstaanbaar en grammaticaal correct zijn. Het startpunt voor NLU is het eindpunt voor NLG en omgekeerd: het eindpunt van NLU is het startpunt van NLG. (Kumar, 2018; Reiters & Dale, 2000).

Figuur 2.5 geeft schematisch weer hoe de drie concepten met elkaar verbonden zijn. Het toont eveneens dat “Natural Language Understanding” in het proces voor “Natural Language Generation” komt.

⁶In deze paper worden zowel de volledige term als de afkorting gebruikt

⁷In deze paper worden zowel de volledige term als de afkorting gebruikt

NLP = NLU + NLG



Figuur 2.5: Structuur Natural Language Processing

Source: <https://towardsdatascience.com/nlp-vs-nlu-vs-nlg-know-what-you-are-trying-to-achieve-nlp-engine-part-1-1487a2c8b696>

2.1.8 Conversational Search

Zoovu beschrijft conversational search als volgt:

Conversational search is the ultimate way to convert searchers into buyers by leveraging AI to optimize every step of the buyer's journey. Using AI to understand and predict what the customer needs to increase conversion and customer satisfaction. Brand and retailers gain insights about their customers and product performance across channels, regions, and languages.

— Zoovu, 2019

Conversational search kan vergeleken worden met een winkelbediende die helpt zoeken naar het ideale product voor de klant. Door gebruik te maken van AI gaan er, via een conversatie met de gebruiker, gerichte vragen aan de gebruiker gesteld worden om zo exact te vinden wat de gebruiker wil. Het kan dus gezien worden als een chatbot met expertise in het voorspellen en opzoeken van zaken die voldoen aan de wensen van de gebruiker.

Conversational search heeft vele voordelen ten opzichte van een klassieke zoekfunctie zoals een zoekbalk en filters. Enkele voorbeelden hiervan zijn:

- gebruiksvriendelijker
- persoonlijker
- distributie mogelijkheden
- betere gebruikersondersteuning

Gewone invulformulieren zijn veel minder aantrekkelijk dan wanneer er een conversatie wordt gevoerd. Conversational chatbots zijn veel gebruiksvriendelijker en kunnen veel persoonlijker gaan (door bijvoorbeeld het onthouden van de gebruiker zijn/haar naam).

Bovendien kunnen deze chatbots gemakkelijk extra informatie geven, terwijl bij een invulformulier dit niet steeds het geval is en er daardoor soms zeer lang achter gezocht moet worden op de FAQ-pagina. Chatbots kunnen (zoals eerder vermeld) over verschillende kanalen zoals Messenger en Slack verspreid worden waardoor de toegankelijkheid en klanttevredenheid stijgt. (kore.ai, 2019; Zoovu, 2019)

2.2 Datavisualisatie in het dagelijkse leven

2.2.1 Wetenschappelijke versus Informatie-Visualisatie

Datavisualisatie omvat zowel wetenschappelijke- als informatievisualisatie. Het is een overkoepelende term voor alles dat data omzet in visualisaties. Wetenschappelijke visualisatie gaat zich voornamelijk bezighouden met data met ruimtelijke eigenschappen en dus fysieke objecten vanuit de werkelijkheid voorstellen. Zaken zoals ruimtelijke spreiding en omvang van objecten vallen hieronder. Informatievisualisatie gaat voornamelijk over statistische voorstellingen van data. Dit zijn zaken zoals de aandelenkoers die geen zichtbare vorm hebben in de werkelijkheid. (Nagel, 2006; Zoss, 2019)

Deze taken zullen later van belang zijn voor het identificeren van goede visualisaties voor een chatbotflow.

2.2.2 Indeling van Visualisaties

Het groeperen van datavisualisaties kan op vele manieren. Shneiderman (1996) stelde een model op voor het onderverdelen van visualisaties. Deze zijn van toepassing op informatievisualisatie en niet op wetenschappelijke aangezien deze laatste, per wetenschappelijk domein, zijn eigen conventies gebruikt voor het visualiseren van data⁸. Het model stelt een Type by Task Taxonomy (TTT) voor waarbij de types gegroepeerd worden in functie van wat men wil onderzoeken. De volgende zeven groepen van datatypes worden bekomen:

- 1D/Lineair
- 2D/Planair
- 3D/Volumetrisch
- Tijdsgebonden
- nD/Multidimensionaal
- Boom
- Netwerk

⁸vanaf hier wordt er steeds van informatie visualisatie uitgegaan, tenzij anders aangegeven

1D (of lineaire datatypes) worden omschreven als een geordende lijst van items op basis van een kenmerk. Deze items bestaan uit een enkele lijn tekst. Op zulke data worden vaak taken zoals aantal item berekend, items die aan een bepaalde voorwaarde voldoen gefilterd of het verschil met een vorige versie gemeten. Voorbeelden van zulke data zijn bijvoorbeeld alfabetische lijsten of tekstdocumenten.

2D (of planaire) types worden voornamelijk gebruikt voor mappen van data over een totaal oppervlak. Zaken zoals blauwdrukken en geografische kaarten vallen hieronder. De taken voor deze data zijn het onderzoeken van aanliggende items en van de invloeden en verbanden die ze onderling delen.

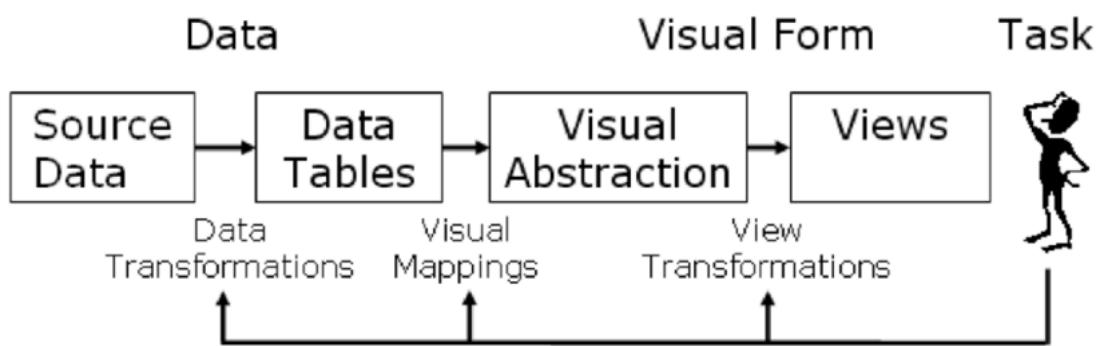
3-dimensionale data stellen objecten vanuit de werkelijkheid voor, met andere woorden objecten met een volume. De taken gaan niet enkel de aanliggende items links en rechts bestuderen maar ook die onder en boven, en binnen en buiten. Een voorbeeld van zo een visualisatie is het mappen van het menselijk lichaam naar een 3-dimensionaal model.

Multidimensionale datatypes met n attributen kunnen gezien worden als punten in een n-dimensioneel stelsel. De taken omvatten verbanden vinden tussen deze attributen en de invloeden die ze op elkaar hebben onderzoeken, kortom: statistische informatie onderzoeken.

Tijdsgebonden datatypes zijn gelijkaardig met 1D datatypes maar met als verschil dat het een start- en eindpunt heeft en dat items kunnen overlappen. De taken die hierop worden uitgevoerd staan in functie van tijd.

Boom gestructureerde datatypes bestaan uit items die gelinkt zijn aan hun ouder. De uitzondering hierop is de root, het startpunt van de boom. De structuur is wat een boom interessant maakt en de taken hierop onderzoeken de ouder-kind relaties en de algemene eigenschappen zoals het aantal niveau's van de boom.

De laatste groep, de *netwerken*, zijn in zekere zin gelijkaardig aan bomen: de items in de dataset zijn hier ook onderling verbonden, maar de regels zijn bij netwerken minder streng als die bij bomen. Een netwerk moet geen start item (root) hebben en mag cykels hebben. Taken zoals kortste pad en aantal paden kunnen hierop uitgevoerd worden.



Figuur 2.6: Het proces voor de creatie van informatie visualisaties

Source: https://www.researchgate.net/figure/This-schematic-diagram-model-represents-the-information-visualization-process_fig2_3260983885

Naast de zeven datatypes beschreef Schneiderman (1996) ook 7 (hoofd)taken die door gebruikers uitvoerbaar moeten zijn op de visualisaties. Deze taken zijn:

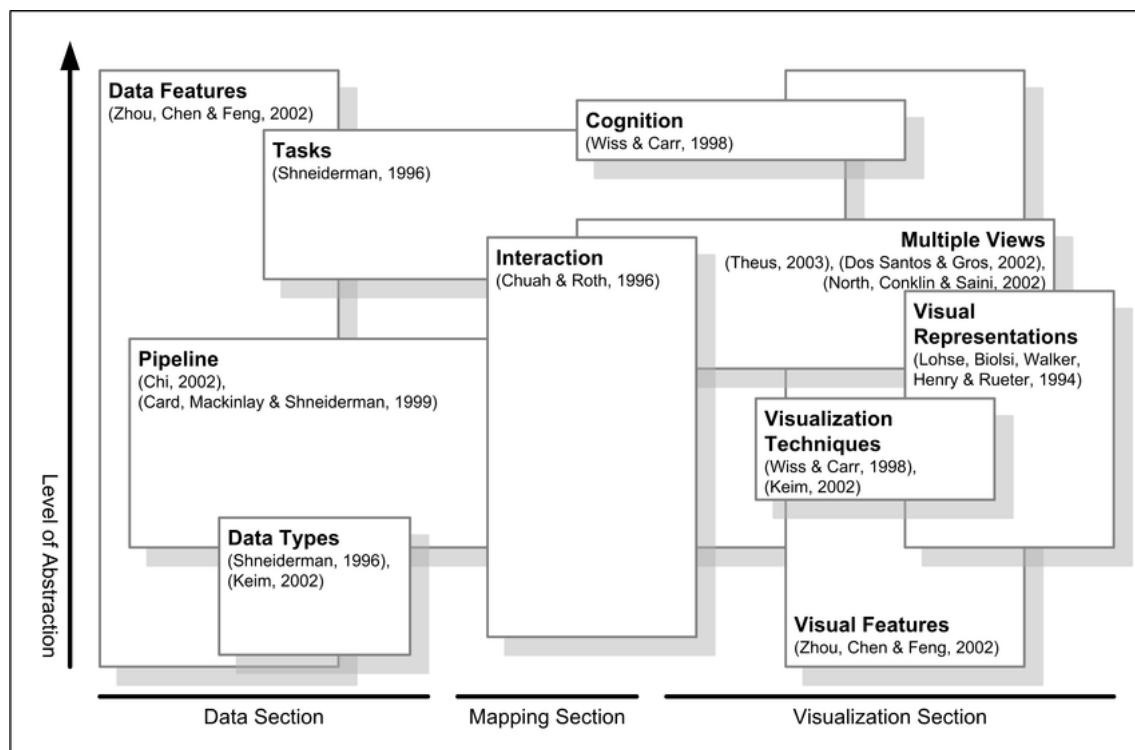
- Details-on-demand: dieper ingaan op een bepaalde groep of individueel item.
- Extract: bepaalde data uit de dataset halen voor individuele verwerking
- Filter: filteren van niet-relevante data
- History: een geschiedenis van de uitgevoerde taken bijhouden
- Overview: een overzicht creëren over de dataset
- Relate: relaties herkennen tussen data
- Zoom: in/uitzoomen naar gelang de interesse in de data

Verscheidene modellen zijn reeds opgesteld voor het beschrijven en indelen van visualisaties. Onderzoekers Sigmar-Olaf Tergan en Tanja Keller hebben een overzicht opgesteld van enkele opmerkelijke modellen en deze opgedeeld volgens abstractie en onderdeel van het visualisatie proces diagram van Card, Mzckinlay en Schneiderman (Figuur 2.6). (Tergan, 2005)

Het visualisatieprocesmodel toont de transformatie van ruwe data tot een visualisatie en de tussenliggende fases. (Card e.a., 1999)

De modellen overlappen regelmatig wat betekent dat ze elkaar aanvullen (Figuur 2.7). Daniel Keim heeft bijvoorbeeld een herwerk model van Schneiderman waarbij hij onder andere de tijdsgebonden data onder ééndimensionale data stekt, algoritmes en software als een nieuw datatype introduceert en bomen en netwerken samenneemt onder de nieuwe naam "Hierarchies & Graphs". (Keim, 2002)

Deze modellen worden niet verder besproken en vallen buiten de scope van deze paper.



Figuur 2.7: Samenvattend model voor informatie visualisatie

Source: https://www.researchgate.net/figure/Interrelationship-of-information-visualization-models-in-information-visualization-model_fig1_221520307e

2.3 Chatbots in het dagelijkse leven

Hoewel vandaag veel populairder, bestaan Chatbots al meer dan vijftig jaar⁹. Er zijn heel wat frameworks op de markt voor het maken van chatbots.

Enkele voorbeelden hiervan zijn:

- IBM Watson Assistant¹⁰
- Dialogflow¹¹
- Amazon Lex¹²
- Microsoft Bot Framework¹³
- Rasa¹⁴

De kenmerken en opties voor de chatbots in volgende secties komen rechtstreeks uit de bijhorende documentatie van de chatbot frameworks.

2.3.1 IBM Watson Assistant

Conversational search is the ultimate way to convert searchers into buyers by leveraging AI to optimize every step of the buyer's journey. Using AI to understand and predict what the customer needs to increase conversion and customer satisfaction. Brand and retailers gain insights about their customers and product performance across channels, regions, and languages.

— IBM, 2020j

Watson Assistant is een IBM-product dat gebruikers de mogelijkheid geeft om Virtuele Assistants te creëren op enterprise-niveau. Er wordt veel aandacht gegeven aan NLU en het geeft de mogelijkheid aan de gebruikers om de informatie die door hun chatbots verzameld wordt, bij te houden in een private cloud. Dit laatste is een kenmerk dat dat uniek is aan Watson Assistant. (IBM, 2020m; Rouse, 2018)

Watson ondersteunt volgende channels om te communiceren met de chatbot:

- Facebook Messenger
- Intercom
- Preview Link
- Slack
- Voice Agent

⁹eerste chatbot, ELIZA gecreëerd in 1966

¹⁰ibm.com/cloud/watson-assistant

¹¹dialogflow.com

¹²aws.amazon.com/lex

¹³dev.botframework.com

¹⁴rasa.com

Indien de gebruiker een andere mogelijkheid wenst, kan deze ook een eigen zelfgemaakt kanaal maken. (IBM, 2020a) Het is mogelijk om zowel spraak als tekst input door te geven aan deze kanalen. (IBM, 2020m)

Daarnaast ondersteunt het framework 13 talen waaronder Nederlands, Frans, Duits en Spaans. (IBM, 2020h)

IBM maakt de claim zich te onderscheiden van de concurrenten door volgende eigenschappen:

- Om verduidelijking vragen wanneer de chatbot meerdere antwoorden terug krijgt
- Het doorgeven naar een menselijke assistent wanneer hier achter wordt gevraagd
- Multitasking door context van de conversatie bij te houden

(IBM, 2020m)

Volgende SDK's¹⁵ worden ondersteund:

- Android
- Go
- Java
- Node.js
- Python
- Ruby
- .NET
- Salesforce
- Swift
- Unity

(IBM, 2020l)

Enkele gebruikers van Watson Assistant zijn:

- Autodesk
- Creval
- Humana

(IBM, 2020i)

Figuur 2.8 toont de grafische gebruikersinterface van Watson Assistant. De interface is als volgt opgebouwd: links staat de navigatiebalk en rechts de inhoud van de geselecteerde optie in de navigatiebalk. Wanneer er rechtsboven op “Try it” wordt geklikt verschijnt er een chat venster waar de chatbot getest kan worden op reeds toegevoegde functionaliteit.

Watson Assistant is het chatbot framework waar deze paper zich gaat op focussen.

¹⁵software development kit

The screenshot shows the IBM Watson Assistant interface with the 'My first skill' project selected. The 'Intents' tab is active, displaying a list of intents. The interface includes a top navigation bar with 'Save new version', 'Try it', 'Create intent', and other options. Below the navigation is a search bar and a table titled 'Examples 11'. The table columns are 'Modified' (with a downward arrow), 'Description', and 'Examples'. The rows list the following intents:

Modified	Description	Examples
a month ago	#ApplicationAccess	13
a month ago	#Bored	9
a month ago	#BYOD	9
a month ago	#Goodbye	12
a month ago	#Greetings	11
a month ago	#GenerateToken	7
a month ago	#ResetPassword	6
a month ago	#SupplierOnboarding	7

At the bottom right, there is a page navigation section showing 'Showing 1-8 of 8 intents' and arrows for navigating through the pages.

Figuur 2.8: IBM Watson Assistant gebruikersinterface

2.3.2 Dialogflow

Dialogflow is Google's oplossing voor een chatbot framework.

Google verdeelt integraties in vier groepen:

- Dialogflow built-in integrations
- Google-provided open source integrations
- Partner integrations
- Independent integrations

(Google, 2020b)

De eerste groep, *Dialogflow built-in integrations*, wordt volledig ondersteund door Dialogflow en kan geconfigureerd worden in de Dialogflow console:

- Actions on Google and Google Assistant
- Dialogflow phone gateway
- Dialogflow Web Demo
- Hangouts Chat
- Facebook Messenger
- LINE
- Slack
- Telegram

(Google, 2020b)

De *Google-provided open source integrations* zijn open source integraties die gemaakt zijn door Google maar die niet rechtstreeks vanuit Dialogflow geconfigureerd kunnen worden. Deze integraties geven een gebruiker de mogelijkheid om eigen integraties te schrijven waarbij deze open source integraties als basis dienen. De lijst van deze integraties:

- Kik
- Skype
- Spark
- Twilio IP Messaging
- Twilio (Text Messaging)
- Twitter
- Viber

(GoogleCloudPlatform, 2020)

De derde groep, *Partner integrations*, omvat integraties die gemaakt zijn door Google partners. Deze integraties worden niet ondersteund door Google: voor problemen of vragen zullen de ontwikkelaars gecontacteerd moeten worden. Volgende integraties worden aangeboden:

- Genesys
- SignalWire
- Voximplant

De *Independent integrations* zijn integraties die op de Dialogflow API gebouwd zijn door niet-partnerorganisaties. Ook deze worden niet ondersteund door Google en er zal bij problemen contact gelegd moeten worden met de eigenaars. (Google, 2020b)

Er worden in totaal 32 talen ondersteund waaronder drie variaties van Chinees¹⁶, vijf van Engels¹⁷ en 2 van Frans¹⁸. (Google, 2020c)

Dialogflow ondersteunt volgende SDK's:

- Go
- Java
- Node.js
- Python
- Ruby
- C#
- PHP

(Google, 2020a)

Dialogflow word onder andere gebruikt door:

- Domino's
- Comcast
- KLM
- Mercedes-Benz

(Dialogflow, 2020)

De gebruikersinterface is gelijkaardig aan die van Watson Assistant en ziet er als volgt uit (Figuur 2.9): links staat de navigatiebalk, in het midden het werkveld ()waar er in dit voorbeeld een intent wordt aangemaakt) en rechts een chatkanaal om de chatbot te testen.

2.3.3 Amazon Lex

Amazon's chatbot framework, Amazon Lex, is een service die deel uitmaakt van Amazon AI. Het gebruik dezelfde technologie die ook achter hun virtuele assistent *Alexa* zit en ondersteunt, net als bovenstaande frameworks, tekst en spraakinvoer. (Amazon, 2020c; Rouse, 2017)

¹⁶Kantonees, Versimpeld en Traditioneel

¹⁷Australië, Canada, Groot-Brittannië, India en Verenigde Staten

¹⁸Canada en Frankrijk

The screenshot shows the Dialogflow web interface. At the top, there's a navigation bar with 'Dialogflow' on the left and a search bar on the right. Below the navigation bar, the main area is titled 'Sherlock' with a status of 'Open'. There are several tabs: 'Intents' (selected), 'Entities', 'Knowledge', 'Fulfillment', 'Integrations', 'Training', 'Validation', 'History', and 'Analytics'. On the left, there's a sidebar with 'Solve-Mystery' selected. The main content area is divided into sections:

- Events:** A table with rows for 'He died last night...', 'Someone killed her', 'Someone died', 'Who didn't?', and 'Sherlock we need your help!!!'.
- Training phrases:** A table with rows for 'Add user expression' and 'Adjective expression'.
- Responses:** A table with rows for 'I know who did it was him!!!', 'There was not one but two murderers your friend Michael and his wife.', 'I know who I was, but I'll tell you in private, my dear Watson', and 'Enter a test response variant'.
- Fulfillment:** A table with rows for 'Text Response' and 'ADD RESPONSES'.
- Diagnostic info:** A table with rows for 'INTENT Solve-Mystery', 'ACTION Not available', and 'USER SAYS Sherlock we need your help', 'DEFAULT RESPONSE There was not one but two murderers your friend Michael and his wife.', and 'COPY CURL'.

At the bottom, there are buttons for 'SAVE', 'Cancel', and 'Try it now'.

Figuur 2.9: Dialogflow gebruikersinterface

Amazon Lex can bereikt worden op mobiele applicaties door gebruik te maken van AWS SDK's of op volgende "Messaging Platforms":

- Kik
- Twilio
- Facebook
- Slack

(Amazon, 2020e, 2020f)

Deze lijst is uitbreidbaar met custom channels wanneer de API het juiste formaat teruggeeft. Het opzetten van zulke kanalen vereist programmeerkennis. (Khan, 2017)

Op dit moment ondersteunt Amazon Lex enkel Engels¹⁹. (Amazon, 2020a)

Volgende SDK's zijn beschikbaar voor Lex:

- Android
- JavaScript
- iOS
- Java
- .NET
- Node.js
- PHP
- Python
- Ruby
- Mobile Web
- React Native

(Amazon, 2020b)

Lex volgt dezelfde indeling voor de gebruikersinterface als de twee bovenstaande frameworks (Figuur 2.10). Amazon Lex hanteert de term "Slot types" in plaats van "entities" maar het heeft hetzelfde doel en functies. (Amazon, 2020d)

Enkele klanten van Amazon die Amazon Lex gebruiken zijn:

- NASA
- American Heart Association
- Dynatrace
- Rubrik

(Amazon, 2020c)

¹⁹US English

The screenshot shows the AWS Lambda interface for the BookTrip bot. The top navigation bar includes 'Services' (selected), 'Latest', 'Resource Groups', 'AWS Lambda', 'Editor', 'Settings', 'Channels', and 'Monitoring'. The main area is divided into several sections:

- Intents:** Shows 'BookHotel' with a dropdown for 'Latest' or 'Previous'. Below it are 'Sample utterances' (e.g., 'I would like to book a flight.', 'Book a (Nights) night stay in (Location)', 'I want to make hotel reservations', 'Book a hotel'), and 'Lambda initialization and validation' (checkbox checked).
- Slot types:** Lists 'RootTypeValues' and 'Error Handling'.
- Slots:** A table showing four slots: 'Location' (Required, Priority 1, Type 'Location', Value 'e.g. AMAZON.US_CITY'), 'CheckinDate' (Required, Priority 2, Type 'CheckinDate', Value 'Built-in'), 'Nights' (Required, Priority 3, Type 'Nights', Value 'Built-in'), and 'RoomType' (Required, Priority 4, Type 'RoomType', Value 'RoomTypeValues').
- Confirmation prompt:** Contains a checkbox for 'Confirmation prompt' and a message placeholder: 'Okay, I have you down for {Nights} night stay in {Location}, starting {CheckinDate}. Shall I book the reservation?' with a note '(Cancel if the user says "no")'.
- Fulfillment:** Options include 'AWS Lambda function' (radio button selected) and 'Return parameters to client'.
- Response:** Includes 'Add Message' and 'Enable response card' (checkbox checked).

A preview window at the top right shows a conversation transcript:

```

book a hotel
What city will you be staying in?
New York
What day do you want to check in?
Friday
How many nights will you be staying?
5
What type of room would you like, queen, king or deluxe?
deluxe

```

Below the preview are sections for 'Clear chat history', 'Inspect response', 'Dialog State: Commitment', and 'Intent BookHotel' (Stats: 4/4, CheckinDate: 2020-04-03, Location: New York, Nights: 5, RoomType: deluxe).

Figuur 2.10: Amazon Lex gebruikersinterface

2.3.4 Microsoft Bot Framework

Ook Microsoft maakt het mogelijk om bots te creëren. Microsoft Bot Framework is Microsoft's uitgebreide framework voor het ontwikkelen van chatbots met ondersteuning van LUIS. Microsoft LUIS is verantwoordelijke voor de NLU en ondersteunt 11 talen²⁰, net als text en voice input(Figuur 2.11). (Microsoft, 2019c)

Er is soms verwarring over wat het verschil is tussen Microsoft Bot Framework en Azure Bot Service. Ook op de documentatiepagina's van Microsoft is het verschil moeilijk te vinden en worden de termen samen en door elkaar gebruikt. Deze paper gaat uit van het onderscheid dat gemaakt wordt in Figuur 2.11. De assumptie is als volgt: Azure Bot Service is een onderdeel van het Bot Framework en houdt zich uitsluitend bezig met het maken van de chatbot zelf, terwijl Bot Framework ook alle functionaliteiten biedt die niet rechtstreeks de structuur van de bot beïnvloeden. Dit zijn zaken zoals webhooks, programmaticke taken en (externe) kennis.

Microsoft vermeldt in de documentatie dat volgende channels worden voorzien:

- Cortana
- Direct Line
- Direct Line (Web Chat)
- Email
- Facebook
- GroupMe
- Kik
- Skype
- Skype Business
- Slack
- Teams
- Telegram
- Twilio

(Microsoft, 2019b)

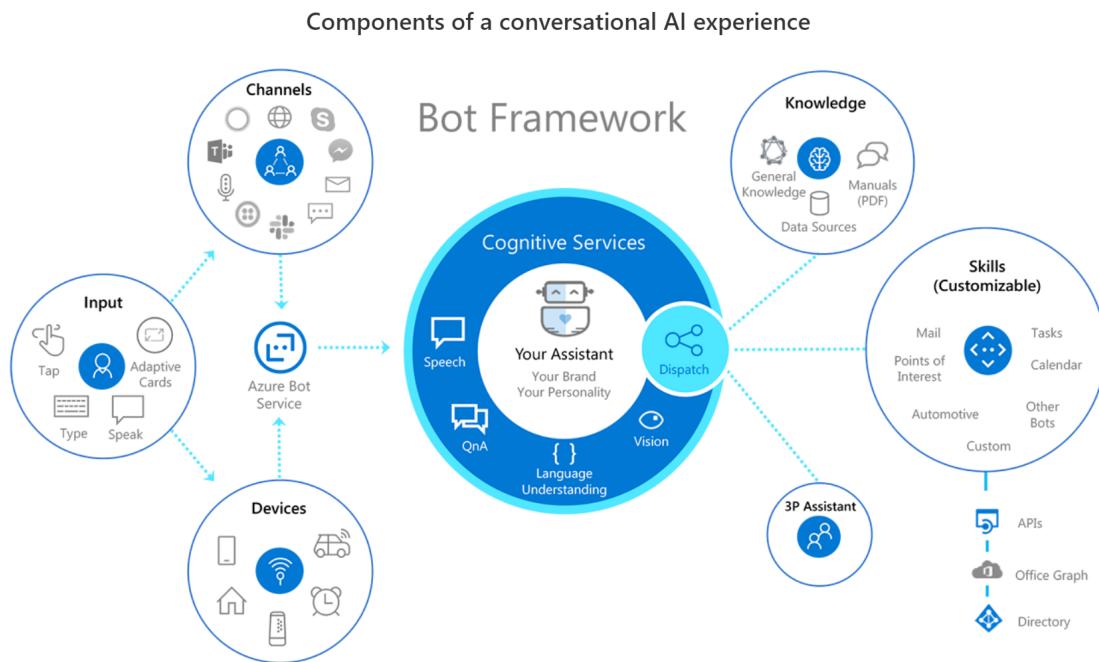
De github van Bot Framework SDK vermeldt aanvullend:

- Exchange
- Line
- Alexa²¹
- Google Home²¹
- Google Hangouts²¹
- Cisco Webex²¹
- Console²¹

(Microsoft, 2020a)

²⁰gedeeltelijke of volledige ondersteuning

²¹niet ondersteund door Microsoft



Figuur 2.11: Overzicht Microsoft Bot Framework

Source: <https://dev.botframework.com>

Naast deze bovenstaande channels is het ook mogelijk om eigen kanalen aan te maken.
(The Bot Framework Team, 2017)

Chatbots in het Microsoft Bot Framework kunnen gebouwd worden met volgende SDK's:

- JavaScript
- TypeScript
- C#
- Java²²
- Python²²

(Microsoft, 2019a)

In tegenstelling tot eerder besproken bot frameworks, voorziet Microsoft geen grafische gebruikersinterface voor het ontwikkelen van chatbots. Dit kan een struikelblok zijn voor gebruikers met minimale programmeerkennis.

Enkele gebruikers van Azure Bot Service zijn:

- Adobe
- UPS
- La Liga
- Progressive

²²nog in ontwikkeling

(Microsoft, 2020b)

2.3.5 Rasa (Open Source)

In general, open source refers to any program whose source code is made available for use or modification as users or other developers see fit. Open source software is usually developed as a public collaboration and made freely available.

— Rouse, 2009

Rasa is het eerste open source chatbot framework in deze lijst. De software is volledig afhankelijk van de community die meer dan 300 contributers heeft. Rasa voorziet drie versies van hun framework:

- Rasa Open Source
- Rasa X
- Rasa Enterprise

Rasa Enterprise is, in tegenstelling tot de eerste twee opties, niet gratis en de prijs is afhankelijk van elke individuele situatie. (Rasa, 2020f)

Rasa voorziet ondersteuning voor volgende channels maar maakt het ook gemakkelijk om zelf een kanaal te maken of op een eigen website te plaatsen:

- Cisco Webex
- Google Hangouts
- Mattermost
- Microsoft Bot Framework
- Messenger
- RocketChat
- Slack
- Telegram
- Twilio

(Rasa, 2020d)

Rasa biedt de mogelijkheid om zelf talen toe te voegen en te trainen. Er zijn reeds 8 talen gedefinieerd die kunnen worden geïmporteerd. (Rasa, 2020c)

De Rasa SDK is gebouwd op Python. Kennis van de programmeertaal is een groot voordeel wanneer je kiest voor Rasa. Het is ook mogelijk om Rasa chatbots te ontwikkelen met andere programmeertalen dan Python, door gebruik te maken van third-party software zoals Articulate. (discover.bot, 2019; Rasa, 2020g)

Zoals alle bovenstaande frameworks, ondersteunt Rasa zowel text als voice input. (Rasa, 2020e)

	Watson Assistant	DialogFlow	Lex	Bot Framework	Rasa
invoer	tekst en spraak	tekst en spraak	tekst en spraak	tekst en spraak	tekst en spraak
# ondersteunde talen	13	32	1	11	8
# ondersteunde SDK's	10	7	11	3	1
# ondersteunde kanalen	5	15	4	13	9
eigen kanaal	ja	ja	ja	ja	ja
(G)UI	ja	ja	ja	nee	nee

Tabel 2.1: Overzicht van chatbot frameworks

Volgende bedrijven maken gebruik van Rasa voor het ontwikkelen van chatbots:

- Lemonade
- Adobe
- Orange
- Engie
- N26

(Rasa, 2020a, 2020b)

2.3.6 Samenvatting

In tabel 2.1 staan eerder besproken aspecten van de chatbot frameworks samengevat. Er worden al snel enkele dingen duidelijk. Wanneer men een groot aanbod aan talen wil geven is Amazon Lex niet de beste keuze. Als men weinig tot geen (programmeer)kennis heeft over het maken van chatbots of simpelweg liever wenst te werken met een grafische gebruikersinterface, is het interessanter om eerder voor de eerste drie opties te kiezen dan voor de laatste twee.

De keuze zal afhangen van heel wat zaken. Enkele voorbeelden hiervan zijn:

- Gewenste software talen
- Ondersteuning voor talen en kanalen
- Flexibiliteit
- Eerdere ervaringen
- Private cloud of niet

2.4 Bestandsformaten voor Afbeeldingen

Om een visualisatie weer te geven, te printen, bewerkingen op uit te voeren... moet deze eerst bewaard worden in een bestand. De bestandsformaten die kunnen gebruikt worden om afbeeldingen bij te houden worden in deze sectie besproken. De bestandsformaten worden opgedeeld in twee groepen: vector- en rasterafbeeldingen.

2.4.1 Vectorafbeeldingen

Lexico geeft voor een vector in de wiskundige fysica de volgende definitie:

A quantity having direction as well as magnitude, especially as determining the position of one point in space relative to another.
— Lexico, 2020b

En specifiek voor computing:

Denoting a type of graphical representation using straight lines to construct the outlines of objects.
— Lexico, 2020b

Vectorafbeeldingen zijn dus bestanden die bestaan uit een reeks van wiskundige vergelijkingen (vectoren). Deze vergelijkingen zijn instructies die beschrijven hoe vormen gecreëerd moeten worden en samen een geheel vormen: de visuele weergave van de afbeelding. Deze vormen gaan van simpele lijnen tot zeer complexe structuren en kunnen zowel 2D als 3D zijn.

Vector afbeeldingen kunnen gemakkelijk gewijzigd worden door simpelweg de instructies voor de vormen aan te passen, waardoor ze ook schaalbaar zijn zonder aan kwaliteit te verliezen. Ze kunnen ook rasterafbeeldingen bevatten en worden omgezet naar rasterafbeeldingen. Voorbeelden van deze bestandtypes zijn:

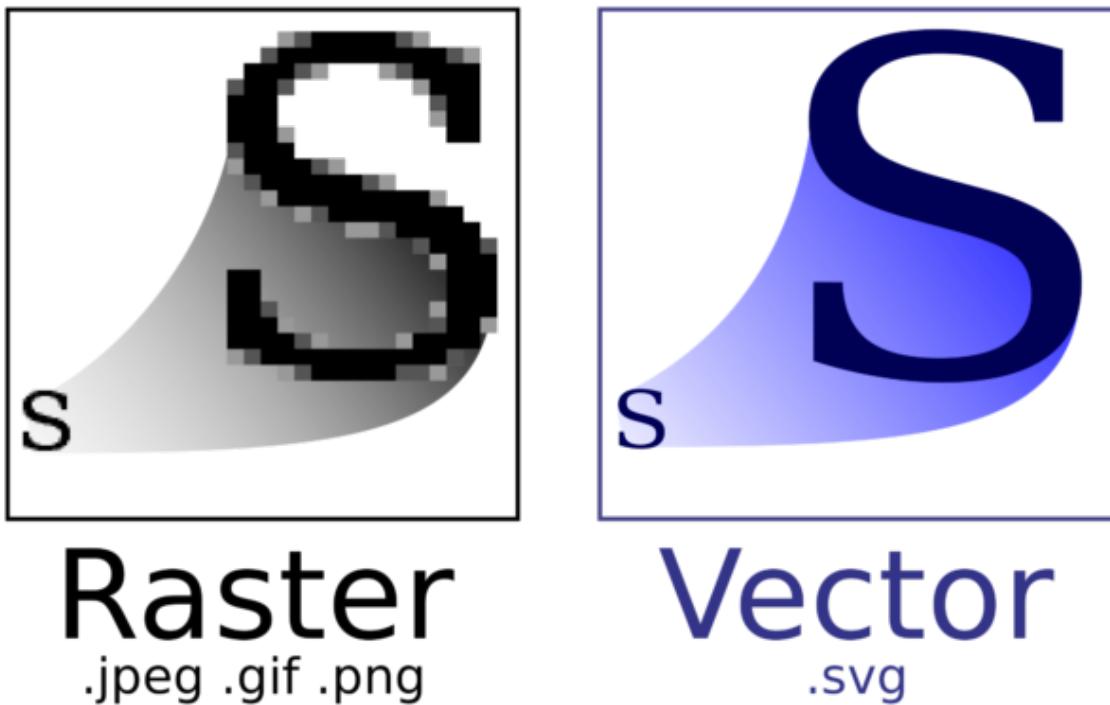
- CGM (Computer Graphics Metafile)
- ODG (OpenDocument Graphic File)
- SVG (Scalable Vector Graphics)
- XML (Extensible Markup Language)

(Dallwitz, 2010; Olympus Press, 2013; Rouse, 2006)

2.4.2 Rasterafbeeldingen

A ‘pixel’ (picture element) is the smallest component of a computer image. When an image is being manipulated by the computer, a pixel is regarded as a dot of a single colour (although when the image is displayed, each pixel may be made up of even smaller dots). An image is made up of pixels located on a rectangular grid.
— Dallwitz, 2010

In tegenstelling tot vectorafbeeldingen, bestaan *rasterafbeeldingen* niet uit wiskundige instructies die vormen beschrijven maar uit een verzameling van *pixels*, waarbij voor



Figuur 2.12: Het verschil tussen vector- en rasterafbeelding na inzoomen

Source: <https://iimagedesign.com/vector-vs-raster-for-logo-design/>

elke pixel de kleur wordt bijgehouden in een bitmap. Rasterafbeeldingen worden vaak bitmapafbeeldingen genoemd.

Hoewel niet onmogelijk is het zeer moeilijk om rasterafbeeldingen te schalen. Een rasterafbeelding kan dus zeer gemakkelijk onderscheiden worden van een vectorafbeelding door in te zoomen zoals in afbeelding 2.12 het geval is. Wanneer de afbeelding een “roosterstructuur” begint te vertonen, heeft men te maken met een rasterafbeelding.

De resolutie definieert het aantal pixels in de breedte en de hoogte en dus ook het totaal aantal pixels. Bijvoorbeeld voor een afbeelding met resolutie 1280×1024 is de afbeelding 1280 pixels breed, 1024 pixels hoog en bevat in totaal 1.310.720 pixels.

Voorbeelden van rasterafbeeldingen zijn:

- GIF (Graphic Interchange Format)
- JPG (Joint Photographic Experts Group)
- PNG (Portable Network Graphics)
- TIFF (Tag Image File Format)

(Dallwitz, 2010; Figma, 2020)

Vector	Raster
onafhankelijk van schaal en resolutie	afhankelijk van schaal en resolutie
beter voor gedetailleerde afbeeldingen met veel kleurverschillen zoals foto's	beter voor afbeeldingen met minder kleuren diversiteit zoals logo's en illustraties
makkelijk te converteren naar raster	zeer moeilijk te converteren naar vector
komen veel voor	komen minder voor
grote afbeelding = groot bestandformaat	grote afbeelding ≠ groot bestandformaat
gebaseerd op pixels	gebaseerd op wiskundige instructies

Tabel 2.2: Overzicht van vector- en rasterafbeeldingen

2.4.3 Vector versus Raster

Vector- en rasterafbeeldingen zijn zeer verschillend van elkaar en de keuze is steeds afhankelijk van de situatie waarin de afbeelding wordt gebruikt. Beiden hebben hun voor- en nadelen. In tabel 2.2 worden de verschillen tussen beide weergegeven.

In het algemeen wordt er gekozen voor rasterafbeeldingen als:

- het over foto's gaat
- het veel kleurendiversiteit bevat
- het er realistisch en aantrekkelijk moet uitzien
- er grafische effecten zoals "Glow" en "Drop Shadow" worden toegepast

Men kiest voor vectorafbeeldingen als:

- het gaat over logo's, illustraties en technische tekeningen
- in- en uitzoomen belangrijk is
- het op elke beeldformaat voorzien moet zijn
- de inhoud van de afbeelding makkelijk moet kunnen worden gewijzigd

(Gomez Graphics Vector Conversions, 2020; Vectr, 2020)

3. Methodologie

Het onderzoek start met een uitgebreide literatuurstudie die teruggevonden kan worden in Hoofdstuk 2. Deze literatuurstudie definieert de onderwerpen waarop dit onderzoek gaat focussen en bekijkt enkele belangrijke en veelvoorkomende chatbot frameworks die er vandaag op de markt zijn. De literatuurstudie tracht een duidelijke uitleg te geven over de aspecten van dit onderzoek in die mate dat deze verstaanbaar zijn voor een lezer met minimale kennis van informatica, en meer specifiek chatbots.

Vervolgens wordt er naar het export JSON-bestand gekeken dat teruggevonden wordt in Bijlage B. In Hoofdstuk 4 worden deze data gegroepeerd op basis van hun relevantie voor een flow en worden data zonder meerwaarde niet opgenomen in de visualisatie.

In Hoofdstuk 5 worden de visualisatietypes van Shneiderman (1996), zoals beschreven in Hoofdstuk 2, verder onderzocht en wordt er gekeken welke types niet gebruikt kunnen worden bij het visualiseren van de flow.

In Hoofdstuk 6 worden de individuele visualisaties bekeken van elk goedgekeurd visualisatietype uit Hoofdstuk 5 en worden deze visualisaties getest op dezelfde voorwaarden die in Hoofdstuk 5 ook gebruikt werden.

In Hoofdstuk 7 wordt er voor de resterende visualisaties gekeken welke voor- en nadelen ze hebben en bij welke omstandigheden de visualisaties niet langer representatief of duidelijk zijn. Er wordt gekeken welke manipulaties/bewerkingen er mogelijk zijn op deze visualisaties en welke van de voorgaande problemen opgelost kunnen worden door de visualisaties interactief te maken met deze bewerkingen.

In Hoofdstuk 8 wordt er een visualisatie gemaakt op basis van data van een IBM Watson chatbot.

In Hoofdstuk 9 worden de conclusies uit vorige hoofdstukken opgeliist en samengevat om een antwoord te geven op de onderzoeks vragen uit Hoofdstuk 2.

3.1 Definitie en Requirements Chatbotflow

Deze paper gaat op zoek naar de mogelijke visualisaties van een chatbot. De flow van zo een chatbot wordt in deze sectie omschreven en er worden afspraken gemaakt over de minimale eisen waaraan een chatbotflow moet voldoen.

Zoals in Hoofdstuk 2 besproken is bestaat een chatbot uit een of meerdere dialogs waarbij elke dialog wordt opgemaakt uit meerdere nodes. Dit onderzoek definieert een chatbot als volgt:

Een chatbotflow is een visuele vertoning van één of meerdere dialogs die op hun beurt bestaan uit één of meerdere nodes en waarbij de ouder-kind relatie visueel waarneembaar is.

De chatbot visualisatie moet dus:

- minstens één dialog hebben
- elke dialog heeft een root node
- elke node is visueel verbonden met zijn parent¹

De chatbot nodes moeten voorzien worden van de nodige informatie om zowel zichzelf te kunnen identificeren als uitleg over de situatie in de huidige node te kunnen geven. De volgende zaken zijn nodig voor het visualiseren van een chatbot node:

- de node moet een unieke naam hebben
- indien de node een trigger heeft², moet deze raadpleegbaar zijn

Deze sectie beschrijft niet hoe een visualisatie van een chatbotflow eruit moet zien, maar welke informatie deze moet laten zien.

3.2 Evaluatie Chatbot Data

Hoofdstuk 4 onderzoekt de structuur van het exportbestand en bekijkt de relevantie van de data in dit bestand. Aangezien niet alle data even belangrijk zijn wordt er gebruik gemaakt van de MoSCoW-methode. Deze zorgt ervoor dat elk dataveld in het bestand een prioriteit krijgt. De visualisaties van de goedgekeurde types na Hoofdstuk 5 zullen hierop getest worden.

¹tenzij deze zelf de root is

²hetzij een intent, hetzij een entity, hetzij een actie

De volgende prioriteiten kunnen worden toegekend:

- Must-have
- Should-have
- Could-have
- Won't-have

De *Must-haves* zijn de datavelden die in elke visualisatie verplicht zichtbaar moeten zijn voordat er “ingrijpende” operaties worden uitgevoerd. Dit betekent dat als een visualisatie wordt gegenereerd en er Must-haves ontbreken, deze visualisatie direct wordt afgekeurd. Aanvullend betekent dit niet dat alle velden van een Must-have aanwezig moeten zijn die aanwezig zijn in het startbestand. Een visualisatie die inzoomt op een bepaald deel van de flow moet enkel de data laten zien van de Must-haves die binnen dat deel vallen. Een voorbeeld hiervan kan zijn dat er bijvoorbeeld maar vanuit één root vertrokken wordt, dan zullen andere “bomen” niet zichtbaar zijn en zullen deze data niet getoond worden.

OPMERKING: hoveren en zoomen worden niet gezien als ingrijpende operaties.

Should-haves verschaffen de informatie die belangrijk is om de flow en de individuele nodes te begrijpen. De Should-haves vertellen meer over de werking van de chatbot. Een voorbeeld hiervan is bijvoorbeeld hoe er in een bepaalde node terecht wordt gekomen. Dit kan dan door het herkennen van een intent of door een JumpTo-instructie zijn. Deze informatie kan zowel direct getoond worden als tevoorschijn komen door een operatie uit te voeren.

De *Could-haves* zijn alle data die meer vertellen over de inhoud van de chatbot maar niet over de flow. Deze data zullen nooit direct zichtbaar zijn en zullen door het uitvoeren van een operatie, bijvoorbeeld iets openklikken, getoond kunnen worden.

Alle data die irrelevant is voor de visualisaties valt onder de *Won't-haves*: deze data worden nooit in een visualisatie getoond. Deze data kunnen bijvoorbeeld parameters bevatten die voor de Watson Assistant tool belangrijk zijn voor het inlezen van de data of vertrouwelijke informatie bevatten over de werking van het intern systeem.

3.3 Voorwaarden Visualisatietypes

In een vergelijkende studie worden er voorwaarden beschreven waaraan de onderzochte items trachten te voldoen en kan er op basis daarvan bepaald worden welke de beste zijn. Dit is niet doel van deze paper. Dit onderzoek gaat op zoek naar welke visualisaties kunnen gebruikt worden en in welke situaties. Aanvullend worden de bewerkingen bekeken die het mogelijk maken om visualisaties te gebruiken die op het eerste zicht niet relevant waren.

Bij het filteren van visualisaties wordt er eerst gekeken naar de grote types die besproken zijn in Hoofdstuk 2. De types moeten aan volgende voorwaarden voldoen om vatbaar te zijn voor verdere evaluatie:

- De visualisaties moeten de link tussen nodes visueel kunnen weergeven (ouder-kind relatie).
- De nodes moeten individueel kunnen bekijken worden en moeten beschouwd worden als een aparte entiteit.

Bovenstaande voorwaarden zijn specifiek om te bepalen of het type de flow correct kan voorstellen. Er moet echter nog een derde voorwaarde gecontroleerd worden wanneer aan de bovenste twee voorwaarden wordt voldaan. Deze extra voorwaarde is een algemene voorwaarde voor datavisualisaties:

- Elk type heeft een bepaald doel dat onderzocht moet worden. Als dit doel niet samenvalt met waar dit onderzoek naar op zoek is dan voldoet het type niet, ongeacht of er een visualisatie van een chatbotflow gemaakt kan worden die geldig zou zijn of niet.

Dit laatste wil zeggen dat, als het doel van een type bijvoorbeeld het op zoek gaan naar de verschillen tussen twee items in de dataset is, dit geen gepast type is omdat het een ander aspect van de data weergeeft dan het doel dat onderzocht wordt. Het is dus niet enkel van belang of het al dan niet mogelijk is om dit te visualiseren. In het geval dat het type “misbruikt” wordt, of de voorgestelde data kunnen leiden tot foutieve interpretaties, volstaat het niet.

4. Opbouw Watson Assistant Chatbot

In functie van dit onderzoek is er gebruik gemaakt van de online IBM tool voor het maken van een Watson Assistant chatbot. Het resultaat is een chatbot om de structuur van een Watson Assistant chatbot te onderzoeken en deze te gebruiken bij het maken van het prototype. Deze chatbot kan geëxporteerd worden naar een JSON-bestand. Dit bestand is in bijlage B opgesplitst in meerdere secties zodat deze gemakkelijk kan worden bestudeerd.

4.1 Algemene structuur van de chatbot

De basisstructuur, zoals gezien kan worden in Sectie B.1, bestaat uit volgende items:

- een lijst van intents
- een lijst van entiteiten
- metadata
- een lijst van (dialog-)nodes
- een lijst van counterexamples
- systeeminstellingen
- een veld voor toestemming van data verzamelen
- een veld voor de naam van de “skill”
- een veld voor taal van de chatbot
- een veld voor beschrijving van de chatbot

In de volgende secties komen al deze items aan bod en wordt er bepaald welke informatie er relevant is om te tonen en welke niet.

4.2 Chatbot Informatie

Deze sectie neemt alle datavelden samen die weergegeven worden in Sectie B.2, zijnde:

- metadata
- een lijst van counterexamples
- systeemininstellingen
- een veld voor toestemming van data verzamelen (“learning_opt_out”)
- een veld voor de naam van de “skill”
- een veld voor taal van de chatbot
- een veld voor beschrijving van de chatbot

Metadata kan beschreven worden als:

A set of data that describes and gives information about other data.
— Lexico, 2020a

De metadata zeggen hier iets meer over de API van de chatbot. Deze info is uiteraard niet relevant om te visualiseren en bijgevolg krijgen alle metadata het label *Won't-have*.

Counterexamples zijn eigenlijk Entities en Intents met als bedoeling irrelevante informatie te detecteren en zo deze te negeren. Stel dat je een cupcake chatbot maakt, dan zal deze weinig afweten over astronomische zaken. Uiteraard is deze ook niet relevant voor de visualisaties.

Voorbeelden van systeemininstellingen zijn spellingscontrole en chatbotsuggesties.

Het veld voor toestemming om data te verzamelen is een booleaanse waarde, die wanneer op “false” gezet wordt, IBM de toestemming geeft om gegevens te verzamelen van de chatbot. Dit is hier het geval. Ook dit zijn geen nuttige data om te visualiseren.

De counterexamples, systeemininstellingen en het toestemmingsveld behoren allemaal tot de *Won't-haves*.

De naam, taal en beschrijving zijn wel interessante zaken. De naam is een *Must-have* en moet aanwezig zijn bij elke visualisatie. De taal en beschrijving zijn iets minder belangrijk en behoren tot de *Could-haves*.

4.3 Intents

Zoals eerder besproken zijn intents een essentieel onderdeel van een chatbot. De intent-data in Sectie B.3 bestaat uit volgende datavelden:

- de naam van de intent
- een lijst van voorbeelden
- de beschrijving van de intent

Een intent is een belangrijk onderdeel voor een chatbot node en de werking ervan, zoals in Sectie 3.1 wordt vermeld. De identificatie van de intent, de naam, zal dus opgenomen moeten worden bij de visualisatie. Het valt onder de categorie van de *Should-haves*.

De beschrijving en de lijst van voorbeelden zeggen iets meer over de intent en niet over de node. De beschrijving zegt echter wel iets meer over de intent en wordt bijgevolg bij de *Could-haves* gerekend. De voorbeelden zullen vrijwel nooit van belang zijn. Wanneer er nood aan is kan er altijd een bijlage met opsomming worden voorzien. Zij vallen dus onder de *Won't-haves*.

4.4 Entities

Net als intents kunnen entities als een trigger dienen om een node te activeren en kunnen daarnaast ook gebruikt worden om informatie uit *slots* te labelen. Voor entities kunnen de volgende datavelden teruggevonden worden (Zie Sectie B.4):

- de naam van de entity
- een veld voor gebruik van “fuzzy matching”
- een lijst van waarden behorende tot deze entity

Zoals bij intents is ook de naam van belang voor de node. Bijgevolg wordt ook de naam van een intent als een *Should-have* beschouwd. De instelling voor het gebruik van fuzzy matching is niet van belang en valt onder de *Won't-haves*.

De waarden van een entity kunnen van belang zijn als er specifiek naar een bepaalde waarde wordt gekeken. Indien er geen specifieke waarde(n) van belang is/zijn, wordt deze lijst en al zijn items beschouwd als *Could-haves*. De rest van deze sectie gaat er van uit dat er wel sprake is van individuele waarden. De waarden bestaan uit:

- de naam van de waarde
- het type van de waarde
- een lijst van synoniemen/patronen

De naam is hier uiteraard de identificator en wordt toegevoegd aan de lijst van *Should-haves*. De lijst van synoniemen/patronen vertelt niets over de chatbot node maar wel over de entity en wordt gelabeld als een *Could-have*.

Er zijn drie entity types, namelijk:

- synoniem entity (“Synonym entity”)
- patroon entity (“Pattern entity”)
- systeem entity (“System entity”)

De eerste entity, de *synoniemen*, groepeert waarden die bij eenzelfde categorie horen en, zoals uit de naam afleidbaar is, voorziet de mogelijkheid om synoniemen voor deze waarden te geven. De *Patronen* maken gebruik van reguliere expressies om zaken zoals emails en telefoonnummers te zoeken in de gebruikersinput. De laatste categorie, de *systeem* entities, zijn entities die standaard worden meegeleverd door IBM. Deze entitytypes zijn niet belangrijk voor de visualisaties maar kunnen in sommige gevallen informatief zijn wanneer uit de naam niet afgeleid kan worden wat deze entity betekent en de node meer informatie vereist. In dat geval wordt het gelabeld als een *Should-have*, anders valt het onder *Could-have*.

De synoniemen of patronen zijn, zoals de lijst van voorbeelden bij intents, niet relevant voor de node en ook niet voor de entity. Zij worden dus ook het label *Won't-have* toegekend.

4.5 Dialog Nodes

De laatste groep, de *dialog nodes*, zijn de interessantste groep om te visualiseren. Alle data van deze categorie bevatten informatie die rechtstreeks van toepassing zijn op de nodes die gevisualiseerd worden.

4.5.1 Dialog Nodes Types

Een dialog node is steeds één van volgende types(IBM, 2020g):

- standard
- frame
- slot
- event handler
- response condition

Een *standard* type is, zoals de naam al zegt, het standaard type van elke node.

De *frame* node is een node met minstens één slot node als kind. Naast deze eigenschap is er geen verschil met een standaard node

Een *slot* node houdt informatie over een individueel slot¹ van de parent frame bij.

De *event handler* node definieert een event-afhandeling van een standard of frame node. Dit kan bijvoorbeeld gaan over wat er moet gebeuren als de gebruiker een response geeft die niet overeenstemt met wat de chatbot verwacht. Stel dat een chatbot vraagt voor hoeveel personen er moet worden gereserveerd en de gebruiker antwoordt met datum of gewoon geen aantal, dan moet zo een event zeggen wat de chatbot moet doen om verder te kunnen gaan.

Het laatste type is de *response condition*. Deze nodes worden gebruikt om antwoord aan

¹Zie hoofdstuk 2

een gebruiker te geven op basis van vooraf bepaalde voorwaarden. Dit kan bijvoorbeeld zijn om een meer gepersonaliseerd antwoord te geven of de informatie die een gebruiker gaf te bevestigen.

Al deze types zijn belangrijk om de dialog nodes te visualiseren. Het is echter niet van belang welke type een node heeft aangezien deze informatie van belang is voor de verwerking en niet voor de visualisatie. Ze worden tijdens het verwerken toch allemaal omgezet naar eenzelfde formaat. Het type is dus een *Won't-have*.

Van elk type kan er minstens één voorbeeld teruggevonden worden in Sectie B.5.

4.5.2 Standard (en Frame) Dialog Nodes

De volgende data kunnen teruggevonden worden voor *standaard en frame dialog nodes*:

- het type van de node²
- de titel van de node
- de output van de node³
- een verwijzing naar de parent (ID)³
- de trigger condities
- de ID van de dialog node
- een verwijzing naar de voorgaande sibling (ID)³

Zoals vermeld in Sectie 4.5.1 is het type niet van belang en zal in volgende secties ook niet meer besproken worden. De naam is wat de node identiteit geeft en is bijgevolg een *Must-have*. Hoewel er geargumenteerd kan worden dat het ID juist de identiteit geeft, is het ID niet iets wat de node betekenis geeft en deze wordt onder de *Won't-haves* gestoken.

De output kan vele formaten aannemen en deze worden niet individueel behandeld. Het type, of meerdere types, van deze output is van belang en wordt gelabeld als "Should-have" terwijl zaken zoals de responses en de "selection_policy" het label *Could-have* krijgen. Bij gebruik van response condition nodes ontbreekt de output en wordt dit vervangen door een veld "meta_date"⁴. De output wordt immers opgemaakt uit deze onderliggende nodes en deze zullen moeten worden opgevraagd om de output-informatie te verkrijgen. De metadata zijn uiteraard *Won't-haves*.

Zoals het type en de ID, zijn ook de verwijzingen naar zowel de parent(-ID) en voorgaande sibling(-ID) enkel van belang tijdens het visualiseren. Het is immers een ID van een bovenliggende of naburige node en dus zijn beide ook *Won't-haves*.

De condities om de nodes te triggeren zijn van cruciaal belang voor het navigeren in een chatbot, en bijgevolg ook voor een dialogflow. De condities worden beschouwd als *Must-haves*.

²zie Sectie 4.5.1

³Niet altijd aanwezig

⁴zie de laatste node in de bijlage

Er zijn enkele nodes die ook een veld “next_step” bevatten. Dit veld geeft instructies over wat er moet gebeuren na de response van de chatbot. Er kan aan de hand van dit veld direct worden verder gegaan of gesprongen worden naar een node dat geen kind is van de huidige node. Door middel van zo een sprong (“jump_to”) kunnen er lussen gemaakt worden of kan er naar andere dialogs genavigeerd worden. Als dit veld er niet is wordt er gewoon gewacht op de gebruiker. Dit veld is interessant voor het visualiseren van de flow maar de inhoud zelf moet niet gevisualiseerd worden en is een *Won’t-have*.

4.5.3 Slot Nodes

Een *Slot node* heeft de volgende datavelden:

- het type van de node²
- een verwijzing naar de parent (ID)
- de naam van de variabele
- de ID van de dialog node
- een verwijzing naar de voorgaande sibling (ID)³

Zoals besproken in de vorige sectie, zijn ID’s niet interessant voor het visualiseren en worden deze drie velden gelabeld als *Won’t-haves*. Ook de naam van de variabele is niet interessant en valt ook onder de *Won’t-haves*.

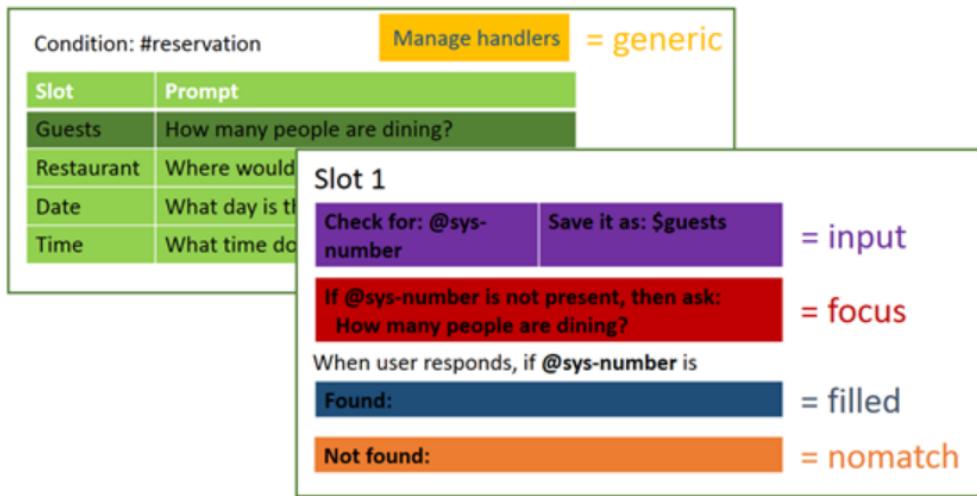
4.5.4 Event Handler Nodes

Event Handler nodes bevatten de volgende data:

- het type van de node²
- de output van de node
- een verwijzing naar de parent (ID)
- de context van het slot³
- de condities³
- de naam van het event
- de ID van de dialog node
- een verwijzing naar de voorgaande sibling (ID)³

(IBM, 2020g) De naam van het event is steeds één van volgende opties:

1. input
2. focus
3. filled
4. generic
5. nomatch



Figuur 4.1: Werking van event handler dialog nodes

Source: <https://cloud.ibm.com/docs/assistant-icp?topic=assistant-private-dialog-api>

In Figuur 4.1 worden de meeste node events getoond in volgorde waarin ze worden uitgevoerd. De *generic*-optie wordt niet getoond in de figuur maar wordt uitgevoerd tussen de *filled* en *nomatch* wanneer deze voorzien is. De *generic*-optie is ook de enige optie die naast slot nodes ook frame nodes als parent kan hebben.

Deze events zijn validaties voor het verkrijgen van informatie uit gebruikersinput en bijgevolg niet bepaald belangrijk voor het visualiseren van de flow. In sommige instanties kan het toch interessant zijn voor een gebruiker om te zien of er validaties gebeuren en welke deze zijn. Het label *Could-have* wordt toegewezen. De conditie verwijst naar het verwachte responstype⁵ en zegt op welk stuk informatie deze node controle doet. Het is enkel belangrijke dit mee te geven aan de visualisatie wanneer ook de “event_name” wordt opgevraagd en krijgt dus hetzelfde label, namelijk *Could-have*.

De context zegt over welk responstype het gaat en onder welke naam dit wordt bijgehouden. Doordat het responstype al wordt bijgehouden in de condities is dit niet meer van belang en kan de context als *Won’t-have* gelabeld worden.

De ID’s zijn ook hier niet van belang en worden weggelaten (*Won’t-Haves*). De output, dus de vraag om informatie, is niet van belang en wordt ook gelabeld als *Won’t-have*.

⁵entity, intent of context variabele

4.5.5 Response Condition Nodes

De *Response Condition nodes* bestaan uit volgende velden:

- het type van de node²
- de output van de node
- een verwijzing naar de parent (ID)
- de trigger condities
- de ID van de dialog node

De output van deze node is in feite een deel van de output van de ouder standard/frame node. Het type is dus van belang terwijl de rest van de output eerder optioneel is. Het type is een *Should-have* en de overige velden van de output zijn *Could-haves*.

De ID's, van zowel de parent als van de node zelf, zijn *Won't-haves* en de condities zijn *Should-haves*.

4.5.6 Overzicht datavelden chatbot

In Tabel 4.1 worden alle besproken velden weergegeven met hun corresponderende label.

Chatbot Informatie				
	Must have	Should have	Could have	Won't have
Metadata				X
Counterexamples				X
Systeeminstellingen				X
Toestemming datacollectie				X
Naam	X			
Beschrijving		X		
Taal			X	
Chatbot Intents				
	Must have	Should have	Could have	Won't have
Naam		X		
Voorbeelden				X
Beschrijving			X	
Chatbot Entities				
	Must have	Should have	Could have	Won't have
Naam		X		
Fuzzy Matching				X
Waardes	Naam	X		
	Type		X	
	Synoniemen/patronen		X	
Chatbot Nodes				
	Must have	Should have	Could have	Won't have
Algemeen	ID			X
	ID Parent			X
	ID Prev Sibling			X
	Type			X
Standard/Frame	Titel	X		
	Output (Type)		X	
	Output (Overige)			X
	Condities	X		
Slot	Variabele Naam			X
Event Handler	Output			X
	Context			X
	Condities		X	
	Naam Event		X	
Response Condition	Output (Type)	X		
	Output (Overige)			X
	Condities		X	

Tabel 4.1: Overzicht Watson Assistant exportdata

5. Datavisualisatie Types

In dit hoofdstuk worden de visualisatie-types die gedefinieerd zijn door Shneiderman (1996) vanuit Hoofdstuk 2 geëvalueerd en worden enkel degene die toepasbaar zijn gebruikt voor verder onderzoek. Nadien worden visualisaties van de goedgekeurde types verder bekeken. De termen visualisatie en datavisualisatie worden door elkaar gebruikt maar hebben dezelfde betekenis. Een visualisatie is immers altijd een voorstelling van data, of het nu bijvoorbeeld woorden of pixels zijn.

5.1 Types van Datavisualisatie

De types van datavisualisatie worden gegroepeerd in functie van de data die ze moeten weergeven. De beschouwde types zijn:

- 1D/Lineair
- 2D/Planair
- 3D/Volumetrisch
- Tijdsgebonden
- nD/Multidimensionaal
- Boom
- Netwerk

5.2 Lineaire Datavisualisaties

Lineaire of 1-dimensionale visualisaties zijn bedoeld om items op basis van één kenmerk te groeperen. Deze types zijn bedoeld voor bewerkingen zoals zoeken, filteren en het berekenen van aantallen. Een voorbeeld hiervan kan een tekstdocument zijn met een namenlijst van leerlingen in een bepaalde klas, waarbij het kenmerk de naam van de leerling is, of een map in de “File Explorer” van een Windows computer waarbij de naam van elk bestand binnen die map als kenmerk wordt gebruikt. (Shneiderman, 1996)

Er wordt maar aan één van de voorwaarden voldaan die gedefinieerd zijn in Hoofdstuk 3. Voor de eerste voorwaarde geldt dat de ouder-kind relatie voorgesteld moet worden. In het voorbeeld van de “File Explorer” zou dit bijvoorbeeld kunnen opgelost worden door mappen aan te maken. Een map stelt dan de node voor en alle items in die map zijn de kinderen van die node.

Hoewel nu aan de eerste voorwaarde wordt voldaan, wordt er automatisch niet aan de andere voorwaarde voldaan: de data worden gegroepeerd in een map waardoor de kinderen niet zichtbaar zijn en dus niet als individuele entiteiten gezien kunnen worden.

Er is een oplossing waarbij het wel mogelijk zou zijn om aan beide voorwaarden te voldoen. Door een tekstuele voorstelling van de chatbot te nemen kunnen zowel aan de eerste en tweede voorwaarden voldaan worden. Onderstaande codevoorstelling is hier een voorbeeld van. Er zou ook gewoon het geëxporteerde bestand genomen kunnen worden¹. De opzet van het onderzoek en van het prototype is juist om van dit bestand te starten en om een heldere weergave te geven van de flow. Deze oplossing wordt niet aanvaard.

```

node1:
    naam: "root1",
    kinderen: [
        node2:
            naam: "kind1",
            kinderen: [
                node4:
                    naam: "kleinkind1",
                    kinderen: [],
                node5:
                    naam: "kleinkind2",
                    kinderen: []
            ],
        node3:
            naam: "kind2",
            kinderen: [
                node6:
                    naam: "kleinkind3",
                    kinderen: []
            ]
    ]

```

¹zie verder

```
node7:  
    naam: "kleinkind4",  
    kinderen: []  
]  
]
```

Lineaire visualisaties kunnen dus niet gebruikt worden om een chatbotflow voor te stellen en worden niet verder onderzocht.

5.3 Planaire Visualisaties

Bij planaire of 2D data is de oriëntatie van de items van belang. Twee-dimensionale data bevatten zaken zoals geografische kaarten en plattegronden. Bij deze visualisaties is de positie van de items ten opzichte van elkaar van groot belang. Wanneer items zich binnen andere items bevinden wil dit zeggen dat ze hier deel van uit maken. Een voorbeeld hiervan is de positie van landen ten opzichte van elkaar (bijvoorbeeld Canada ligt ten noorden van de Verenigde Staten) en steden maken deel uit van het land waarbinnen ze vallen (bijvoorbeeld Gent is een Belgische stad omdat het binnen de grenzen van België valt).

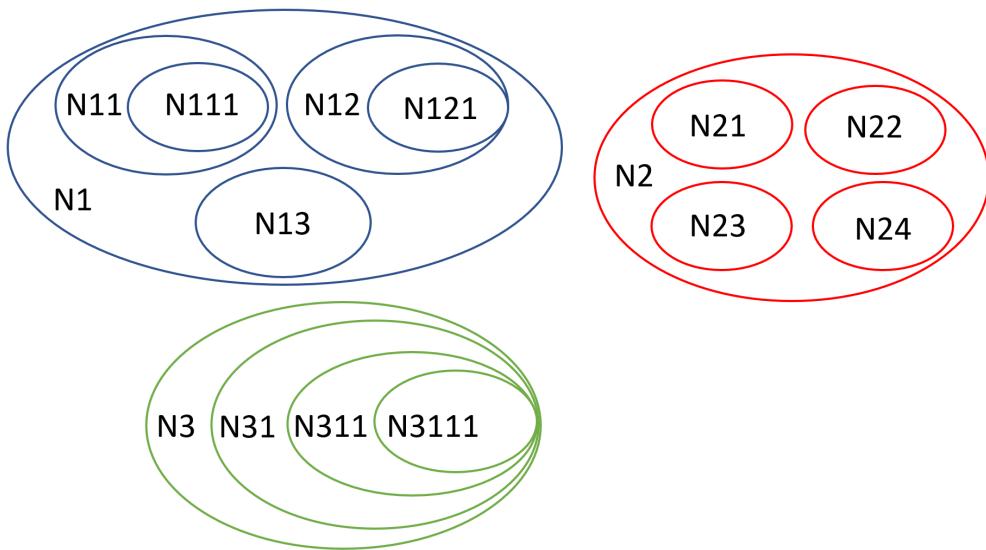
De eerste voorwaarde wordt vervuld wanneer men het bovenstaande principe benut en nodes nestelt in elkaar. Ook aan het tweede principe wordt voldaan aangezien men nodes individueel kan weergeven. Men zou op eerste zicht kunnen zeggen dat dit type geldig is voor het visualiseren van een flow. Figuur 5.1 toont hoe een flow gevisualiseerd zou kunnen worden met 2D. In deze visualisatie is de ouder-kind relatie duidelijk weergegeven door het in elkaar te nestelen van nodes.

Hoewel aan de twee eerste voorwaarden wordt voldaan is planaire visualisatie geen bruikbaar type. Zoals eerder besproken is de relatie tussen “siblings” van belang bij 2-dimensionale visualisatie. Dit zou betekenen dat er rekening moet gehouden worden met waar bijvoorbeeld nodes N1, N2 en N3 ten opzichte van elkaar liggen. Voor de flow van de chatbot maakt de volgorde van “siblings” geen verschil en kan deze visualisatie leiden tot een foute interpretatie. Daarnaast valt het ook op dat deze visualisatie heel snel onoverzichtelijk zou worden.

Het is duidelijk dat ook planaire visualisaties onbruikbaar zijn voor de visualisatie van de flows en deze worden niet meer verder bestudeerd.

5.4 Volumetrische Datavisualisaties

Volumetrische visualisaties tonen objecten vanuit de werkelijkheid. Het kan gezien worden als een uitbreiding van planaire visualisaties, waarbij er naast links, rechts, onder en boven ook rekening moet gehouden worden met voor en achter. Dit kan een uitdaging zijn bij visualisaties als er door overlap informatie niet langer zichtbaar is. 3D visualisaties hebben



Figuur 5.1: Voorbeeld van een voorstelling van een flow met een planaire visualisatie

dezelfde essentie als 2D: de positie is van groot belang, met de uitbreiding van een extra dimensie.

Ook bij 3D worden aan de eerste twee voorwaarden voldaan. Zoals bij 2D wordt er ook hier gebruik gemaakt van het nestelen van nodes om de relatie ouder-kind aan te duiden en kunnen nodes individueel getoond worden. Figuur 5.2 is een voorbeeld van hoe een chatbotflow voorgesteld kan worden met een 3-dimensionale visualisatie. De grote bollen zijn twee nodes en deze hebben binnenin kinderen: de eerste (van links) heeft drie en de tweede heeft vijf kinderen.

Echter, zoals bij 2-dimensionale visualisaties is hier terug de onderlinge ligging tussen nodes het probleem waardoor er aan de laatste voorwaarde niet wordt voldaan. De weergave wordt zelfs sneller onoverzichtelijk, zeker wanneer er geen operaties zoals het roteren van de visualisatie mogelijk zijn.

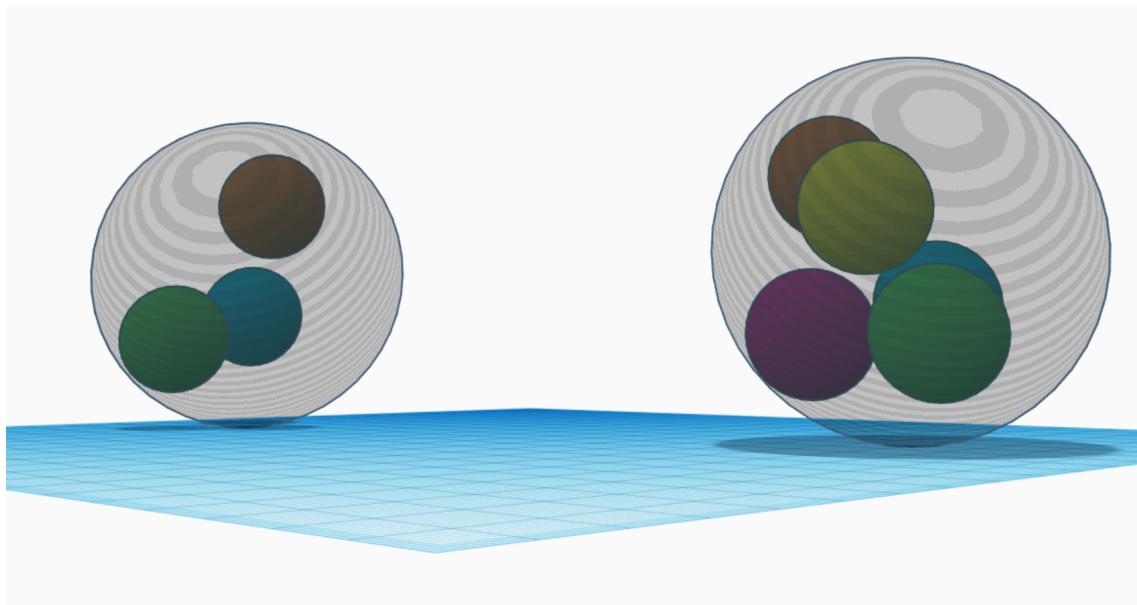
Zoals de twee voorgaande types, is ook dit type onbruikbaar.

5.5 Tijdsgebonden Datavisualisaties

Tijdsgebonden visualisaties gaan over data waarbij tijd een hoofdrol speelt. Historische presentaties, zoals tijdlijnen en planningen, zijn hier goede voorbeelden van. Deze visualisaties zijn zeer gelijkaardig aan 1D visualisaties met twee verschillen:

- items hebben een start- en eindtijd
- items kunnen overlappen

(Shneiderman, 1996)



Figuur 5.2: Voorbeeld van een voorstelling van een flow met een volumetrische visualisatie

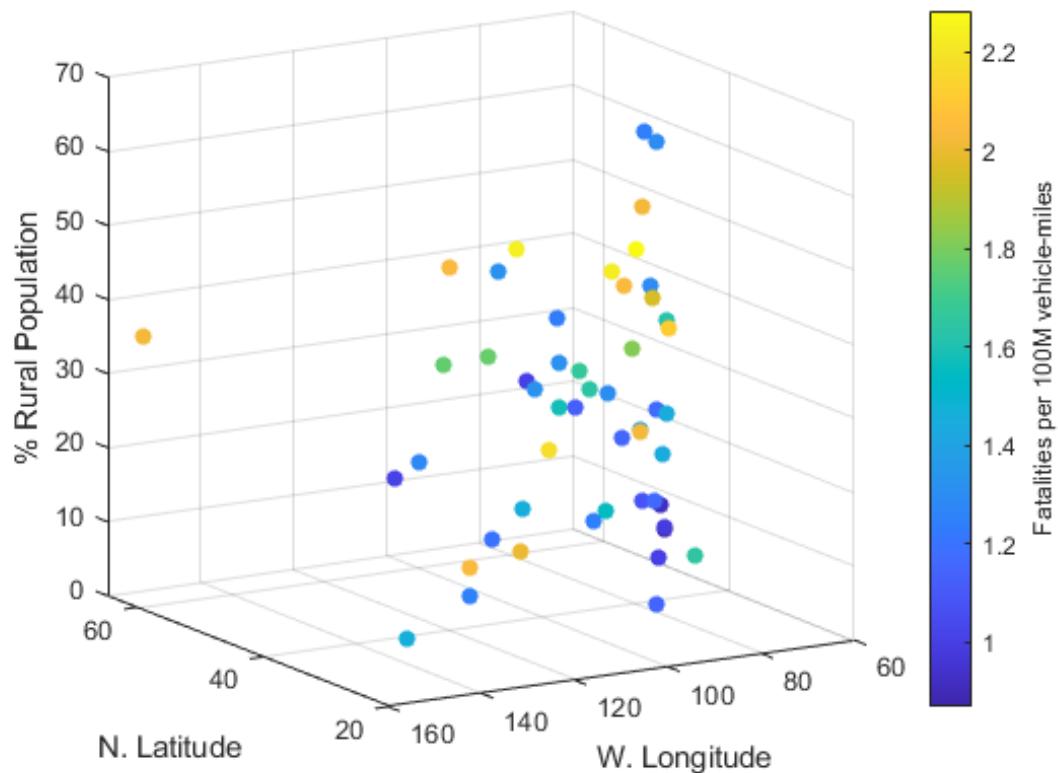
Aangezien deze twee verschillen geen invloed hebben op de voorwaarden, gelden dezelfde uitkomsten als bij 1-dimensionale visualisaties. De conclusie blijft dus ook hetzelfde: dit type is niet bruikbaar om een chatbotflow voor te stellen.

5.6 Multidimensionale Datavisualisaties

Multidimensionale data omvatten voornamelijk relationele en statistische databanken en worden voornamelijk gebruikt voor het manipuleren van multidimensionale data, aldus Shneiderman (1996). Deze datavisualisaties zijn bedoeld voor statistisch onderzoek zoals het zoeken van verbanden tussen variabelen van de items of om patronen te ontdekken. Het voorstellen van multidimensionale visualisaties kan gebeuren via 2D of 3D voorstellingen en, afhankelijk van het aantal dimensies, door zaken toe te voegen zoals scrollbars of een extra as of door kleuren te introduceren. Een voorbeeld waar er gekozen wordt om een extra dimensie weer te geven, door gebruik van kleur kan teruggevonden worden in Figuur 5.3

Hoewel er 2D en 3D voorstellingen gebruikt worden mag dit niet verward worden met 2-dimensionale en 3-dimensionale datavisualisaties, die besproken worden in Sectie 5.3 en 5.4. Deze gaan immers over de vorm van de data, niet over de vorm van de visualisatie. Het is dus mogelijk om een multidimensionale visualisatie voor te stellen zoals in Figuur 5.1 en 5.2, maar de regels (zijnde de essentie van de posities van de items) zijn niet per se geldig.

Dit laatste betekent niet dat de uitkomsten voor de eerste twee voorwaarden dezelfde zijn als die voor Sectie 5.3 en 5.4. Hoewel de regels van ligging ten opzichte van elkaar, zoals bij de planaire en volumetrische hier niet van toepassing zijn, slaagt ook dit type niet voor



de laatste voorwaarde. De multidimensionale visualisaties kunnen in principe gebruikt worden zonder dat er verkeerde interpretaties kunnen gemaakt worden, maar het doel is om statistische operaties uit te voeren zoals het zoeken van verbanden tussen items uit verschillende categorieën. Dit is niet waar dit onderzoek over gaat.

Hoewel er veel geldige visualisaties zouden kunnen gemaakt worden vallen ook de multidimensionale datavisualisaties in de groep van onbruikbare types.

5.7 Boom Datavisualisaties

Boom- en hierachische structuren bestaan uit individuele items waarbij de kinderen gelinkt worden aan hun ouders. Deze links kunnen informatie bevatten over de relatie ouder-kind. Het belang van dit type ligt in de structuur van de visualisaties.

Het valt direct op dat niet enkel direct aan de voorwaarden wordt voldaan maar ook dat de focus ligt op het vervullen van de voorwaarden. De items in de boom zijn uiteraard de nodes en de linken tussen deze items zijn de relatie ouder-kind die in de voorwaarde wordt besproken. Ook aan de laatste voorwaarde wordt voldaan.

Doordat aan alle drie de voorwaarden wordt voldaan kan er besloten worden dat boomdatavisualisaties bruikbaar zijn voor het voorstellen van de flow.

5.8 Netwerk Datavisualisaties

Netwerk datavisualisaties kunnen gezien worden als een uitbreiding op bomen waarbij er andere regels aan toegekend kunnen worden. Netwerken worden ook vaak grafen genoemd. Beide termen worden gebruikt in dit onderzoek. In netwerken kunnen er bijvoorbeeld lussen voorkomen (netwerk met cykels) of is de richting van de link/relatie niet van belang (ongericht netwerk). Hoe dit op de flows toegepast kan worden is hier nog niet van belang. Netwerken zijn, zoals bomen, gefocused op de linken tussen nodes om bijvoorbeeld het aantal tussenstappen of paden te berekenen tussen 2 of meerdere nodes.

Zoals bij bomen worden nodes als aparte entiteiten beschouwd. Hoewel de richting van de nodes kan weggelaten worden en de link ouder-kind daardoor kan wegvalLEN, wil dat niet zeggen dat het altijd zo zal zijn. Om te slagen voor de eerste voorwaarde moet het netwerk gericht zijn. De eerste twee voorwaarden zijn dus al behaaldt. Ook aan de laatste voorwaarde wordt voldaan.

Netwerk visualisaties zijn ook bruikbaar voor het visualiseren van chatbotflows, op voorwaarde dat het netwerk gericht is.

5.9 Conclusie Datavisualisaties types

De lineaire, planaire, volumetrische, tijdsgebonden en multidimensionale visualisatietypes zijn niet bruikbaar om een chatbotflow voor te stellen. Deze types zullen dus ook niet verder onderzocht worden. Enkel de boom- en netwerk gestructureerde visualisaties worden verder onderzocht in de volgende hoofdstukken.

6. Datavisualisaties

‘From Data to Viz’ is a classification of chart types based on input data format.

— from Data to Viz, 2020a

Zoals eerder vermeld, zijn er heel wat classificaties voor visualisatie (-types). Dit onderzoek gebruikt de classificatie van “from Data to Viz” voor het bestuderen van visualisaties die behoren bij de types besproken in Hoofdstuk 5. In dit hoofdstuk bleek dat enkel boom- en netwerk gestructureerde visualisaties gebruikt kunnen worden. De classificatie, die in dit hoofdstuk wordt gebruikt neemt deze samen onder de term “Netwerk”, waarbij er nog verdere opsplitsingen gemaakt worden.

6.1 Netwerk Visualisaties

De visualisaties binnen deze categorie worden opgedeeld volgens een beslissingsboom. Deze kan teruggevonden worden in Figuur 6.1. In volgende secties worden deze visualisaties individueel besproken en gevalideerd volgens, de voorwaarden in Sectie 3.3. De voorwaarden van de types moet immers ook van toepassing zijn op de visualisaties onder deze types.

Volgende visualisaties worden teruggevonden worden onder netwerk visualisaties:

- Network Diagram
- Chord Diagram
- Arc Diagram

- Sankey Diagram
- Heatmap
- Hive
- Dendrogram
- Treemap
- Circular Packing
- Sunburst
- Hierarchical Edge Bundling

6.2 Network Diagram

Netwerk diagrammen of grafen worden onderverdeeld in vier types:

1. ongewogen en ongerichte graaf
2. gewogen en ongerichte graaf
3. ongewogen en gerichte graaf
4. gewogen en gerichte graaf

(from Data to Viz, 2020h)

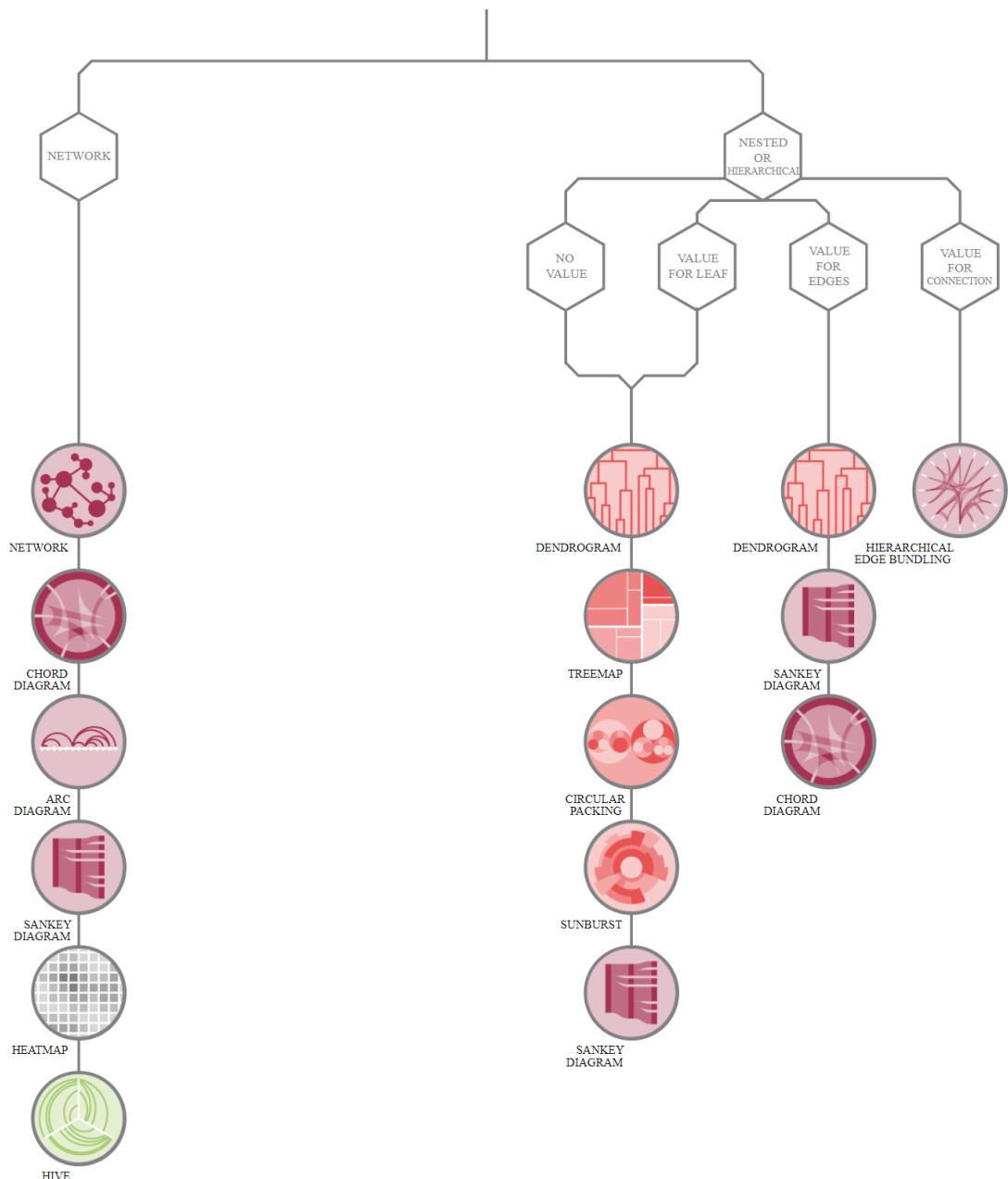
Zoals besproken in Hoofstuk 5, zijn ongerichte netwerken ongeldig omdat voor de relatie niet langer duidelijk is welke node de parent is en welke het kind. De eerste twee opties vallen dus al direct weg. De link tussen de nodes draagt ook geen gewicht en bijgevolg kan de laatste optie ook geschrapt worden. Het enige geldige type is de *ongewogen en gerichte graaf*.

Figuur 6.2 is een goed voorbeeld van hoe een netwerk gebruikt kan worden om een chatbotflow te visualiseren. De *Must-haves* vanuit Hoofdstuk 4 zijn aanwezig en er wordt voldaan aan de voorwaarden uit Hoofdstuk 3. De titel van de chatbot is voorzien en de ouder-kind relaties zijn duidelijk door de pijlen tussen de nodes. Wanneer er gehoverd wordt over een node verschijnt de trigger conditie: de intent met naam “A1” wordt herkend.

6.3 Chord Diagram

Een *chord diagram* toont nodes en de connecties/relaties hiertussen in een cirkelvormige figuur. De bogen tusen de nodes (in de visualisatie voorgesteld als een deel van de cirkel) stellen de relaties voor. De breedtes van deze bogen zeggen iets meer over het gewicht van de relatie tussen twee nodes ten opzichte van de andere gelinkte nodes. (from Data to Viz, 2020c)

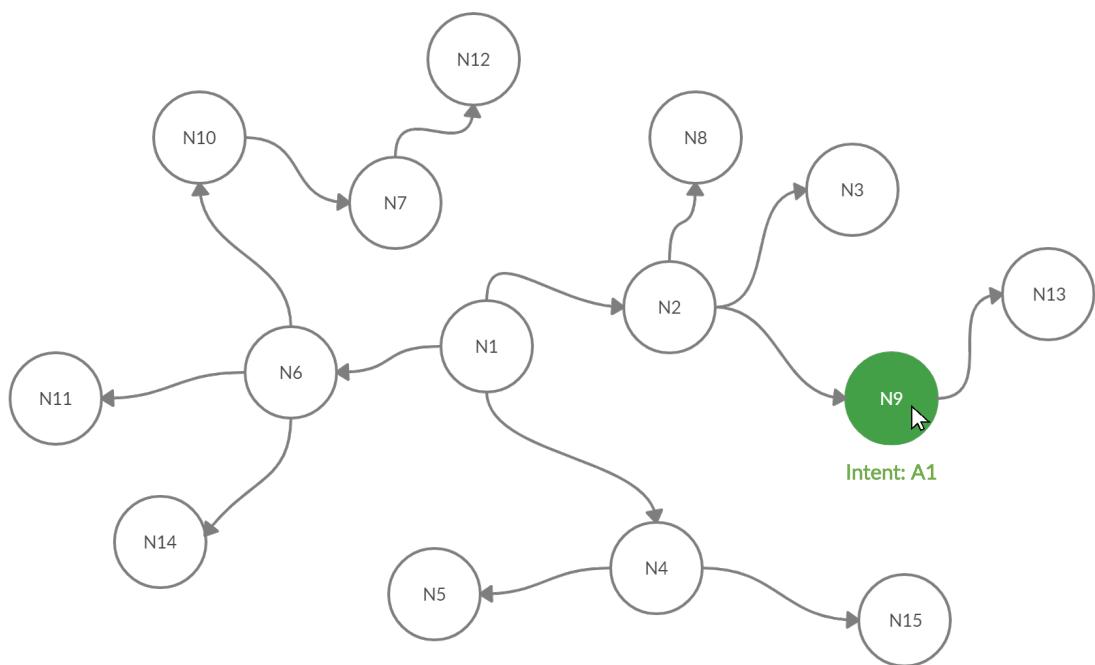
Dit laatste zegt al meteen waarom het een slecht idee is om deze visualisatie te gebruiken. De enige relaties in een chatbotflow zijn de ouder-kind relaties en, zoals in Sectie 6.2 werd besproken, hebben de relaties geen gewicht.



Figuur 6.1: Overzicht van “Netwerk”-visualisaties

Source: <https://www.data-to-viz.com/#explore>

Chatbot: Sherlock



Figuur 6.2: Voorbeeld chatbotflow met een netwerk diagram

Er wordt dus niet voldaan aan de derde voorwaarde uit Sectie 3.3 en dus kan niet gebruikt worden om de chatbotflow te visualiseren.

In Figuur 6.3 wordt er toch een poging gedaan om een flow voor te stellen. In Figuur 6.3a wordt een flow voorgesteld als een boom, terwijl in Figuur 6.3b dit gebeurt aan de hand van een chord diagram. Het wordt al snel duidelijk dat hoe langer en groter de flow wordt, hoe minder overzichtelijk deze tweede visualisatie zal worden. Daarnaast verliest dit diagram zijn nut doordat alle connecties dezelfde breedte hebben. Dit bevestigt nogmaals dat er niet voldaan wordt aan de reeds besproken voorwaarde.

6.4 Arc Diagram

Bij een *arc diagram* worden alle nodes op één as gezet en word door middel van bogen de relatie tussen nodes weergegeven. Elke node, behalve root nodes, heeft dus minimaal 1 boog die toekomt en de parent-child relatie voorstelt. Waar een netwerkdiagram wint op vlak van structuur, wint het boogdiagram in het duidelijker weergeven van labels (naam en trigger conditie(s) node). Ook kunnen er gemakkelijk verbanden gevonden worden tussen meerdere dialogs. De “Jump To” link kan hier dus ook makkelijk getoond worden. (from Data to Viz, 2020b)

Er wordt zowel voldaan aan de voorwaarden voor inhoud als aan de tweede en derde voorwaarde voor de visualisatie zelf. De eerste voorwaarde kan falen door onduidelijkheid in linken tussen nodes. Deze kunnen onstaan als:

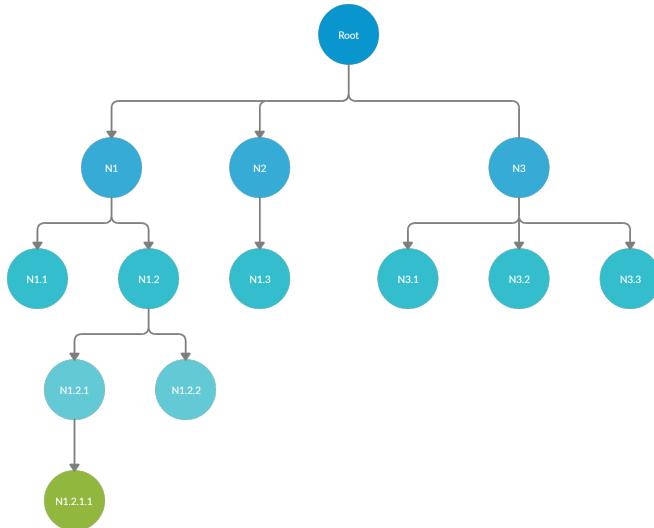
- De nodes op de horizontale as geen volgorde volgen
- Het verschil tussen parent-child en jump-to relaties onduidelijk is

Neem een voorbeeld waar N1 de parent is van N2 en N4, en N2 de parent is van N3. Figuur 6.4 illustreert het belang van de volgorde bij gebruik van een arc diagram. In Figuur 6.4a kan er onmogelijk gezegd worden welke nodes parents zijn terwijl dit in Figuur 6.4b wel kan.

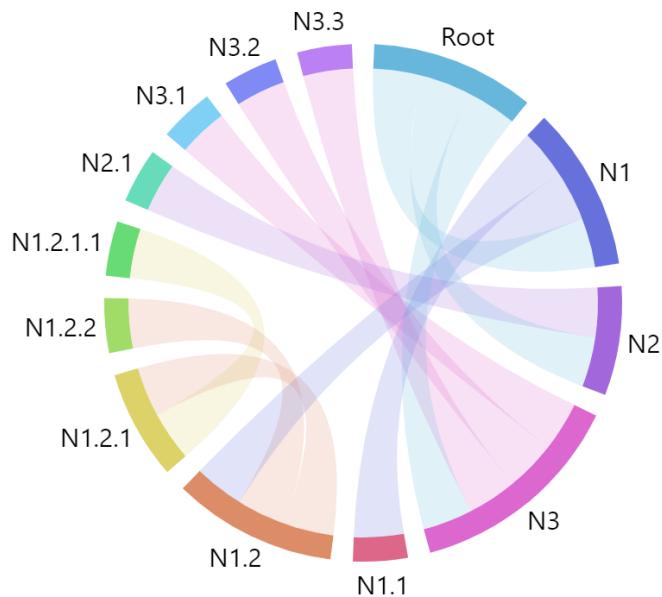
Hetzelfde voorbeeld wordt gebruikt maar N4 heeft nu een “Jump-To”-referentie naar een nieuwe node N0, en N3 heeft ook een “Jump-To”-referentie naar N1. In Figuur 6.5a is het onduidelijk welke boog welke relatie voorstelt. Uit de volgorde zou kunnen geconcludeerd worden dat N0 de parent is van N4. Hetzelfde geldt voor N1 en N3. In Figuur 6.5b worden de “Jump-To”-referenties voorgesteld door een andere kleur boog, een ander type boog of een onderlangse boog.

OPMERKING: bij deze illustraties wordt er geen rekening gehouden met het visualiseren van de triggercondities van de nodes.

Wanneer deze twee gevallen worden gerespecteerd wordt er voldaan aan de eerste voorwaarde en kan bijgevolg besloten worden dat deze visualisatie gebruikt kan worden voor een chatbotflow.



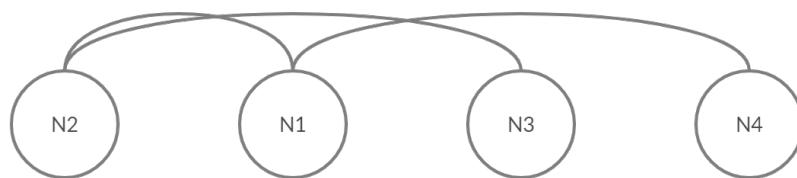
(a) Flow als een boom



(b) Flow met een chord diagram

Figuur 6.3: Voorbeeld chatbotflow met chord diagram

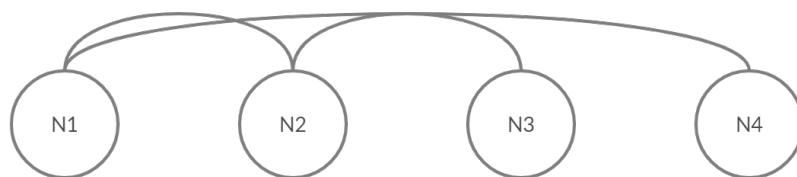
Chatbot: Sherlock



Geen volgorde

(a) Horizontale as heeft geen volgorde

Chatbot: Sherlock

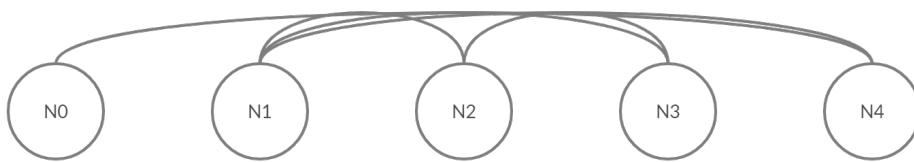


Parent staat steeds links

(b) Horizontale as heeft wel een volgorde

Figuur 6.4: Probleem met volgorde bij een arc diagram

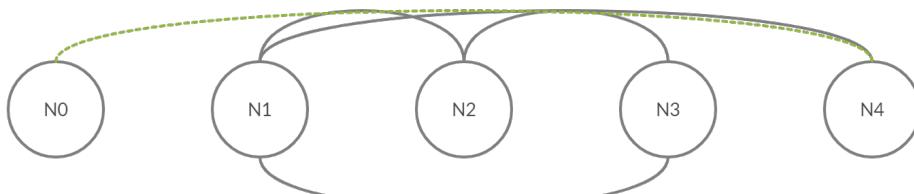
Chatbot: Sherlock



Parent staat steeds links

(a) Verschil parent-child en “Jump-To” onduidelijk

Chatbot: Sherlock



Parent staat steeds links

(b) Verschil parent-child en “Jump-To” duidelijk

Figuur 6.5: Probleem met “Jump-To”-referenties bij een arc diagram

Er is ook een variant waarbij er gewichten worden gebruikt voor de linken tussen nodes. (Ferdio, 2020a; from Data to Viz, 2020b)

Zoals eerder besproken worden er geen gewichten toegewezen aan relaties bij een chatbot-flow en wordt deze variant als ongeldig beschouwd.

6.5 Sankey Diagram

Sankey diagrammen worden gebruikt om flows voor te stellen, meer bepaald de proportionele verhoudingen met de totale flow. (from Data to Viz, 2020i)

Hoewel het mogelijk is om een flow voor te stellen met een Sankey diagram, zal de flow leiden tot foutieve interpretaties. De chatbotflow heeft immers geen gewichten tussen de relaties waardoor dit type “misbruikt” wordt (derde voorwaarde, zie Sectie 3.3).

Bijgevolg kan een Sankey diagram niet gebruikt worden om een chatbotflow te visualiseren.

6.6 Heatmap

Heatmap is really useful to display a general view of numerical data, not
to extract specific data point.
— from Data to Viz, 2020g

Op basis van de structuur van een *heatmap* zou je intuïtief al kunnen concluderen dat deze niet geschikt is voor het visualiseren van een chatbotflow. Een chatbotflow is immers niet gestructureerd als een matrix. Er wordt ook niet aan de voorwaarden voldaan van Sectie 3.3. De nodes zijn geen numerieke waardes, en hebben ook geen gewicht, waardoor onmiddellijk de derde voorwaarde wordt geschonden. Ook is het niet mogelijk om de ouder-kind relaties te visualiseren.

Heatmaps worden dus niet gebruikt voor visualisaties van chatbotflows.

6.7 Hive

Hives worden gebruikt om van een “onoverzichtelijk” netwerk naar een overzichtelijk lineair model over te gaan. De nodes van een netwerk worden op een as geplaatst op basis van een eigenschap. Elke as heeft een type of groep waartoe nodes kunnen behoren. De nodes worden standaard gesorteerd volgens aantal linken in een netwerk. De linken tussen nodes worden voorgesteld als bogen die tussen twee assen lopen. (Bostock, 2012; Krzywinski, 2011)

Bij de chatbotflow zijn er echter geen kenmerken waarop de nodes van de flow (zinvol) gesorteerd kunnen worden. Dit zou betekenen dat alle nodes op één rechte zouden liggen.

Als er wordt gekozen om de afspraak van linken te respecteren faalt de parent-child relatie voorwaarde. Wordt er gekozen om dit niet te doen, dan faalt de derde voorwaarde: de visualisatie wordt immers misbruikt.

Een hive kan dus niet gebruikt worden om een chatbotflow voor te stellen.

6.8 Dendrogram

Een *dendrogram* is een boomdiagram dat gebruikt wordt om hiërarchische data voor te stellen. De boom start met een “root”-node, waaruit nieuwe nodes (kinderen) vertrekken die op hun beurt de ouder zijn voor nieuwe nodes Er zijn twee types van dendrogrammen: de ene wordt gebouwd uit hiërarchische data, de andere ontstaat door het clusteren van data en zo onderlinge verbanden te zoeken. (Ferdio, 2020b; from Data to Viz, 2020e)

De eerste voldoet aan alle voorwaarden en kan gebruikt worden om een flow te visualiseren. Figuur 6.6 illustreert hoe een mogelijke flow er uit zou kunnen zien. Bij de andere zijn de bovenliggende nodes gecreëerd door het clusteren van onderliggende nodes, waarbij een bepaalde gemiddelde waarde werd bekomen. Deze waarde is dan de waarde voor de “parent”-node. Zo zouden steden als nodes kunnen geclusterd worden volgens afstand in vogelvlucht.

Het tweede type voldoet dus niet omdat ofwel de eerste voorwaarde voor visualisaties wordt geschonden (door parent-child relaties te laten bepalen door clusters), ofwel de derde voorwaarde (wanneer de data zo gemanipuleerd worden dat de ouder-kind relaties overeenkomen met de clusters waardoor de visualisatie zijn waarde verliest).

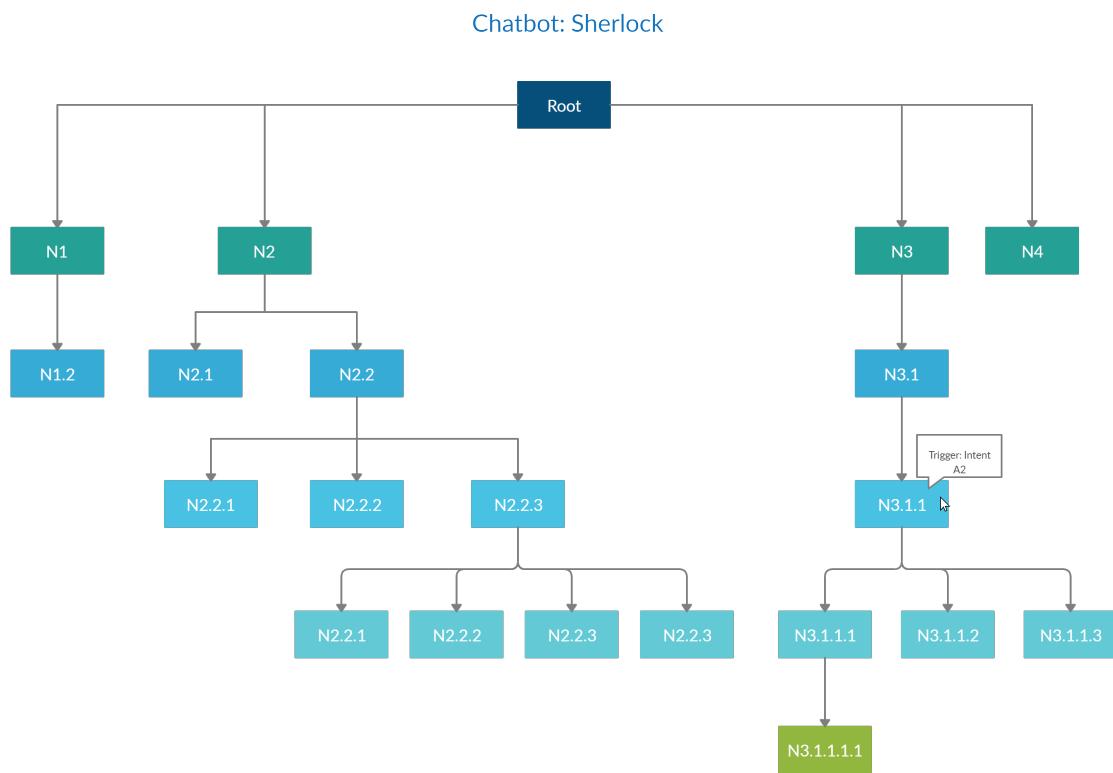
6.9 Treemap

Een *treemap* visualiseert hierachische data in een rechthoekige figuur, waarin elke node een rechthoek binnen de figuur voorstelt, en de kinderen een rechthoek binnen de rechthoek van hun ouder zijn. (from Data to Viz, 2020k)

Hoewel dit op eerste zicht een geldige visualisatie voor flows zou zijn, kan het zijn dat volgende geldt: de grootte van de rechthoek is proportioneel aan de waarde van het item. De nodes in een chatbotflow hebben echter geen numerieke waarde en bijgevolg faalt deze visualisatie voor de laatste voorwaarde.

Moest er van uit gegaan worden dat alle sibling nodes eenzelfde waarde hebben dan faalt de voorwaarde nog steeds. Neem een voorbeeld waarbij node N1 drie kinderen heeft, namelijk N1.1, N1.2 en N1.3 en een sibling N2 met als child nodes N2.1 en N2.2. Als N1.1, N1.2, N1.3, N2.1 en N2.2 allemaal eenzelfde grootte hebben (grootte = x), dan is de grootte van N1 = 3X + C¹ terwijl de grootte van N2 = 2X + C. Dit zou betekenen dat N1 een grotere waarde heeft dan zijn sibling N2, wat het bovenstaande tegenspreekt.

¹ neem constante C voor witruimte tussen de zijden van een node en de zijden van diens kinderen



Figuur 6.6: Voorbeeld chatbotflow met een dendrogram

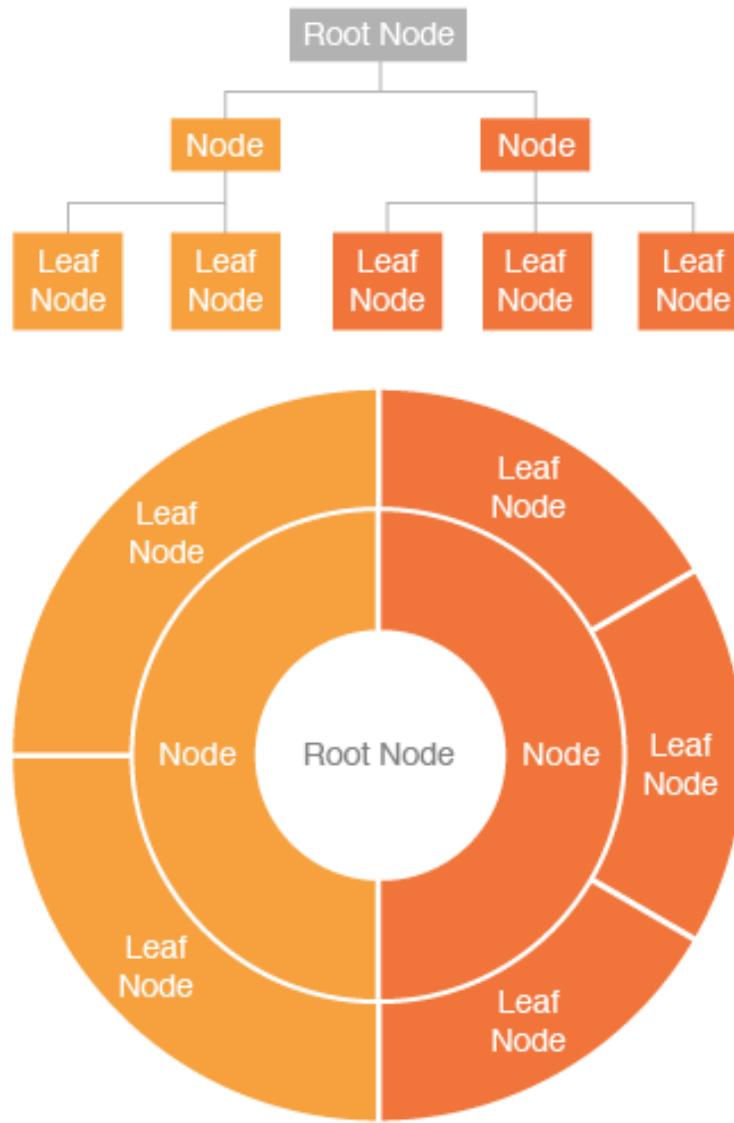
Het is mogelijk om de eigenschap van proportionele waarde achterwege te laten. In dit geval wordt er ook voldaan aan de derde voorwaarde en kan een treemap als nog gebruikt worden voor het visualiseren van een chatbotflow.

6.10 Circular Packing

Circular packing is een alternatief voor een treemap, waarbij er cirkels gebruikt worden in plaats van rechthoeken. Bijgevolg slaagt ook deze visualisatie voor alle voorwaarden, ten minste als ook hier geen rekening wordt gehouden met waarden van nodes. (from Data to Viz, 2020d; The Data Visualisation Catalogue, 2020b)

6.11 Sunburst

Een *sunburst* diagram wordt gebruikt voor het visualiseren van hiërarchische data in de vorm van ringen, waarbij de root de binnenste ring is en elk opvolgend child zich rond zijn parent vormt. De sunburst kan gebruikt worden om de data als deel van een geheel weer te geven of als een flow voor te stellen. Uiteraard is de tweede optie het meest van toepassing. Er kan ook gekozen worden om al dan niet de bogen proportioneel voor te stellen. (from Data to Viz, 2020j; The Data Visualisation Catalogue, 2020d)



Figuur 6.7: Voorbeeld chatbotflow met een sunburst

Source: https://datavizcatalogue.com/methods/sunburst_diagram.html

Zoals bij treemaps en circular packing werd besproken, zullen proportionele voorstellingen leiden tot een ongeldige visualisatie en wordt er gekozen om dit niet te doen. Er wordt voldaan aan de visualisatie voorwaarden en de gewenste data kunnen getoond worden. Bijgevolg is een sunburst een geldig type. In Figuur 6.7 wordt een visuele voorstelling van een flow getoond.

6.12 Hierarchical Edge Bundling

Hierarchical Edge Bundling allows to visualize adjacency relations between entities organized in a hierarchy.

— from Data to Viz, 2020g

Hoewel bovenstaande definitie de indruk geeft dat *hierarchical edge bundling* een goede visualisatie zou zijn, is dit niet het geval. Er wordt steeds gestart van een dendrogram waarbij de structuur in een cirkelvorm gelegd. Daarna worden de “aangrenzende relaties” gelegd door bogen tussen de bladeren² te trekken. Als laatste wordt de hiërarchische structuur weggelaten waardoor enkel de nodes op de cirkel overblijven.

Hierdoor verdwijnen de ouder-kind relaties en worden enkel de uiterste nodes getoond, met als gevolg dat de eerste twee voorwaarden voor visualisaties falen. Hierarchical edge bundling kan dus niet gebruikt worden om een chatbotflow voor te stellen. Figuur 6.8a toont de eerste stap waar hiërarchische data gevisualiseerd worden met een dendrogram en Figuur 6.8b toont het eindresultaat, na alle linken te trekken en de hiërarchische structuur te verwijderen.

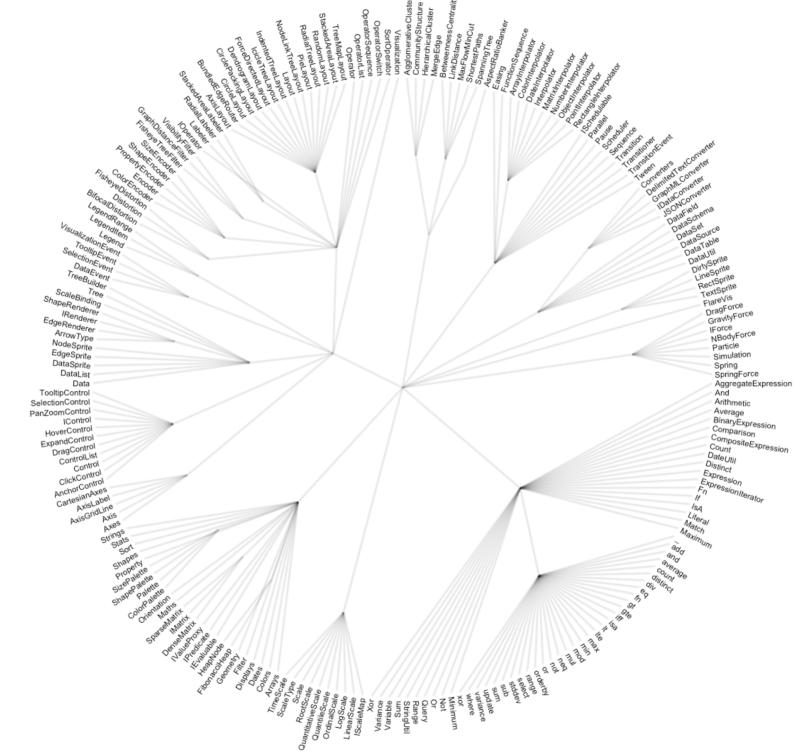
6.13 Conclusie

Voor het visualiseren van chatbotflows kunnen volgende visualisaties gebruikt worden:

- Network Diagram
- Arc Diagram
- Dendrogram
- Treemap
- Circular Packing
- Sunburst

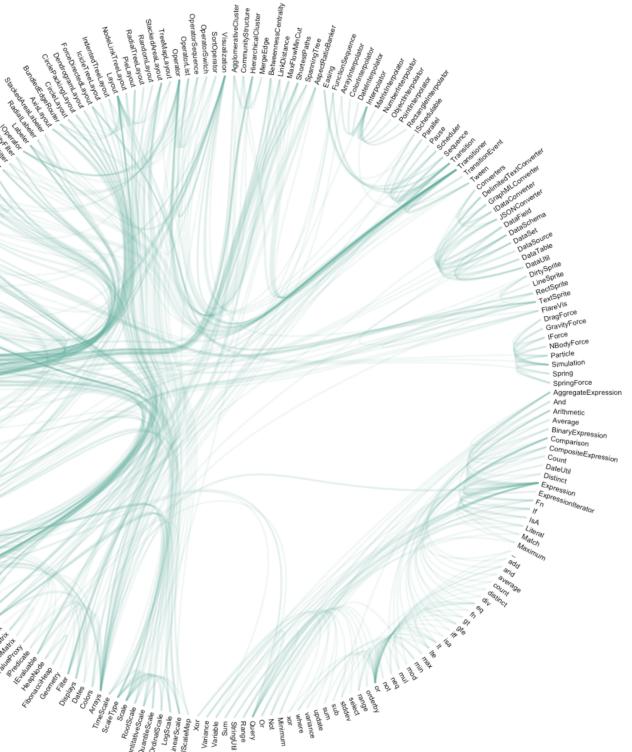
Hoewel dit buiten de scope van dit onderzoek valt, zouden de visualisaties zoals de Sankey diagram en de heatmap gebruikt kunnen worden bij analytisch onderzoek tijdens de ontwikkeling en het onderhouden van een chatbot. In dit geval kunnen de relaties ook gewichten dragen. Een voorbeeld hiervan zou het procentueel aantal gebruikers zijn dat langs een node passeert ten opzichte van alle gebruikers die langs hun ouder passeren. DialogFlow voorziet hiervoor al een visualisatie met een Sankey diagram, zoals

²bladeren of leaves zijn de uiterste nodes, ook wel de nodes zonder kinderen



(a) Start met een dendrogram

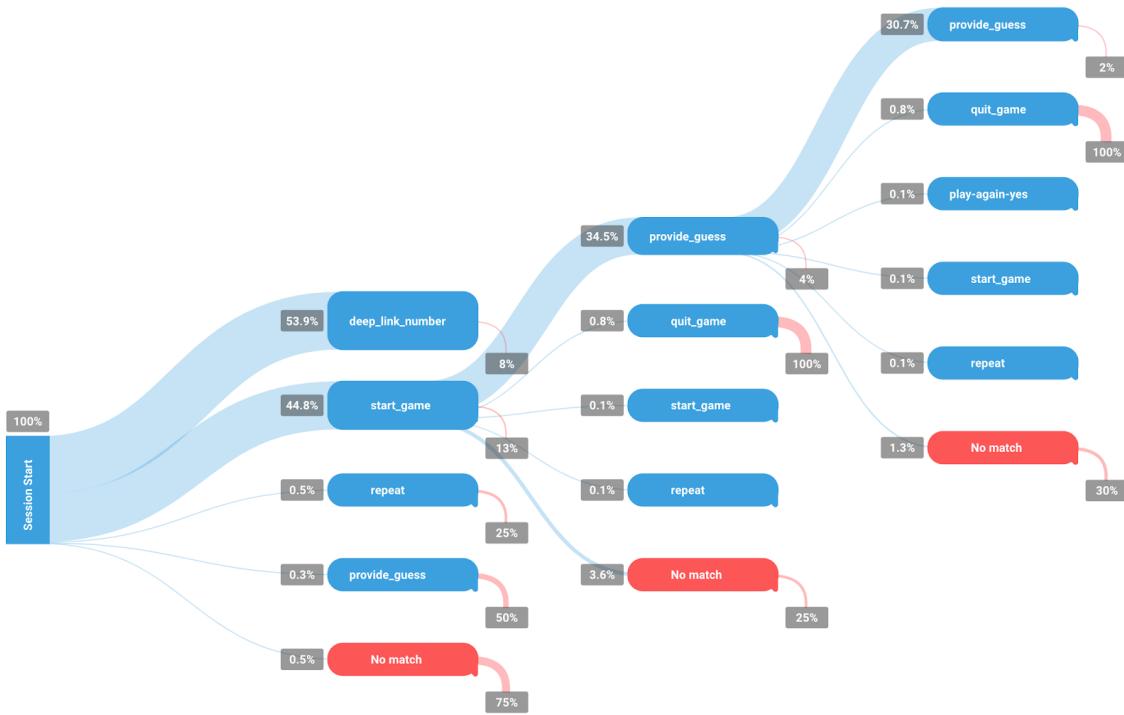
Source: https://www.data-to-viz.com/graph/edge_bundling.html



(b) Eindresultaat van hierarchical edge bundling

Source: https://www.data-to-viz.com/graph/edge_bundling.html

Figuur 6.8: Voorbeeld van hierarchical edge bundling



Figuur 6.9: Voorbeeld procentuele spreiding van een DialogFlow chatbotflow

Source: <https://medium.com/google-developers/analytics-for-actions-d8212a6bf6a5>

teruggevonden kan worden in Figuur 6.9. Dit kan een klant helpen kiezen welke takken van de chatbot verder moeten worden uitgebreid en welke er kunnen weggelaten worden.

7. Interactie bij Visualisaties

Dit hoofdstuk onderzoekt de overgebleven visualisaties uit het vorige hoofdstuk en kijkt welke problemen er kunnen optreden in elke visualisatie. Er wordt gekeken of deze problemen kunnen opgelost worden door gebruik te maken van operaties die door een gebruiker kunnen worden uitgevoerd. Deze interacties zijn reeds besproken in Hoofdstuk 2, namelijk:

- Details-on-demand
- Extract
- Filter
- History
- Overview
- Relate
- Zoom

7.1 Algemeen

De operaties **History**, **Overview** en **Details-on-demand** worden niet per individuele waarde besproken.

De geschiedenis (history) is nuttig om een overzicht van uitgevoerde operaties bij te houden en operaties ongedaan te maken. Dit zullen vaak extract- en filter-operaties zijn. Wanneer een extract- of filter-operatie ongedaan gemaakt kan worden, door een nieuwe extract/filter operatie uit te voeren, zal de history-operatie zijn nut verliezen en weinig extra impact op de visualisatie hebben. In dit geval is het niet nodig om een history-operatie te voorzien.

Het overzicht (overview) is een noodzakelijke vereiste die in vorige hoofdstukken reeds is afgetoetst en is dus van toepassing op elke visualisatie die in dit hoofdstuk wordt behandeld.

De *details-on-demand*-operatie is geen ingrijpende operatie (zie Hoofdstuk 3) maar een ondersteunende operatie. De visualisatie wordt overzichtelijker bij gebruik van deze operatie maar er zal niets veranderen aan de structuur/layout van de visualisatie. Deze operatie houdt zich bezig met het tonen en verbergen van de *Should-haves* van Hoofdstuk 3 en 4. Deze operatie zal dus bij elke operatie gebruikt worden om een beter overzicht te geven.

7.2 Network

Zoals reeds in Hoofdstukken 5 en 6 vermeld, zijn enkel gerichte *netwerken* bruikbaar. Netwerken geven duidelijk de ouder-kind relaties weer door pijlen te trekken tussen nodes. Een netwerk groeit naarmate er meer nodes bijkomen. In -en uitzoomen zijn nodig om te grote netwerken nog steeds te kunnen bestuderen.

De **zoom**-operatie zorgt ervoor dat nodes en hun informatie terug zichtbaar worden wanneer een netwerk te groot wordt. Het is echter niet voldoende om enkel te kunnen zoomen want hoe meer je inzoomt, hoe langer het duurt om door de volledige visualisatie te gaan.

Netwerken kampen echter met een nog groter probleem: hoe meer data er in de visualisatie zitten, hoe meer kans dat er zich zogenaamde “haarballen” vormen. Haarballen zijn netwerken die onleesbaar zijn doordat alle nodes opgehoopt zijn er geen overzichtelijke structuur meer is. (from Data to Viz, 2020h; The Data Visualisation Catalogue, 2020c)

Volgende operaties zijn noodzakelijk bij het verhelpen van haarballen:

- Extract
- Filter

Filteren maakt het mogelijk om:

- individuele nodes (zonder kinderen) onzichtbaar te maken (bladeren knippen).
- nodes en al hun kinderen onzichtbaar te maken (takken knippen).
- volledige netwerken (dialogs) onzichtbaar te maken (bomen knippen).

Bij het **extracten** wordt er niet in de originele visualisatie geknipt maar wordt er een nieuwe visualisatie gemaakt met enkel de gewenste data:

- een of meerdere non-root nodes en hun kinderen (takken extracten).
- een of meerdere root nodes en hun kinderen (bomen extracten).

Zoals in Hoofdstuk 4 besproken, kunnen nodes relaties hebben met elkaar die geen ouder-kind relaties zijn. Dit kan wanneer een node voorzien is van een “Jump_To”-veld. Een **relate**-operatie kan hier gebruikt worden om deze relaties weer te geven. Er moet opgelet

worden dat de relaties ouder-kind en jump-to te onderscheiden zijn.

7.3 Arc Diagram

Bij netwerken kunnen veel data leiden tot een onleesbare visualisatie. Bij *arc diagrams* is dit nog een groter probleem. Waar netwerken twee dimensies gebruiken worden bij boogdiagrammen alle nodes op één dimensie verspreid. Bovendien moet de volgorde van de nodes goed doordacht zijn. (from Data to Viz, 2020b; The Data Visualisation Catalogue, 2020a)

Net zoals bij netwerken zullen de operaties **Extract** en **Filter** nodig zijn om onoverzichtelijke diagrammen aan te passen en terug leesbaar te maken. **Zoomen** zal ook hier een belangrijke operatie zijn.

Hoewel arc diagrams geldige visualisaties zijn, kunnen zij best vermeden worden voor grote datasets en meer gebruikt worden voor stukken van dialogs.

Ook hier kan er gebruik gemaakt worden van een **relate**-operatie. Hoofdstuk 6 besprak al enkele risico's die hieraan verbonden zijn en hoe deze opgelost kunnen worden.

7.4 Dendrogram

Dendrogrammen geven een goed beeld van hiërarchische data en zijn bijgevolg ideaal om de hiërarchische structuur van een chatbotflow weer te geven. Een dendrogram groeit in breedte en diepte naarmate een niveau meer kinderen krijgt en bladeren kinderen krijgen (en dus niet langer bladeren zijn). De structuur van een dendrogram blijft overzichtelijk, ongeacht de grootte van de dataset.

Wanneer de visualisatie zeer groot wordt zal **zoomen** wel nodig zijn om informatie van individuele nodes te kunnen zien. Hoewel minder noodzakelijk als bij netwerken en boogdiagrammen, kunnen **extracts** en **filters** handig zijn om minder relevante bomen, takken en bladeren te snoeien.

Door de hiërarchische structuur van een dendrogram moeten er geen extra acties, zoals een andere kleur geven, uitgevoerd worden om ouder-kind en jump-to relaties te onderscheiden. **Relates** kunnen gebruikt worden om bijvoorbeeld alle jump-to relaties op te lichten, maar het zou ook mogelijk zijn om van een individuele node de corresponderende eind-node van de jump-to te laten oplichten of er naartoe te springen.

Een dendrogram kan horizontaal of verticaal gericht worden. Doordat de naam van een node lang kan zijn is horizontale spreiding de beste keuze. (from Data to Viz, 2020e)

Een dendrogram kan in elke situatie gebruikt worden zonder dat deze onduidelijk wordt.

7.5 Treemap en Circular Packing

Een *treemap* en *circular packing* zijn quasi hetzelfde met als enige verschil dat de eerste gebruik maakt van rechthoeken en de tweede van cirkels of ellipsen. De keuze tussen treemap en circular packing is als volgt: de treemap maakt gebruik van alle ruimte, terwijl bij circular packing veel witruimte voorkomt maar de hiërarchie duidelijker getoond wordt. (The Data Visualisation Catalogue, 2020b, 2020e)

Bij vorige visualisaties blijven alle nodes even groot, ongeacht de diepte en de breedte van de flow. Dit is hier niet het geval. Bij netwerken, boogdiagrammen en dendrogrammen wordt de ouder-kind relatie weergegeven door lijnen te trekken tussen de nodes. Bij treemaps en circular packings gebeurt dit door de kinderen van een node in de parent-node te steken.

Het valt direct op dat hoe dieper of breder de flow is, hoe onoverzichtelijker de visualisatie wordt. Het is dus noodzakelijk om te kunnen **zoomen** in de visualisatie. (from Data to Viz, 2020d, 2020k)

Het is ook handig om te **filteren**. Wanneer een node is verwijderd verdwijnen ook alle kinderen van deze node. De siblings van de verwijderde node verdelen de verkregen vrije ruimte onder elkaar. Dit proces gebeurt recursief voor alle onderliggende kinderen.

Een mogelijkheid van **extracten** is als volgt: een extractie vanaf een bepaalde node creëert een nieuwe treemap/circular mapping waarbij de geselecteerde node de root is van de nieuwe visualisatie. Dit maakt navigatie vaak makkelijker dan door in te zoomen.

Het tonen van jump-to relaties is hier niet gemakkelijk. Een mogelijkheid voor een **relate**-operatie zou het oplichten van de destinatie-node zijn, wanneer er over de start-node gehooverd wordt, maar dit zal een slechte oplossing zijn wanneer de jump-to over meerdere niveaus gaat. Ook hiervoor kan er beter gekozen worden voor een netwerk, boogdiagram of dendrogram.

7.6 Sunburst

De laatste visualisatie, het *sunburst diagram*, is het omgekeerde van een treemap/circular packing. Bij een treemap en circular packing wordt er van buiten naar binnen gegaan, bij een sunburst diagram ligt de root in het midden en vormen de kinderen lagen rond de root. Zoals bij treemaps en circular packings zijn de kinderen steeds kleiner of even groot als hun ouder. Ook hier is de **zoom**-operatie vereist voor deze visualisatie.

Extraheren kan op een soortgelijke manier gebeuren: een gewenste node vormt de root van een nieuw sunburst diagram, met de daarbij horende kinderen er rond gelegen.

Ook **filteren** gebeurt op dezelfde manier: de kinderen nemen het verkregen vrije stuk van de cirkel en verdelen hun kinderen recursief over de extra oppervlakte.

Ook hier is het zeer moeilijk om duidelijk de jump-to relaties weer te geven. Zoals bij treemaps en circular packings zou de **relate**-operatie kunnen gebeuren door op te lichten of wat te vergroten. Dit is dus ook niet een ideale visualisatie wanneer men deze relaties wenst weer te geven.

7.7 Conclusie

Bij netwerken en arc diagrammen is het belangrijk om vooraf een goed idee te hebben van de grootte van de dataset. Als de dataset groot is, zijn operaties noodzakelijk om deze visualisaties leesbaar te maken.

Treemaps, circular packings en sunburst diagrammen zijn goed als men een *deel-van-een-geheel* representatie wenst. Als het van belang is om de jump-to relaties weer te geven kan er beter voor een andere visualisatie worden gekozen.

Dendrogrammen kunnen in elke situatie gebruikt worden en hebben geen nadelen bij het weergeven van de data. Deze visualisatie wordt als een ideale oplossing beschouwd voor het weergeven van een chatbotflow.

8. Van Chatbot naar Visualisatie

In vorige hoofdstukken werden alle onderdelen van een chatbotflow besproken en werden er voorwaarden gelegd aan de visualisaties die deze flows visualiseren. In functie van dit onderzoek werd een softwareproject gemaakt waarin er twee visualisaties uitgewerkt werden. Deze visualisaties vertrokken vanaf het exportbestand uit Bijlage B en werden reeds besproken in Hoofdstuk 4.

8.1 Specificaties van de Demo

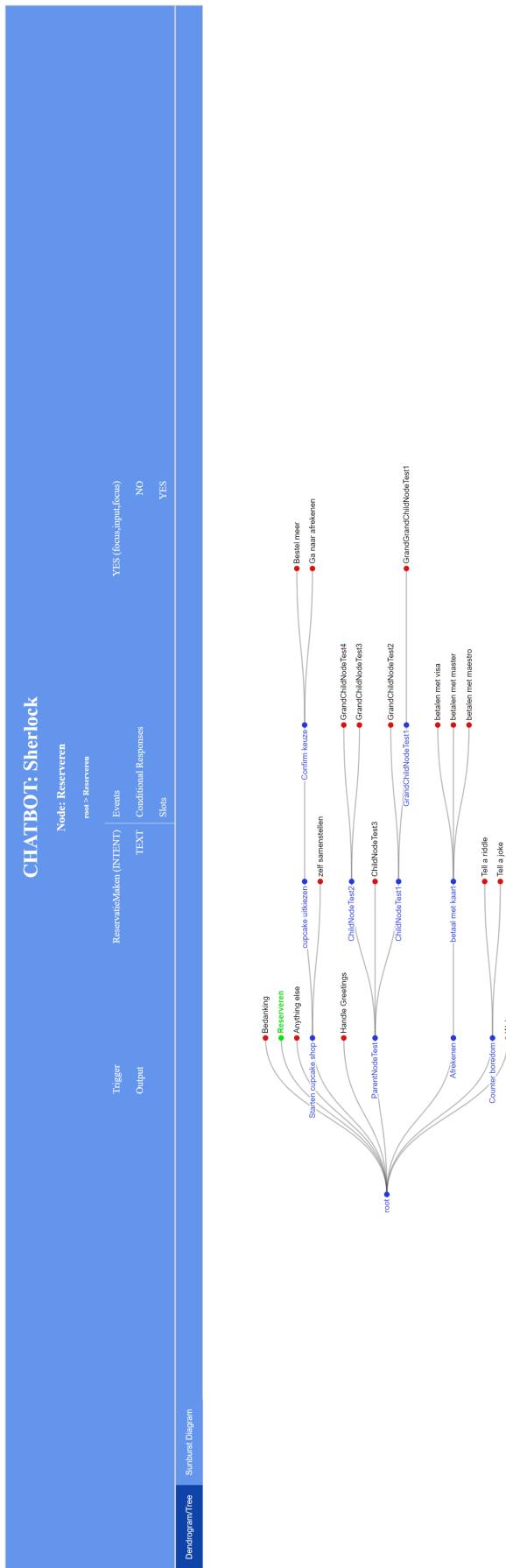
Het project werd gemaakt met HTML en CSS. De bijhorende scripten werden geschreven in JavaScript en D3.js. Voor elke visualisatie werd er een apart script voorzien met D3-instructies. Beide visualisatiescripten maken gebruik van een derde script waarin het JSON-bestand werd getransformeerd naar een JavaScript-Object. Dit script kan teruggevonden worden in bijlage C.

Voor beide visualisaties werd er een SVG gegenereerd. In Hoofdstuk 2 werden twee bestandsformaten voor afbeeldingen besproken: vector- en rasterafbeeldingen. Vectorafbeeldingen hebben twee grote voordelen die goed van pas komen bij het visualiseren van een chatbotflow en het interacteren met deze visualisaties:

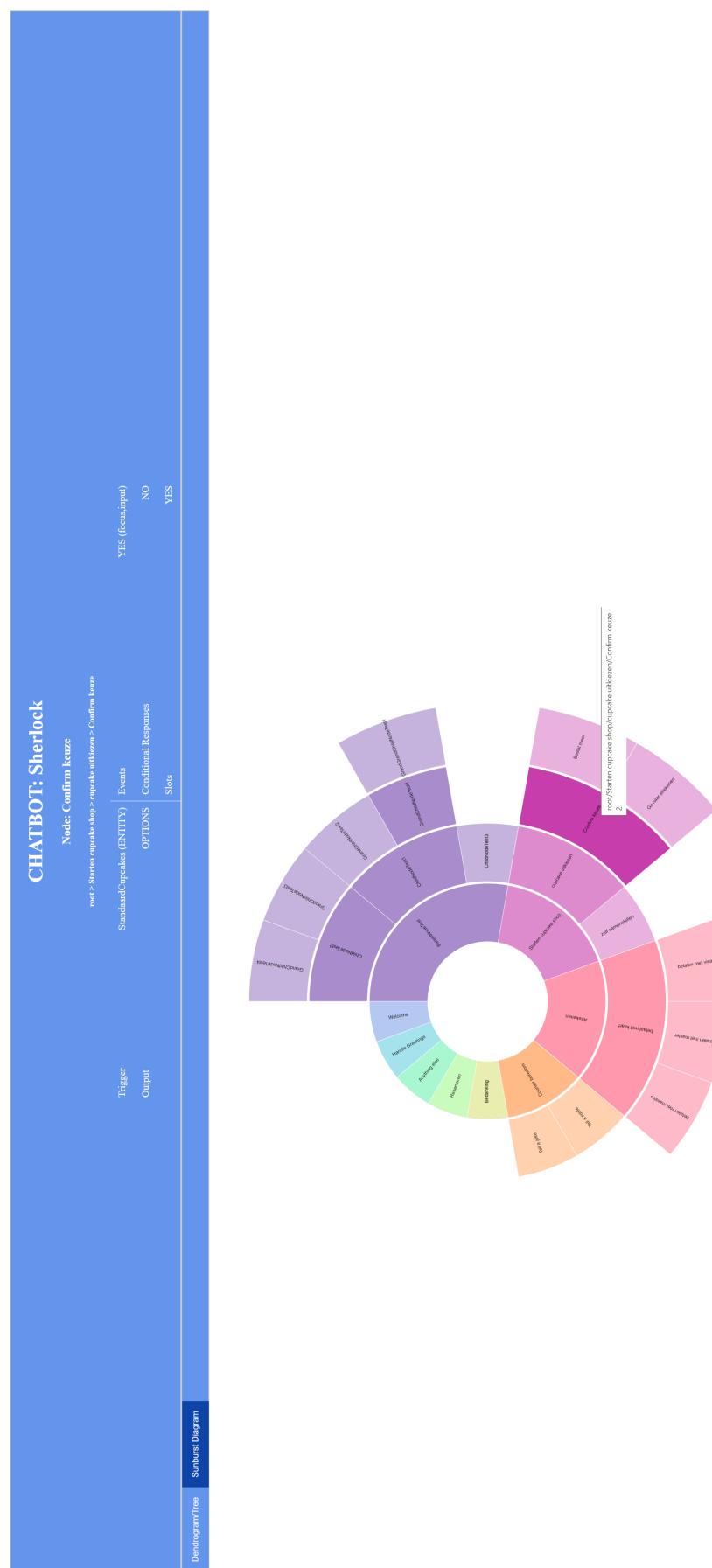
- ze kunnen makkelijk gewijzigd worden
- ze behouden hun kwaliteit bij het in- en uitzoomen

SVG-bestanden vallen onder deze groep en zijn bijgevolg ideaal voor het visualiseren van de chatbotflow.

Bachelorproof IBM Watson Assistant Visualization



Figuur 8.1: Visualisatie van een chatbotflow met een dendrogram



Figuur 8.2: Visualisatie van een chatbotflow met een sunburst-diagram



Figuur 8.3: Het detailpaneel van het demoproject

8.2 Layout van de Demo

De interface van het project bestaat uit twee componenten: een paneel met de details over de chatbot en de geselecteerde node en een paneel met de visualisatie van de chatbotflow. Er kan door middel van de tabs (in de balk tussen beide componenten) gewisseld worden tussen de twee visualisaties. Een voorbeeld van het dendrogram en het sunburst-diagram worden respectievelijk teruggevonden in Figuren 8.1 en 8.2.

Het detailpaneel toont volgende data:

- de naam van de chatbot
- de naam van de geselecteerde node
- het pad van de root tot de geselecteerde node
- de trigger van de geselecteerde node (naam en type)
- de outputformaten van de geselecteerde node (text, image, pauze, option, suggestion)
- de ondersteunende nodes van de geselecteerde node (event, slot en conditional responses)

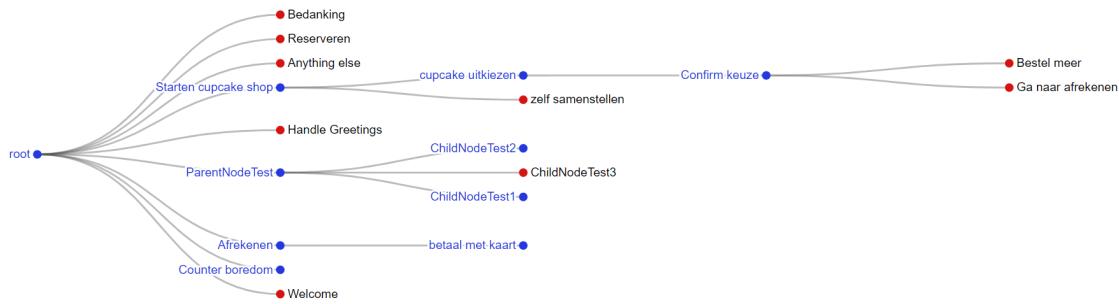
Een voorbeeld van dit paneel kan teruggevonden worden in Figuur 8.3.

8.3 Chatbotflow met een Dendrogram

Het dendrogram ondersteunt de volgende operaties:

- in- en uitzoomen (zoom)
- filteren van ongewenste paden (filter)
- details van een node weergeven bij hoveren (details-on-demand)
- overzicht geven van de volledige chatbotflow (overview)

Nodes met kinderen worden in het blauw gezet en de node-titel staat links van het punt terwijl nodes zonder kinderen roodgekleurd zijn en de titel rechts van het punt staat zoals in Figuur 8.4. Door het klikken op een node met kinderen worden de kinderen in- of uitgeklapt (afhankelijk van de huidige situatie).



Figuur 8.4: Overzicht chatbotflow met een dendrogram

8.4 Chatbotflow met een Sunburst-Diagram

Het sunburst-diagram ondersteunt de volgende operaties:

- in- en uitzoomen (zoom)
- extraheren van een pad (extract)
- details van een node weergeven bij hoveren (details-on-demand)
- overzicht geven van de volledige chatbotflow (overview)

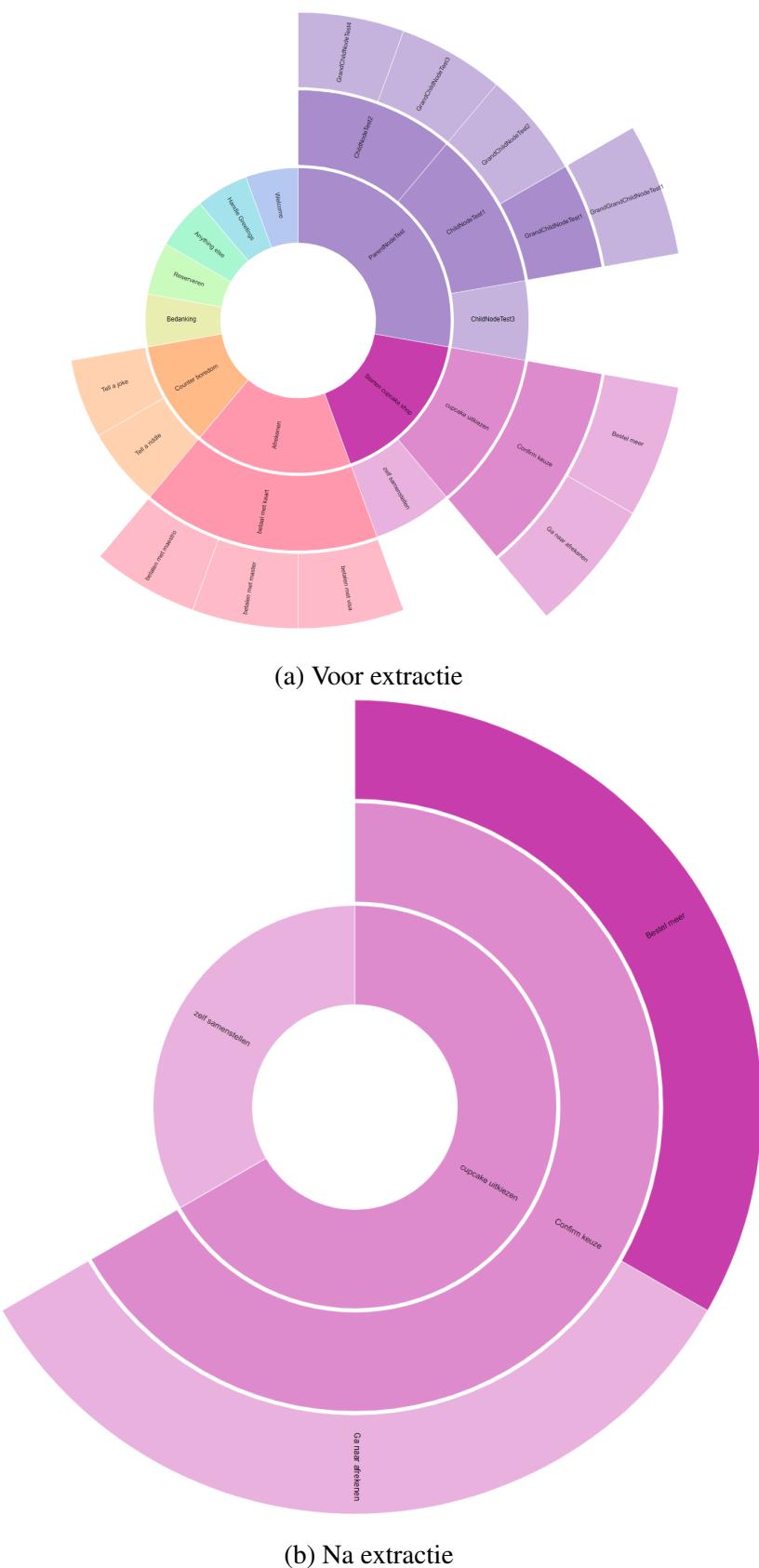
In Figuur 8.5 wordt er getoond hoe de extractie van een pad gebeurt. In Figuur 8.5a wordt er op de node “Starten cupcake shop” geklikt. Hierdoor wordt het sunburst-diagram opnieuw getekend met de node “Starten cupcake shop” als nieuwe root zoals in Figuur 8.5b. Wanneer in het midden van de cirkel geklikt wordt, wordt het diagram opnieuw getekend met de parent van de huidige root als nieuwe root.

8.5 Evaluatie van de Visualisaties

In Hoofdstuk 3 werd er opgelegd dat elke visualisatie bepaalde data moet bevatten, de zogenaamde *must-haves*. In Hoofdstuk 4 werden volgende data gelabeld als *must-haves*:

- naam van de chatbot
- naam van elke node
- trigger van elke node

Bij beide visualisaties kunnen deze data teruggevonden worden. Er werden ook voorwaarden gesteld waaraan de visualisaties zelf moeten voldoen. Aangezien beide visualisaties werden goedgekeurd in Hoofdstuk 6, wordt er automatisch aan deze eisen voldaan. Bijgevolg zijn deze beide visualisaties correcte voorstellingen van een chatbotflow.



Figuur 8.5: Extraheren van het gewenste pad met een sunburst-diagram

9. Conclusie

Dit onderzoek trachtte een antwoord te vinden op de vraag welke visualisaties gebruikt kunnen worden om de flow van een IBM Watson Assistant chatbot te visualiseren. Deze vraag heeft een tweede deel, namelijk: welke bewerkingen of operaties moeten uitgevoerd kunnen worden en met welke redenen? Deze vragen konden pas beantwoord worden door eerst enkele deelvragen op te lossen.

9.1 Hoofdonderzoeksvergadering Visualisaties

De eerste deelonderzoeksvergadering gaat op zoek naar de elementen die een chatbot opmaken. Deze vraag werd grotendeels beantwoord in de literatuurstudie in Hoofdstuk 2 en er wordt verder op ingegaan in Hoofdstuk 4. De chatbot omvat volgende zaken:

- intents
- entities
- metadata
- dialognodes
- counterexamples
- systeemininstellingen
- algemene chatbot informatie

De tweede deelvraag gaat hierop verder maar filtert alle zaken die niet belangrijk of nuttig zijn om op te nemen in een visualisatie. De metadata, counterexamples en systeemininstellingen zijn irrelevant om te visualiseren. Bij de algemene info van een chatbot is de naam van de chatbot uiteraard belangrijk om te tonen. Een chatbot kan immers niet geïdentificeerd worden zonder een naam. De beschrijving en taal kunnen getoond worden

als extra informatie maar dit is niet noodzakelijk. Het toestemmingsveld voor datacollectie kan ook weggelaten worden.

Dialognodes zijn punten waarover wordt gelopen door in *dialoog* te gaan met de gebruiker. Dit zijn de elementen die de *flow* van een chatbot opmaken en zijn dus van essentieel belang om te visualiseren.

Intents en entities geven deze nodes inhoud en betekenis. Ze worden gebruikt om zaken te herkennen in de gebruikersinput en zo een correct antwoord te formuleren. Hoewel niet noodzakelijk voor de visualisatie, zijn ze een zeer belangrijk onderdeel van de chatbot en wordt er aangeraden om ze te vermelden als triggers van nodes en voor het identificeren van data bij slots.

Voor een uitgebreid overzicht van alle datavelden van een Watson Assistant chatbot wordt er verwezen naar tabel 4.1.

De dialognodes zijn aan elkaar verbonden met een ouder-kind relatie en vormen bijgevolg een hiërarchische structuur. De flows kunnen daardoor voorgesteld worden als bomen, of netwerken wanneer deze gericht zijn. Er kunnen in principe cyckels ontstaan of connecties met andere bomen/netwerken gemaakt worden wanneer men ervoor opteert om de “jump-to”-relaties op te nemen in de visualisatie. In dit geval moeten de relaties duidelijk onderscheidbaar zijn van elkaar. Dit geeft een antwoord op de derde deelvraag.

Deze drie deelvragen helpen bij het identificeren van correcte visualisaties. Dit onderzoek stelt voor om volgende visualisaties te gebruiken voor een chatbotflow:

- dendrogram
- (gericht) netwerk
- arc diagram
- treemap
- circular packing
- sunburst diagram

Dendrogrammen zijn in elke situatie bruikbaar en worden daarom steeds aangeraden om te gebruiken alvorens er wordt gekeken naar een andere visualisatie. Treemaps, circular packings en sunburst zijn interessant wanneer men de grootte van dialogs wenst te vergelijken en om een *deel-van-een-geheel* overzicht te krijgen. Ze zijn echter niet aan te raden wanneer de “jump-to”-relaties getoond moeten worden.

Gerichte netwerken zijn in feite een minder overzichtelijke versie van een dendrogram en zouden nooit verkozen mogen worden boven een dendrogram. De arc diagram heeft, net als netwerken, het nadeel dat het onbruikbaar is wanneer de dataset te groot wordt. Het kan wel de “jump-to”-relaties overzichtelijk weergeven wanneer ze bijvoorbeeld onder de aas worden getrokken.

9.2 Hoofdonderzoeksvergadering Bewerkingen/Operaties

De laatste deelonderzoeksvergadering onderzoekt het soort operaties dat kan toegepast worden op een visualisatie. De volgende zeven types werden besproken:

- Details-on-demand
- Extract
- Filter
- History
- Overview
- Relatea
- Zoom

Deze operaties leiden tot de volgende conclusie voor de tweede hoofdonderzoeksvergadering.

De belangrijkste operatie is het *in- en uitzoomen* op een visualisatie. Naarmate een visualisatie groter wordt, wordt de nood om te zoomen en alles weer leesbaar te maken, steeds groter. Daarnaast zijn de operaties *extract* en *filter* van belang om de hoeveelheid aan informatie in een visualisatie te beperken. *Details-on-demand* speelt hierin ook een groter rol door belangrijke, maar niet essentiële, informatie te verbergen tot zolang er niet naar wordt gevraagd. Op deze manier blijft de visualisatie overzichtelijk.

De *relate*-operatie kan een belangrijke rol spelen wanneer de “jump-to”-relaties moeten getoond worden. Ten slotte is elke visualisatie, na het genereren van een chatbotflow, een *overview* en is de *history*-operatie een optionele operatie die weggelaten kan worden als de filters en extracts omkeerbaar zijn door opnieuw een filter/extract-operatie uit te voeren.

9.3 Prototype

Het prototype is geschreven met de programmeertaal JavaScript. Dit is dezelfde taal die ook achter de gebruikte softwarebibliotheek D3.js zit. Deze bibliotheek maakt het mogelijk om interactieve visualisaties te maken. De gegenereerde visualisatie is een SVG-bestand.

SVG-bestanden behoren tot de familie van vectorafbeeldingen. Zoals in Hoofdstuk 2 vermeld werd zijn vectorafbeeldingen ideaal om in -en uit te zoomen en ondersteunen deze bijgevolg de zoom-operatie.

De invoer van dit prototype is het JSON-(export)bestand dat eerder werd besproken in Hoofdstuk 3 en waarvan een deel teruggevonden kan worden in Bijlage B. Dit bestand ondergaat eerst wat transformaties (geschreven in JavaScript) om een JSON-formaat te bereiken dat zonder problemen kan geïnterpreteerd worden door de D3 library. Dit transformatiescript wordt besproken in Bijlage C.

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 Terminologie

A.1.1 Artificiële Intelligentie

Artificiële intelligentie, ook kunstmatige intelligentie of kortweg AI, heeft geen éénduidige definitie. Er zijn doorheen de tijd heel wat definities aan toegekend. In deze paper wordt er gekozen om het model van Russel & Norvig te volgen (Figuur A.1).

Het model beschrijft artificiële intelligentie als een combinatie van vier concepten: menselijk denken, menselijk handelen, rationeel denken en rationeel handelen.

Menselijk denken gaat dieper in op het modelleren van het menselijk brein. A.d.h.v. van dit model probeert men de denkwijze van mensen te achterhalen (Kent, 2019).

Het concept van menselijk handelen kwam tot stand toen Alan Turing de Turing test creëerde om te onderzoeken of een computer een menselijke ondervrager kan misleiden door zich voor te doen als een mens. De test slaagt als de ondervrager geen onderscheid kan maken tussen mens en computer (Stuart & Peter, 2016).

Het derde gebied is rationeel denken, waarbij de redeneringsprocessen en de bijhorende conclusies worden beschreven. Bij rationeel handelen tracht men rationele agenten op

<p>Thinking Humanly</p> <p>“The exciting new effort to make computers think . . . <i>machines with minds</i>, in the full and literal sense.” (Haugeland, 1985)</p> <p>“[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . .” (Bellman, 1978)</p>	<p>Thinking Rationally</p> <p>“The study of mental faculties through the use of computational models.” (Charniak and McDermott, 1985)</p> <p>“The study of the computations that make it possible to perceive, reason, and act.” (Winston, 1992)</p>
<p>Acting Humanly</p> <p>“The art of creating machines that perform functions that require intelligence when performed by people.” (Kurzweil, 1990)</p> <p>“The study of how to make computers do things at which, at the moment, people are better.” (Rich and Knight, 1991)</p>	<p>Acting Rationally</p> <p>“Computational Intelligence is the study of the design of intelligent agents.” (Poole <i>et al.</i>, 1998)</p> <p>“AI . . . is concerned with intelligent behavior in artifacts.” (Nilsson, 1998)</p>

Figuur A.1: Indeling van definities van artificiële intelligentie

Source: <https://eetn.eu/knowledge/detail/Evidence-Summary–Artificial-Intelligence-in-education>

te stellen: een agent die de beste uitkomst bekomt. Deze uitkomst is niet per se de best mogelijke uitkomst maar de beste uitkomst volgens de kennis die de agent op dat moment heeft. Een agent kan simpelweg beschreven worden als een entiteit die zijn omgeving kan waarnemen door gebruik te maken van sensoren en invloed kan uitoefenen binnen diezelfde omgeving (Lievens, 2019).

A.1.2 Chatbot

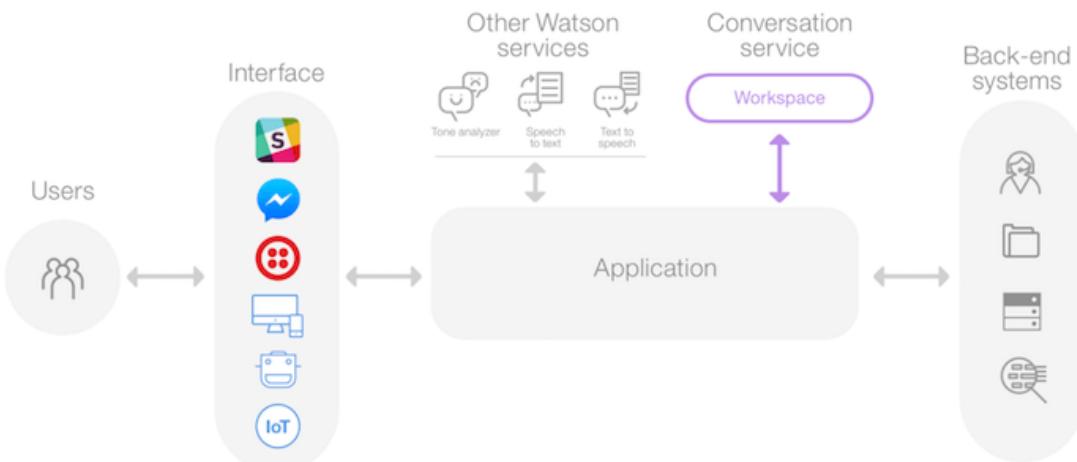
Een chatbot kan gedefinieerd worden als een programma, dat door op basis van menselijke interactie¹, conversaties simuleert (Rouse, 2019). Een IBM Watson² chatbot bestaat uit meerdere onderdelen: de voornaamste onderdelen zijn:

- intent
- entity
- dialog

Een intent is een verzameling van mogelijke gebruikersinput met een gemeenschappelijk doel. Entities worden gezien als een onderdeel van een gebruikersinput en worden gebruikt om synoniemen of gelijkaardige definities te herkennen en te labelen. Tenslotte kan dialog gezien worden als de flow van interacties met de gebruiker, die opgesteld wordt op basis

¹door middel van tekst of spraak

²dit chatbot framework wordt gebruikt door Clever en is dus het meest relevante voor het onderzoek



Figuur A.2: De werking van een IBM Watson Assistant chatbot

Source: https://www.ibm.com/cloud/architecture/tutorials/watson_conversation_support

van intents en entities (IBM, 2020k).

A.1.3 Natural Language Processing

NLP, of Natural Language Processing³, is een onderdeel van het onderzoeksgebied artificiële intelligentie dat zich bezighoudt met het verstaanbaar maken van mensentaal voor een computer. Het omvat zaken zoals language modeling, parsing en morphology. (Otter e.a., 2019).

NLP ligt aan de basis van elke chatbot en is noodzakelijk om gepast te kunnen reageren. Wanneer een chatbot gebruikersinput ontvangt via een interface, zoals bijvoorbeeld Messenger, dan zal via NLP de boodschap vertaald worden naar een probleem dat verstaanbaar is voor het programma. Wanneer de input voor het programma duidelijk is, kan de bot op zoek gaan naar een correct antwoord in het achterliggend systeem. (Figuur A.2).

A.2 Introductie

Chatbots worden steeds meer gebruikt in een professionele context, door zowel grote bedrijven als kleine ondernemingen. Chatbots kunnen overal teruggevonden worden: van het bestellen van eten⁴, ondersteunen van de klantenservice⁵ of simpelweg een afspeellijst van je favoriete muziek maken⁶.

Clever is een consultancybedrijf dat chatbots ontwikkelt voor klanten. De communicatie

³vanaf hier wordt steeds NLP gebruikt

⁴facebook.com/Dominos

⁵bol.com/nl/klantenservice

⁶rythmbot.co

met hun klanten is helaas niet eenvoudig omdat de visualisatie van de werking, specifiek de chatbot flows, niet visueel zichtbaar zijn. Voor iemand zonder een IT-achtergrond kan het zeer moeilijk zijn om zo een flow voor te stellen en de complete werking van de chatbot te begrijpen. Het handmatig uittekenen van deze structuur is tijdrovend en de informatie ervan is snel verouderd.

Er is nood aan een softwareoplossing om de communicatie tussen de klanten en ontwikkelaars, gedurende het volledig proces, te verbeteren. Deze software zou de flows van hun bots moeten omzetten naar een visuele weergave. De visualisaties moeten ook interactief zijn om bijvoorbeeld annotaties te geven aan nodes binnen de flow of extra info van een node weer te geven. De software dient ook correct met de gewenste input overweg kunnen.

Volgende onderzoeksvragen worden beantwoord:

- Welke elementen zijn nodig voor de opbouw van een chatbot flow?
- Hoe kunnen deze elementen visueel worden voorgesteld?
- Welke JavaScript libraries kunnen gebruikt worden voor visuele voorstelling van de chatbot flow?
- Welke data visualisaties zijn relevant en worden reeds gebruikt voor het voorstellen van een chatbot flow?
- Welke visualisatie frameworks bestaan er voor de visualisatie van chatbot flows?

A.3 State-of-the-art

Hoewel vandaag veel populairder, bestaan Chatbots al meer dan vijftig jaar⁷. Er zijn heel wat frameworks op de markt voor het maken van chatbots. Enkele voorbeelden hiervan zijn:

- Dialogflow⁸
- IBM Watson⁹
- Microsoft Bot Frameworks¹⁰
- BotPress¹¹
- ChatterBot¹²

Verschillende van deze frameworks voorzien zelf al enkele mogelijkheden voor het tonen van data visualisaties. Dialogflow kan bijvoorbeeld de recente flow geven en tonen welke functionaliteiten het vaakst worden gebruikt (Figuur A.3).

Clever maakt gebruik van IBM Watson Assistant voor de NLP van hun chatbots. Daarnaast

⁷eerste chatbot, ELIZA gecreëerd in 1966

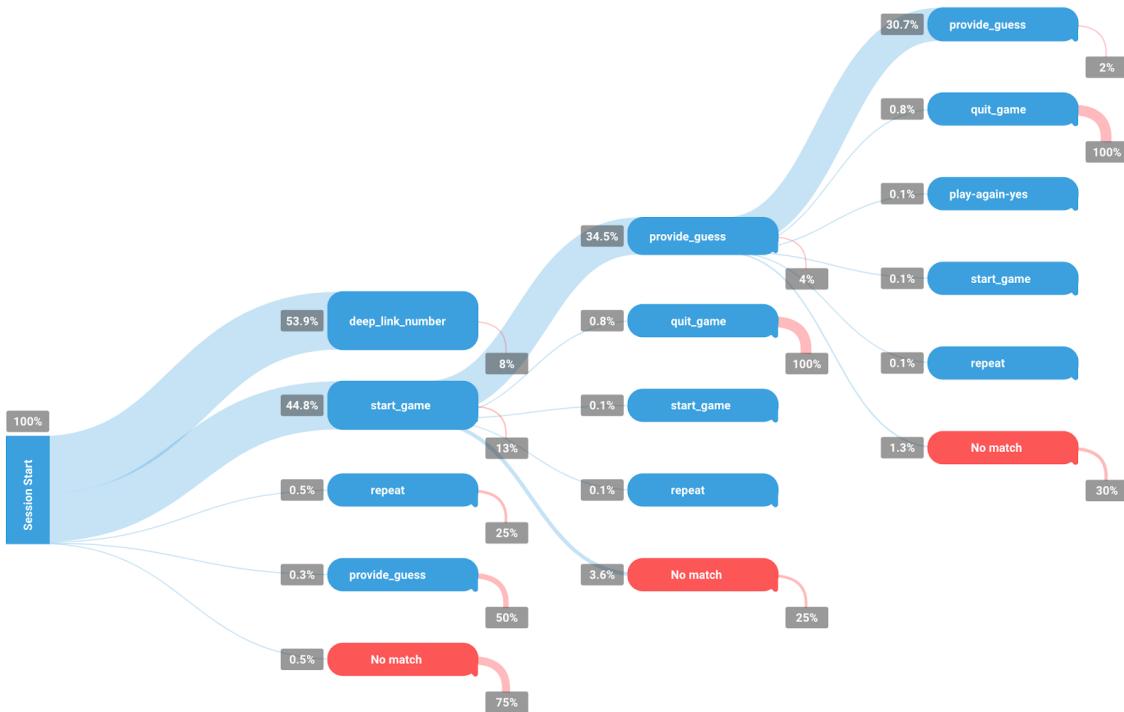
⁸dialogflow.com

⁹ibm.com/watson/products-services

¹⁰dev.botframework.com

¹¹botpress.io

¹²chatterbot.readthedocs.io/en/stable



Figuur A.3: De sessie flow van een Dialogflow chatbot visueel weergegeven

Source: <https://medium.com/google-developers/analytics-for-actions-d8212a6bf6a5>

werken ze met de MERN¹³ stack voor de ontwikkeling van hun applicaties. MongoDB werkt aan de hand van JSON¹⁴ bestanden wat de input zal zijn voor het prototype (MongoDB, 2019).

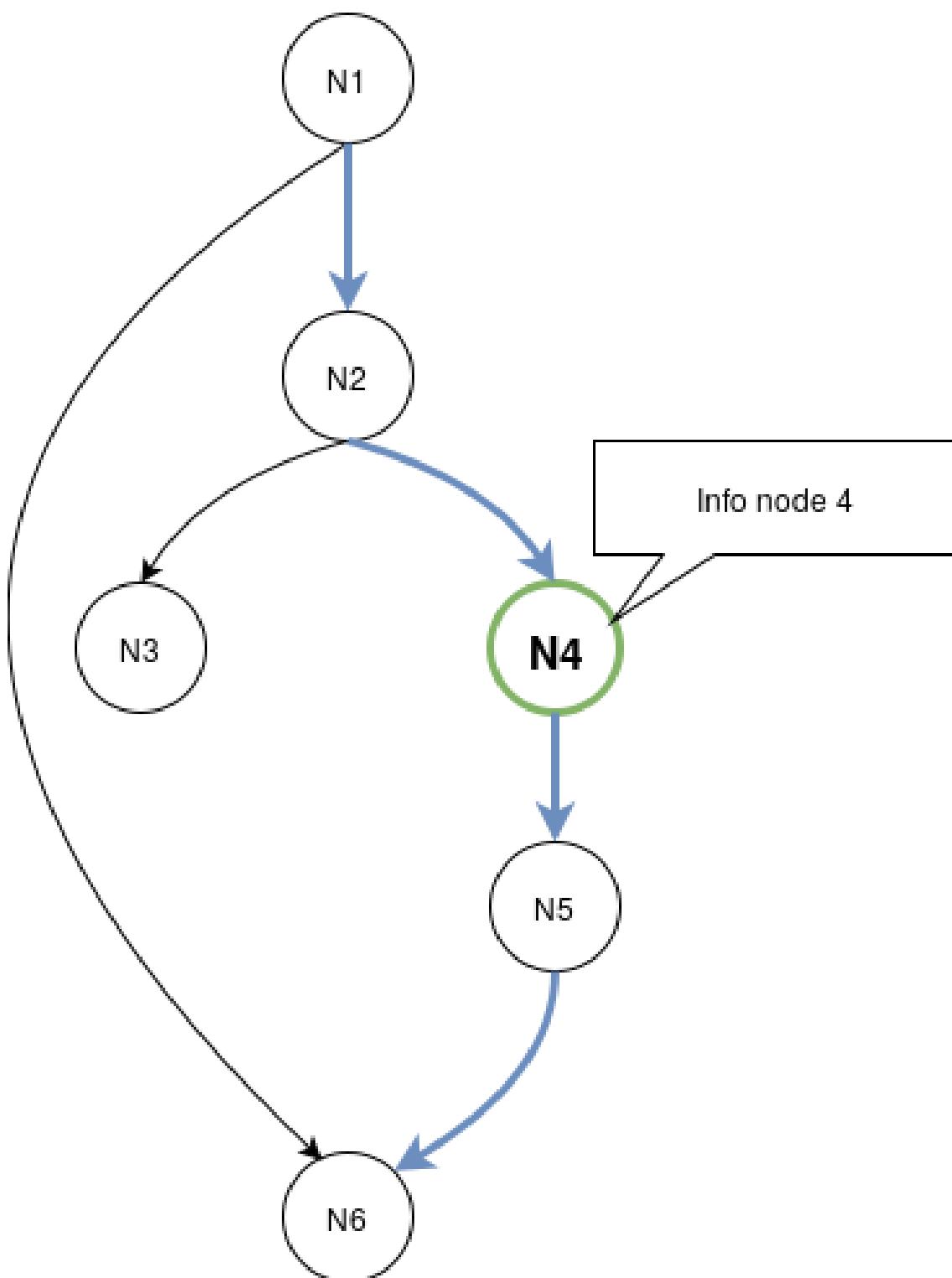
A.4 Methodologie

Het onderzoek start met een uitgebreide literatuurstudie naar hoe een IBM Watson Assistant chatbot werkt, uit welke onderdelen een flow bestaat en hoe deze kunnen omgezet worden naar een visuele representatie. De technologieën die er al zijn voor het visualiseren van deze flows worden onderzocht om zo een goed beeld te hebben van hoe het prototype zou moeten werken en welke functionaliteiten er mogelijk zijn. Hierbij wordt er niet enkel gekeken naar visualisaties van een IBM chatbot maar ook naar chatbots in het algemeen.

Daarna wordt er gezocht naar relevante libraries voor het implementeren van de JavaScript software. Als laatste wordt er een prototype opgesteld voor het visualiseren van een gegeven JSON-bestand en wordt er een korte handleiding geschreven. Voor het prototype krijgt functionaliteit voorrang op kwantiteit en opmaak.

¹³MongoDB, Express, React and Node.js

¹⁴opslag gebeurt a.d.h.v. BSON bestanden (Binary-Encoded JSON)



Figuur A.4: Een mogelijke voorstelling van een interactieve graaf

A.5 Verwachte resultaten

Er zijn verschillende soorten visualisaties voor het weergeven van een chatbotflow. Decision trees en flowchart zijn voorbeelden hiervan (of Mechanical Engineers, 1947). Voor het prototype is een boom- of graafstructuur de beste keuze. Er zal ook rekening mee moeten gehouden worden dat de visualisaties dynamisch zijn zodat er interactief mee gewerkt kan worden. Het prototype zal dus niet enkel het éénmalig mappen zijn van een flow, maar systematisch opnieuw gebouwd worden naar wens van de gebruiker. Naarmate het onderzoek en het prototype vordert kan het aantal soorten visualisatie of de functionaliteiten ervan nog wijzigen. Een voorstelling van zo een interactieve visualisatie kan teruggevonden worden in Figuur A.4.

A.6 Verwachte conclusies

Voor het maken van het prototype is de best mogelijke uitkomst een combinatie van een of meerdere libraries die het visueel aspect van de prototype kunnen vergemakkelijken. De libraries zullen dus moeten kunnen werken met JSON-bestanden of er zal nog een conversiefunctie voorzien moeten worden. Het prototype voldoet aan de vooropgestelde eisen waarbij de focus is gelegd op de correcte werking functionaliteit en duidelijkheid, en minder op het aantal functionaliteiten en opmaak.

B. Exportbestand Watson Assistant

In deze bijlage kan het exportbestand van Sherlock weergegeven. Sherlock is een niet-gepubliceerde IBM Watson Assistant chatbot dat opgesteld is in functie van het onderzoek. Vanwege de omvang van het JSON-bestand, worden deze onderverdeeld in meerdere secties. Deze secties worden besproken in Hoofdstuk 4.

B.1 Basis Structuur

De basis structuur van elke IBM Watson Assistant chatbot ziet er als volgt uit:

```
{  
    "intents": [  
        ...  
    ],  
    "entities": [  
        ...  
    ],  
    "metadata": {  
        ...  
    },  
}
```

```
"dialog_nodes": [  
    ...  
],  
"counterexamples": [  
    ...  
],  
"system_settings": {  
    ...  
},  
"learning_opt_out": "",  
"name": "",  
"language": "",  
"description": ""  
}
```

B.2 Chatbot Informatie

De volgende code is een beschrijving van de chatbot zelf, niet de inhoud ervan.

```
...  
"metadata": {  
    "api_version": {  
        "major_version": "v2",  
        "minor_version": "2018-11-08"  
    }  
},  
...  
"counterexamples": [],  
"system_settings": {  
    "off_topic": {  
        "enabled": true  
    },  
    "disambiguation": {  
        "prompt": "Did you mean:",  
        "enabled": true,  
        "randomize": true,  
    }  
}
```

```
        "max_suggestions": 5,
        "suggestion_text_policy": "title",
        "none_of_the_above_prompt": "None of the above"
    },
    "human_agent_assist": {
        "prompt": "Did you mean:"
    },
    "spelling_auto_correct": true
},
"learning_opt_out": false,
"name": "My first skill",
"language": "en",
"description": ""
```

B.3 Chatbot Intents

Een overzicht van enkele intents van Sherlock kunnen hieronder teruggevonden worden.

```
"intents": [
{
    "intent": "CupcakeSamenstellen",
    "examples": [
        {
            "text": "ik wil een cupcake maken"
        },
        {
            "text": "mag ik een cupcake maken voor een vriend"
        },
        {
            "text": "kan ik een cupcake samenstellen"
        },
        {
            "text": "is het mogelijk om een cupcake te maken"
        },
        {
            "text": "ik wil zelf een maken"
        },
        {
            "text": "Ik wil zelf een cupcake samenstellen"
        },
        {
            "text": "Ik wil zelf een cupcake maken"
        }
    ],
}
```

```
"description": "De gebruiker wenst zelf een cupcake samen te
    ↪ stellen"
},
{
    "intent": "ReservatieMaken",
    "examples": [
        {
            "text": "Reservatie voor vier"
        },
        {
            "text": "Ik zou willen reserveren voor vanavond"
        },
        {
            "text": "ik wil boeken voor een etentje"
        },
        {
            "text": "kan ik een tafel reserveren?"
        },
        {
            "text": "kan ik een etentje boeken"
        },
        {
            "text": "Zou ik kunnen een reservatie plaatsen voor
                ↪ vanmiddag?"
        },
        {
            "text": "Ik zou een reservatie willen maken"
        }
    ],
    "description": ""
},
{
    "intent": "Bored",
    "examples": [
        {
            "text": "super saai"
        },
        {
            "text": "ik ben verveeld"
        },
        {
            "text": "dit is zo saai"
        },
        {
            "text": "Kan je me helpen, ik verveel me"
        }
    ]
}
```

```
{  
    "text": "Ik verveel me"  
},  
{  
    "text": "I'm bored"  
},  
{  
    "text": "het is hier saai, kan je helpen?"  
},  
{  
    "text": "I'm super bored right now"  
},  
{  
    "text": "het is hier niet intressant"  
}  
],  
]  
,  
"description": ""  
},  

```

```
        "text": "ik wil een uitkiezen"
    },
    {
        "text": "uitkiezen"
    }
],
"description": "De gebruiker een cupcake van het menu te
    ↪ kiezen"
},
{
    "intent": "BezoekenCupcakeWinkel",
    "examples": [
        {
            "text": "ik wil een cupcake eten"
        },
        {
            "text": "ik wil een cupcake kopen"
        },
        {
            "text": "kan ik ergens cupcakes krijgen"
        },
        {
            "text": "is het mogelijk cupcakes te kopen"
        },
        {
            "text": "is het mogelijk cupcakes te verkrijgen"
        },
        {
            "text": "waar kan ik cupcakes kopen"
        },
        {
            "text": "waar kan ik cupcakes eten"
        },
        {
            "text": "kunnen we ergens cupcakes vinden"
        },
        {
            "text": "kunnen we cupcakes eten"
        },
        {
            "text": "verkoop je cupcakes"
        }
    ],
"description": "De gebruiker wenst een cupcake te komen"
},
```

...

B.4 Chatbot Entities

Volgende code toont de export van enkele entities.

```
...  
  
"entities": [  
{  
    "entity": "kaartType",  
    "values": [  
        {  
            "type": "synonyms",  
            "value": "maestro",  
            "synonyms": []  
        },  
        {  
            "type": "synonyms",  
            "value": "visa",  
            "synonyms": []  
        },  
        {  
            "type": "synonyms",  
            "value": "master",  
            "synonyms": [  
                "credit"  
            ]  
        }  
    ],  
    "fuzzy_match": true  
},  
{  
    "entity": "kaartOfCash",  
    "values": [  
        {  
            "type": "synonyms",  
            "value": "cash",  
            "synonyms": [  
                "contant",  
                "munten",  
                "briefjes",  
                "kleingeld"  
            ]  
        }  
    ]  
}]
```

```
        },
        {
            "type": "synonyms",
            "value": "kaart",
            "synonyms": [
                "card",
                "met kaart"
            ]
        }
    ],
    "fuzzy_match": true
},
{
    "entity": "deeg",
    "values": [
        {
            "type": "synonyms",
            "value": "Deeg op grootmoeders wijze",
            "synonyms": [
                "grootmoeder",
                "homemade",
                "eigen recept",
                "eigen"
            ]
        },
        {
            "type": "synonyms",
            "value": "Verrassingsdeeg",
            "synonyms": [
                "verrassing",
                "surprise",
                "geheim"
            ]
        },
        {
            "type": "synonyms",
            "value": "Standaard deeg",
            "synonyms": [
                "standaard",
                "gewoon",
                "normaal",
                "basis",
                "klassieke"
            ]
        }
    ],
}
```

```
        "fuzzy_match": true
    },
    {
        "entity": "sprinkels",
        "values": [
            {
                "type": "synonyms",
                "value": "vanille",
                "synonyms": [
                    "witte",
                    "lichte"
                ]
            },
            {
                "type": "synonyms",
                "value": "chocolade",
                "synonyms": [
                    "choco",
                    "zwarte",
                    "donkere"
                ]
            },
            {
                "type": "synonyms",
                "value": "regenboog",
                "synonyms": [
                    "rainbow",
                    "alle kleuren",
                    "kleurtjes"
                ]
            }
        ],
        "fuzzy_match": true
    },
    {
        "entity": "cookiesTopping",
        "values": [
            {
                "type": "synonyms",
                "value": "bueno",
                "synonyms": []
            },
            {
                "type": "synonyms",
                "value": "mars",
                "synonyms": []
            }
        ]
    }
}
```

```
},
{
  "type": "synonyms",
  "value": "chocolate chip",
  "synonyms": [
    "chip cookies",
    "amerikaans"
  ]
},
{
  "type": "synonyms",
  "value": "oreo",
  "synonyms": []
}
],
"fuzzy_match": true
},
{
  "entity": "StandaardCupcakes",
  "values": [
    {
      "type": "synonyms",
      "value": "framboos-sprinkels cupcake",
      "synonyms": [
        "raspberry-sprinkels",
        "raspberry en sprinkels",
        "sprinkels en framboos",
        "sprinkels framboos"
      ]
    },
    {
      "type": "synonyms",
      "value": "kers-oreo cupcake",
      "synonyms": [
        "kers + oreo",
        "kers en oreo",
        "cherry and oreo"
      ]
    },
    {
      "type": "synonyms",
      "value": "aardbei-cookies cupcake",
      "synonyms": [
        "aardbei en cookies cupcake",
        "strawberry cookies",
        "strawberry/koekjes cupcake",
        "aardbei cookies cupcake"
      ]
    }
  ]
}
```

```
        "aardbei en koekjes"
    ]
}
],
"fuzzy_match": true
},
{
"entity": "KeuzeMenuOptie",
"values": [
{
    "type": "synonyms",
    "value": "Optie1",
    "synonyms": [
        "1",
        "eerste"
    ]
},
{
    "type": "synonyms",
    "value": "Optie2",
    "synonyms": [
        "2"
    ]
},
{
    "type": "synonyms",
    "value": "Optie3",
    "synonyms": [
        "3"
    ]
}
],
"fuzzy_match": false
},
{
"entity": "Movies",
"values": [
{
    "type": "synonyms",
    "value": "Lord of the Rings",
    "synonyms": [
        "lotr",
        "lord ot rings"
    ]
},
{

```

```

        "type": "synonyms",
        "value": "Star Wars",
        "synonyms": [
            "revenge of the sith",
            "the phantom menace",
            "attack of the clones",
            "a new hope",
            "the empire strikes back",
            "return of the jedi"
        ]
    },
    {
        "type": "synonyms",
        "value": "Harry Potter",
        "synonyms": [
            "Harry P"
        ]
    }
],
"fuzzy_match": true
},
],
...

```

B.5 Chatbot Nodes

Als laatste zijn er nog de nodes van de chatbot. Enkele voorbeelden kunnen hieronder teruggevonden worden.

```

...
"dialog_nodes": [
{
    "type": "standard",
    "title": "betaal met kaart",
    "output": {
        "generic": [
        {
            "title": "En met welke kaart wenst u te
            ↢ betalen?",
            "options": [
            {
                "label": "Maestro",

```

```
        "value": {
            "input": {
                "text": "maestro"
            }
        }
    },
{
    "label": "Master",
    "value": {
        "input": {
            "text": "master"
        }
    }
},
{
    "label": "Visa",
    "value": {
        "input": {
            "text": "visa"
        }
    }
}
],
"response_type": "option"
}
]
},
"parent": "node_7_1587131892229",
"conditions": "@kaartOfCash:kaart",
"dialog_node": "node_8_1587133522792"
},
{
    "type": "slot",
    "parent": "node_4_1583185372461",
    "variable": "$numbers",
    "dialog_node": "slot_3_1583185423484",
    "previous_sibling": "handler_2_1583185422391"
},
{
    "type": "event_handler",
    "parent": "node_4_1583185372461",
    "event_name": "focus",
    "dialog_node": "handler_2_1583185422391"
},
{
    "type": "slot",
```

```

    "parent": "node_2_1587132643052",
    "variable": "$StandaardCupcakes",
    "dialog_node": "slot_10_1587132866089",
    "previous_sibling": "node_2_1587134674646"
},
{
    "type": "standard",
    "title": "Bestel meer",
    "parent": "node_2_1587132643052",
    "next_step": {
        "behavior": "jump_to",
        "selector": "body",
        "dialog_node": "node_5_1586610239372"
    },
    "conditions": "@KeuzeMenuOptie:Optie2",
    "dialog_node": "node_2_1587134674646",
    "previous_sibling": "node_7_1587134554450"
},
{
    "type": "standard",
    "title": "Ga naar afrekenen",
    "output": {
        "generic": [
            {
                "values": [
                    {
                        "text": "Ik stuur u door naar de
                            ↗ kassa!"
                    },
                    {
                        "text": "Top! Volgt u mij tot aan
                            ↗ de kassa?"
                    }
                ],
                "response_type": "text",
                "selection_policy": "sequential"
            }
        ]
    },
    "parent": "node_2_1587132643052",
    "next_step": {
        "behavior": "jump_to",
        "selector": "body",
        "dialog_node": "node_7_1587131892229"
    },
    "conditions": "@KeuzeMenuOptie:Optie1",
}

```

```
        "dialog_node": "node_7_1587134554450"
    },
    {
        "type": "event_handler",
        "output": {
            "text": {
                "values": [
                    "And for how many?"
                ],
                "selection_policy": "sequential"
            }
        },
        "parent": "slot_3_1583185423484",
        "event_name": "focus",
        "dialog_node": "handler_1_1583185423489",
        "previous_sibling": "handler_2_1583185423489"
    },
    {
        "type": "event_handler",
        "output": {},
        "parent": "slot_3_1583185423484",
        "context": {
            "numbers": "@numbers"
        },
        "conditions": "@numbers",
        "event_name": "input",
        "dialog_node": "handler_2_1583185423489"
    },
    {
        "type": "event_handler",
        "output": {},
        "parent": "slot_10_1587132866089",
        "event_name": "focus",
        "dialog_node": "handler_2_1587132866096",
        "previous_sibling": "handler_5_1587132866096"
    },
    {
        "type": "event_handler",
        "output": {},
        "parent": "slot_10_1587132866089",
        "context": {
            "StandaardCupcakes": "@StandaardCupcakes"
        },
        "conditions": "@StandaardCupcakes",
        "event_name": "input",
        "dialog_node": "handler_5_1587132866096"
```

```
},
{
  "type": "standard",
  "title": "cupcake uitkiezen",
  "output": [
    {
      "generic": [
        {
          "title": "Perfect! en welke cupcake  
↪ wenst u?",  

          "options": [
            {
              "label": "Kers/Oreo",
              "value": {
                "input": {
                  "text": "Framboos-  
↪ oreo cupcake  
↪ "
                }
              }
            },
            {
              "label": "Framboos/Sprinkels",
              "value": {
                "input": {
                  "text": "Framboos-  
↪ sprinkels  
↪ cupcake"
                }
              }
            },
            {
              "label": "Aardbei/Cookies",
              "value": {
                "input": {
                  "text": "Aardbei-  
↪ cookies  
↪ cupcake"
                }
              }
            }
          ],
          "response_type": "option"
        }
      ]
    },
    "parent": "node_5_1586608437803",
```

```
        "conditions": "#CupcakeUitkiezen",
        "dialog_node": "node_5_1586610239372",
        "previous_sibling": "node_4_1586608874360"
    },
    {
        "type": "standard",
        "title": "zelf samenstellen",
        "parent": "node_5_1586608437803",
        "conditions": "#CupcakeSamenstellen",
        "dialog_node": "node_4_1586608874360"
    },
    {
        "type": "standard",
        "title": "betalen met visa",
        "output": {
            "generic": [
                {
                    "values": [
                        {
                            "text": "Met Visa? Geen probleem!
                                ↪ Betaling is gelukt!"
                        },
                        {
                            "text": "Visa? Okido! Betaling
                                ↪ geslaagd!"
                        }
                    ],
                    "response_type": "text",
                    "selection_policy": "sequential"
                }
            ]
        },
        "parent": "node_8_1587133522792",
        "next_step": {
            "behavior": "jump_to",
            "selector": "body",
            "dialog_node": "node_4_1587134014122"
        },
        "conditions": "@kaartType:visa",
        "dialog_node": "node_4_1587134220657",
        "previous_sibling": "node_5_1587134004774"
    },
    {
        "type": "standard",
        "title": "betalen met master",
        "output": {
```

```

    "generic": [
    {
        "values": [
        {
            "text": "Okido. Als u juist uw
            ↪ gegevens wil doorgeven...
            ↪ Top! Dat is in orde!"
        },
        {
            "text": "Even uw gegevens
            ↪ valideren.. Betaling is
            ↪ gelukt!"
        }
    ],
        "response_type": "text",
        "selection_policy": "sequential"
    }
]
},
"parent": "node_8_1587133522792",
"next_step": {
    "behavior": "jump_to",
    "selector": "body",
    "dialog_node": "node_4_1587134014122"
},
"conditions": "@kaartType:master",
"dialog_node": "node_5_1587134004774",
"previous_sibling": "node_8_1587133920384"
},
{
    "type": "standard",
    "title": "betalen met maestro",
    "output": {
        "generic": [
        {
            "values": [
            {
                "text": "U heeft betaald met
                ↪ Maestro!"
            },
            {
                "text": "Betaling met Maestro
                ↪ gelukt!"
            }
        ],
        "response_type": "text",
    }
}

```

```
        "selection_policy": "sequential"
    }
]
},
"parent": "node_8_1587133522792",
"next_step": {
    "behavior": "jump_to",
    "selector": "body",
    "dialog_node": "node_4_1587134014122"
},
"conditions": "@kaartType:maestro",
"dialog_node": "node_8_1587133920384"
},
{
    "type": "frame",
    "title": "Confirm keuze",
    "output": {
        "generic": [
        {
            "title": "U heeft gekozen voor
                ↪ $StandaardCupcakes, wenst u af te
                ↪ rekenen of iets anders te
                ↪ bestellen?",
            "options": [
            {
                "label": "Ik wil afrekenen",
                "value": {
                    "input": {
                        "text": "Optie1"
                    }
                }
            },
            {
                "label": "Ik wil meer bestellen",
                "value": {
                    "input": {
                        "text": "Optie2"
                    }
                }
            }
        ],
        "response_type": "option"
    }
}
],
"parent": "node_5_1586610239372",
```

```
"conditions": "@StandaardCupcakes",
"dialog_node": "node_2_1587132643052"
},
{
  "type": "standard",
  "title": "Bedanking",
  "output": {
    "generic": [
      {
        "values": [
          {
            "text": "bedankt om te shoppen
              ↳ bij Sherlock. Nog een
              ↳ fijne dag."
          },
          {
            "text": "Dankuwel voor uw aankoop
              ↳ ! Nog een goede dag
              ↳ gewenst."
          }
        ],
        "response_type": "text",
        "selection_policy": "sequential"
      }
    ]
  },
  "dialog_node": "node_4_1587134014122",
  "previous_sibling": "Welcome"
},
{
  "type": "frame",
  "title": "Reserveren",
  "output": {
    "generic": [
      {
        "values": [],
        "response_type": "text",
        "selection_policy": "sequential"
      },
      {
        "values": [],
        "response_type": "text",
        "selection_policy": "sequential"
      }
    ]
  },
  "dialog_node": "node_5_1587134014122"
}
```

```
        "conditions": "#ReservatieMaken",
        "dialog_node": "node_4_1583185372461",
        "previous_sibling": "node_7_1583184097089"
    },
    {
        "type": "standard",
        "title": "Starten cupcake shop",
        "output": [
            {
                "generic": [
                    {
                        "values": [
                            {
                                "text": "Hallo, welkom bij
                                    ↳ Sherlock's Cupcakeshop, de
                                    ↳ beste cupcake winkel ter
                                    ↳ wereld. Hier kan u zelf
                                    ↳ cupcakes maken of een
                                    ↳ standaard exemplaar kiezen
                                    ↳ ."
                            },
                            {
                                "text": "Goededag, wenst u zelf
                                    ↳ een cupcake samen te
                                    ↳ stellen of wilt u kiezen
                                    ↳ tussen een van onze reeds
                                    ↳ samengestelde cupcakes?"
                            },
                            {
                                "text": "Hallo, u kan kiezen door
                                    ↳ een op voorhand gemaakte
                                    ↳ cupcake of zelf een
                                    ↳ cupcake te maken."
                            }
                        ],
                        "response_type": "text",
                        "selection_policy": "sequential"
                    }
                ]
            },
            "conditions": "#BezoekenCupcakeWinkel",
            "dialog_node": "node_5_1586608437803",
            "previous_sibling": "node_7_1587131892229"
        },
        {
            "type": "standard",
            "title": "Afrekenen",
```

```

"output": {
    "generic": [
        {
            "values": [
                {
                    "text": "Wenst u te betalen met
                        ↳ kaart of cash?"
                },
                {
                    "text": "Welkom bij de kassa!
                        ↳ Betaalt u met kaart of in
                        ↳ cash?"
                }
            ],
            "response_type": "text",
            "selection_policy": "sequential"
        },
        {
            "title": "Ik betaal",
            "options": [
                {
                    "label": "met kaart",
                    "value": {
                        "input": {
                            "text": "kaart"
                        }
                    }
                },
                {
                    "label": "in cash",
                    "value": {
                        "input": {
                            "text": "cash"
                        }
                    }
                }
            ],
            "response_type": "option"
        }
    ]
},
"conditions": "",
"dialog_node": "node_7_1587131892229",
"previous_sibling": "node_4_1587134014122"
},
{

```

```
"type": "standard",
"title": "Counter boredom",
"output": [
    "generic": [
        {
            "values": [
                {
                    "text": "Hoe kan ik helpen?"
                },
                {
                    "text": "Waarmee kan ik je van  
→ dienst zijn?"
                },
                {
                    "text": "Hoe kan ik dat oplossen  
→ ?"
                },
                {
                    "text": "Hoe kan ik daar iets aan  
→ veranderen?"
                },
                {
                    "text": "Waarmee kan ik je helpen  
→ ?"
                }
            ]
        ],
        "response_type": "text",
        "selection_policy": "sequential"
    }
],
"conditions": "#Bored",
"dialog_node": "node_7_1583184097089",
"previous_sibling": "node_9_1583225688766"
},
{
    "type": "response_condition",
    "output": [
        "text": [
            "values": [
                "Ben je ook fan van Return of the King  
→ ?",
                "De Nazgul zijn het beste aan de films"
            ],
            "selection_policy": "sequential"
        ]
    ]
}
```

```
        },
        "parent": "node_9_1587740289103",
        "conditions": "@Movies:(Lord of the Rings)",
        "dialog_node": "response_4_1587740386710",
        "previous_sibling": "response_2_1587740363996"
    },
    {
        "type": "response_condition",
        "output": {
            "text": {
                "values": [
                    "Oh Harry Potter, dat is echt mijn jeugd!"
                ],
                "selection_policy": "sequential"
            }
        },
        "parent": "node_9_1587740289103",
        "conditions": "@Movies:(Harry Potter)",
        "dialog_node": "response_5_1587740495270",
        "previous_sibling": "response_4_1587740386710"
    },
    {
        "type": "response_condition",
        "output": {
            "text": {
                "values": [
                    "Oh ik ben ook fan van Star Wars.",
                    "Revenge of The Sith is de beste van de reeks"
                ],
                "selection_policy": "sequential"
            }
        },
        "parent": "node_9_1587740289103",
        "conditions": "@Movies:(Star Wars)",
        "dialog_node": "response_2_1587740363996"
    },
    {
        "type": "standard",
        "title": "MovieFan",
        "metadata": {
            "_customization": {
                "mcr": true
            }
        },
        "conditions": "@Movies",
        "dialog_node": "node_9_1587740289103",
```

```
        "previous_sibling": "Welcome"  
    },  
],  
...
```


C. JSON-Transformaties

Deze bijlage bevat scripts die gebruikt werden om het exportbestand in Bijlage B te transformeren naar een JavaScript-Object dat als import zal dienen voor de visualisatiescripts (dendrogram en sunburst). Deze scripts bestaan uit een reeks van de D3-operaties. Het formaat van het gecreëerde Object kan gemakkelijk gebruikt worden met D3.js en is dus ideaal voor de visualisatie scripts.

C.1 Overzicht van Transformatiescript

In deze sectie wordt de volledige structuur van het script weergegeven. Het Object *exportfile* bevat het volledige JSON-structuur vanuit Bijlage B.

```
let nodes = exportfile.dialog_nodes;

//map alle data naar een versimpeld model met enkel nuttige data
let nodes_simpel = nodes.map(n => ({
    name: n.title,
    cond: n.conditions,
    type: n.type,
    parent: n.parent,
    id: n.dialog_node,
    children: [],
    event: n.event_name,
    output: getOutputInfo(n)
}));
```

```
let inner_nodes = {};
let secondary_nodes = {};
let root_nodes = [];
fillCollections();

// verdeelfunctie waarbij er alle nodes over drie velden worden
// verdeeld
function fillCollections(){
    ...
}

// creatiefunctie waarbij er recursief nodes worden toegevoegd aan
// hun ouder
function createHierarchy(node) {
    ...
}

// hulpfunctie om informatie te verkrijgen over slots, events en
// conditional responses
function getSecondaryNodesInfo(node, childnode = null) {
    ...
}

// hulpfunctie om informatie te verkrijgen over output
function getOutputInfo(node) {
    ...
}

// exportfunctie voor visualisatiescripts
export function convert_json() {
    ...
}
```

C.2 Verdeelfunctie

Deze functie splits nodes op in drie categorieën:

- rootnodes
- standard- en framenodes
- ondersteunende nodes (slot, event, conditional response)

Zij worden respectievelijk verdeeld over volgende velden:

- root_nodes (Array)
- inner_nodes (Object)
- secondary_nodes (Object)

```
function fillCollections(){
    let temp;
    for (let i = 0; i < nodes_simpel.length; i++) {
        temp = nodes_simpel[i];
        if (temp.type === "standard" || temp.type === "frame")
            ↪ {
                if (temp.parent == undefined) {
                    root_nodes.push(temp);
                } else {
                    if (inner_nodes[temp.parent] ==
                        ↪ undefined) {
                        inner_nodes[temp.parent] = {
                            children: []
                        };
                    }
                    inner_nodes[temp.parent]
                        .children
                        .push(temp);
                }
            } else {
                if (secondary_nodes[temp.parent] == undefined)
                    ↪ {
                        secondary_nodes[temp.parent] = {
                            children: []
                        };
                    }
                secondary_nodes[temp.parent]
                    .children
                    .push(temp);
            }
        }
    }
}
```

C.3 Creatiefunctie voor Hiërarchische Structuur

Deze recursieve functie maakt de hiërarchische structuur (op basis van een meegegeven node) die nodig is voor de visualisatiescripts. De functie maakt gebruik van het veld “inner_nodes” dat eerder werd opgevuld in Functie C.2. De laatste *if* geeft alle bladeren eenzelfde waarde. Dit is nodig voor de sunburst-diagram.

```

function createHierarchy(node) {
    node = getSecondaryNodesInfo(node);
    let directChildren = inner_nodes[node.id] === undefined ? [] :
        ↪ inner_nodes[node.id].children;
    let entireLineage = [];
    for (let j = 0; j < directChildren.length; j++) {
        entireLineage.push(createHierarchy(directChildren[j]));
    }
    if(directChildren.length === 0){
        node.value = 1;
    }
    node.children = entireLineage;
    return node
}

```

C.4 Hulpfunctie voor Ondersteunende Nodes

In deze functie wordt voor alle *standard-* en *frame* nodes gekeken of er ondersteunende nodes tussen hun kinderen zitten. Volgende types worden als ondersteunende nodes beschouwd:

- slot
- event
- conditional response

Wanneer er zo een node gevonden wordt, dan zal de booleaanse waarde van het correspondeerde veld in de node op “true” worden gezet. Ondersteunende nodes kunnen zelf ook ondersteunende nodes hebben. Daarom wordt er in de laatste *if* een recursieve lus gestart wanneer dit het geval is. Dit is waarvoor de tweede parameter *childnode* dient. Wanneer deze ingevuld is weet men dat de startnode (parameter *node* van type “standard” of “frame”) reeds de juiste velden bevat.

```

function getSecondaryNodesInfo(node, childnode = null) {
    let secondary;
    if (childnode === null) {
        node.events = false;
        node.slots = false;
        node.condResp = false;
        node.eventnames = []
        secondary = secondary_nodes[node.id]
    } else {
        secondary = secondary_nodes[childnode.id];
    }
    let tempvalue;

```

```

    if (secondary !== undefined) {
        for (let i = 0; i < secondary.children.length; i++) {
            tempvalue = secondary.children[i];
            if (tempvalue.type === "event_handler") {
                node.events = true;
                node.eventnames.push(tempvalue.event)
            }
            if (tempvalue.type === "slot") {
                node.slots = true;
            }
            if (tempvalue.type === "response_condition") {
                node.condResp = true;
            }
            if (secondary_nodes[tempvalue.id] !== undefined
                ↪ ) {
                getSecondaryNodesInfo(node, tempvalue);
            }
        }
    }
    return node;
}

```

C.5 Hulpfunctie voor Output

Deze functie gaat op zoek naar welke “outputtypes” een gegeven node bevat. Er wordt een Set teruggegeven van alle gevonden outputtypes voor deze node. Wanneer een node wordt meegegeven dat ofwel geen output heeft, ofwel niet van het type “standard” of “frame” is, wordt er een lege Set teruggegeven.

```

function getOutputInfo(node) {
    let output = new Set();
    if ((node.type !== "standard" && node.type !== "frame") ||
        ↪ node.output == undefined) {
        return output;
    }
    let gen = node.output.generic;
    for (let i = 0; i < gen.length; i++) {
        output.add(gen[i].response_type.toUpperCase());
    }
    return output;
}

```

C.6 Exportfunctie voor Visualisatiescripts

Deze functie geeft een Object terug dat compatibel is met D3-operaties en zal aangeroepen worden door de visualisatiescripts

```
export function convert_json() {
    for (let i = 0; i < root_nodes.length; i++) {
        recursive_hierarchy(root_nodes[i]);
    }
    return {
        name: "root",
        children: root_nodes
    };
}
```

Bibliografie

- Amazon. (2020a, maart 28). *Amazon Lex FAQs: Deployment* (Amazon, Red.). <https://aws.amazon.com/lex/faqs/>
- Amazon. (2020b, maart 28). *Amazon Lex resources* (Amazon, Red.). <https://aws.amazon.com/lex/resources/?nc=sn&loc=5>
- Amazon. (2020c, maart 28). *Amazon Lex: Conversational interfaces for your applications powered by the same deep learning technologies as Alexa* (Amazon, Red.). <https://aws.amazon.com/lex>
- Amazon. (2020d, maart 29). *Custom Slot Types* (Amazon, Red.). <https://docs.aws.amazon.com/lex/latest/dg/howitworks-custom-slots.html>
- Amazon. (2020e, maart 28). *Deploying an Amazon Lex Bot in Mobile Applications* (Amazon, Red.). <https://docs.aws.amazon.com/lex/latest/dg/example2.html>
- Amazon. (2020f, maart 28). *Deploying an Amazon Lex Bot on a Messaging Platform* (Amazon, Red.). <https://docs.aws.amazon.com/lex/latest/dg/example1.html>
- Bikakis, N. (2018). Big Data Visualization Tools. *Encyclopedia of Big Data Technologies*.
- Bostock, M. (2012, maart 18). *Hive Plots* (M. Bostock, Red.). <https://bost.ocks.org/mike/hive/>
- Card, S., Mackinlay, J. & Shneiderman, B. (1999, januari 1). *Readings in Information Visualization: Using Vision To Think*.
- Dallwitz, M. J. (2010, mei 17). *An introduction to computer images* (M. J. Dallwitz, Red.). <https://www.delta-intkey.com/www/images.htm>
- DeplerAI. (2019, april 5). *Chatbots vs Virtual Assistants: What Is The Difference?* (DeplerAI, Red.). <https://chatbotslife.com/chatbots-vs-virtual-assistants-what-is-the-difference-f23287e32165>
- Dialogflow. (2020, maart 28). *Build natural and rich conversational experiences* (Dialogflow, Red.). <https://dialogflow.com/>

- discover.bot. (2019, augustus 19). *Building Bots with the Rasa Framework* (discover.bot, Red.). <https://discover.bot/bot-talk/guide-to-bot-frameworks/rasa/>
- Domo. (2017, juli 17). *Data Never Sleeps 5.0* (Domo, Red.). https://www.domo.com/learn/data-never-sleeps-5?aid=ogsm072517_1&sf100871281=1
- Ferdio. (2020a, april 26). *Arc Diagram* (Ferdio, Red.). <https://datavizproject.com/datatype/arc-diagram/>
- Ferdio. (2020b, april 27). *Dendrogram* (Ferdio, Red.). <https://datavizproject.com/datatype/dendrogram/>
- Figma. (2020, april 2). *Raster Graphics* (Figma, Red.). <https://www.figma.com/dictionary/raster-graphics/>
- from Data to Viz. (2020a, april 25). *About the Project* (from Data to Viz, Red.). <https://www.data-to-viz.com/#contact>
- from Data to Viz. (2020b, april 26). *Arc Diagram* (from Data to Viz, Red.). <https://www.data-to-viz.com/graph/arc.html>
- from Data to Viz. (2020c, april 26). *Chord Diagram* (from Data to Viz, Red.). <https://www.data-to-viz.com/graph/chord.html>
- from Data to Viz. (2020d, april 27). *Circular Packing* (from Data to Viz, Red.). <https://www.data-to-viz.com/graph/circularpacking.html>
- from Data to Viz. (2020e, april 27). *Dendrogram* (from Data to Viz, Red.). <https://www.data-to-viz.com/graph/dendrogram.html>
- from Data to Viz. (2020f, april 25). *Explore* (from Data to Viz, Red.). <https://www.data-to-viz.com/#explore>
- from Data to Viz. (2020g, april 26). *Heatmap* (from Data to Viz, Red.). <https://www.data-to-viz.com/graph/heatmap.html>
- from Data to Viz. (2020h, april 26). *Network Diagram* (from Data to Viz, Red.). <https://www.data-to-viz.com/graph/network.html>
- from Data to Viz. (2020i, april 28). *Sankey Diagram* (from Data to Viz, Red.). <https://www.data-to-viz.com/graph/sankey.html>
- from Data to Viz. (2020j, april 28). *Sunburst* (from Data to Viz, Red.). <https://www.data-to-viz.com/graph/sunburst.html>
- from Data to Viz. (2020k, april 27). *Treemap* (from Data to Viz, Red.). <https://www.data-to-viz.com/graph/treemap.html>
- Gomez Graphics Vector Conversions. (2020, april 2). *Raster vs Vector* (Gomez Graphics Vector Conversions, Red.). https://vector-conversions.com/vectorizing/raster_vs_vector.html
- Google. (2020a, maart 27). *Client libraries overview* (Google, Red.). <https://cloud.google.com/dialogflow/docs/reference/libraries/overview>
- Google. (2020b, maart 27). *Integrations* (Google, Red.). <https://cloud.google.com/dialogflow/docs/integrations/>
- Google. (2020c, maart 3). *Languages* (Google, Red.). <https://cloud.google.com/dialogflow/docs/reference/language>
- GoogleCloudPlatform. (2020, januari 6). *Dialogflow Integration* (GoogleCloudPlatform, Red.). <https://github.com/GoogleCloudPlatform/dialogflow-integrations>
- IBM. (2020a, maart 27). *Adding integrations* (IBM, Red.). <https://cloud.ibm.com/docs/services/assistant?topic=assistant-deploy-integration-add>

- IBM. (2020b, maart 20). *Creating entities* (IBM, Red.). <https://cloud.ibm.com/docs/services/assistant?topic=assistant-entities>
- IBM. (2020c, maart 20). *Defining intents* (IBM, Red.). <https://cloud.ibm.com/docs/services/assistant?topic=assistant-intents>
- IBM. (2020d, maart 23). *Dialog overview* (IBM, Red.). <https://cloud.ibm.com/docs/services/assistant?topic=assistant-dialog-overview>
- IBM. (2020e, maart 13). *Gathering information with slots* (IBM, Red.). <https://cloud.ibm.com/docs/services/assistant?topic=assistant-dialog-slots>
- IBM. (2020f, maart 28). *Making a programmatic call from dialog* (IBM, Red.). <https://cloud.ibm.com/docs/assistant?topic=assistant-dialog-webhooks>
- IBM. (2020g, april 24). *Modifying a dialog using the API* (IBM, Red.). <https://cloud.ibm.com/docs/assistant-icp?topic=assistant-private-dialog-api>
- IBM. (2020h, maart 28). *Supported languages* (IBM, Red.). <https://cloud.ibm.com/docs/assistant?topic=assistant-language-support>
- IBM. (2020i, maart 28). *This is a world with Watson* (IBM, Red.). <https://www.ibm.com/watson/ai-stories/>
- IBM. (2020j, maart 28). *Watson Assistant* (IBM, Red.). <https://www.ibm.com/cloud/watson-assistant/>
- IBM. (2020k, januari 1). *Watson Assistant Documentation* (IBM, Red.).
- IBM. (2020l, maart 28). *Watson SDKs* (IBM, Red.). <https://cloud.ibm.com/docs/assistant?topic=watson-using-sdks>
- IBM. (2020m, maart 28). *What sets Watson Assistant apart* (IBM, Red.). <https://www.ibm.com/cloud/watson-assistant/features/>
- Joshi, N. (2018, december 23). *Yes, Chatbots And Virtual Assistants Are Different!* (N. Joshi, Red.). <https://chatbotslife.com/chatbots-vs-virtual-assistants-what-is-the-difference-f23287e32165>
- Keim, D. (2002). Information Visualization and Visual Data Mining. *Volume 8*.
- Kent, C. (2019, juni 11). *Evidence Summary: Artificial Intelligence in education* (C. Kent, Red.). <https://eetn.eu/knowledge/detail/Evidence-Summary--Artificial-Intelligence-in-education>
- Khan, A. R. (2017, mei 19). *Integrate Your Amazon Lex Bot with Any Messaging Service* (A. R. Khan, Red.). <https://aws.amazon.com/blogs/machine-learning/integrate-your-amazon-lex-bot-with-any-messaging-service/>
- kore.ai. (2019, mei 2). *How Conversational Search, Driven by Chatbots, is Set to Bolster Customer Engagement* (kore.ai, Red.). <https://chatbotslife.com/are-conversational-chatbots-the-next-step-to-conversational-search-for-every-website-2d4ecba2e52b>
- Krzywinski, M. (2011, december 9). *Hive Plots, Rational Network Visualization - Farewell To Hairballs* (M. Krzywinski, Red.). <http://egweb.bcgsc.ca/>
- Kumar, C. (2018, september 25). *NLP vs NLU vs NLG (Know what you are trying to achieve) NLP engine (Part-1)* (C. Kumar, Red.). <https://towardsdatascience.com/nlp-vs-nlu-vs-nlg-know-what-you-are-trying-to-achieve-nlp-engine-part-1-1487a2c8b696>
- Lexico. (2020a, april 18). *metadata* (Lexico, Red.). <https://www.lexico.com/en/definition/metadata>
- Lexico. (2020b, april 2). *vector* (Lexico, Red.). <https://www.lexico.com/en/definition/vector>

- Lievens, S. (Red.). (2019, september 1). *Artificiële Intelligentie*.
- Lola.com. (2016, oktober 5). *NLP vs. NLU: What's the Difference?* (Lola.com, Red.).
<https://medium.com/@lola.com/nlp-vs-nlu-whats-the-difference-d91c06780992>
- McGrath, C. (2017, augustus 1). *Chatbot Vocabulary: 10 Chatbot Terms You Need to Know* (C. Magazine, Red.). <https://chatbotsmagazine.com/chatbot-vocabulary-10-chatbot-terms-you-need-to-know-3911b1ef31b4>
- Microsoft. (2019a, november 15). *About Azure Bot Service* (Microsoft, Red.). <https://docs.microsoft.com/nl-nl/azure/bot-service/bot-service-overview-introduction?view=azure-bot-service-4.0>
- Microsoft. (2019b, december 3). *Categorized activities by channel* (Microsoft, Red.).
<https://docs.microsoft.com/nl-nl/azure/bot-service/bot-service-channels-reference?view=azure-bot-service-4.0>
- Microsoft. (2019c, september 12). *Language and region support for LUIS* (Microsoft, Red.). <https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-language-support#languages-supported>
- Microsoft. (2020a, maart 26). *Bot Framework SDK* (Microsoft, Red.). <https://github.com/microsoft/botframework-sdk>
- Microsoft. (2020b, maart 29). *Gebouwd met Azure Bot Service* (Microsoft, Red.). <https://azure.microsoft.com/nl-nl/services/bot-service/#customer-stories>
- MongoDB. (2019, augustus 1). *MongoDB Architecture Guide: Overview* (MongoDB, Red.).
- Nagel, H. (2006). Scientific Visualization versus Information Visualization. NA.
- of Mechanical Engineers, A. S. (1947, mei 1). *Operation and flow process charts*.
- Olympus Press. (2013, december 6). *Vector & Raster Graphics in Offset Printing* (Olympus Press, Red.). <https://olyppress.com/vector-vs-raster-graphics-in-printing/>
- Otter, D., Medina, J. & Kalita, J. (2019, juli 1). *A Survey of the Usages of Deep Learning for Natural Language Processing*.
- Rasa. (2019, december 2). *NLP vs. NLU: What's the Difference and Why Does it Matter?* (Rasa, Red.). <https://blog.rasa.com/nlp-vs-nlu-whats-the-difference/>
- Rasa. (2020a, maart 29). *Build contextual assistants that really help customers* (Rasa, Red.). <https://rasa.com/>
- Rasa. (2020b, maart 29). *Customer Stories* (Rasa, Red.). <https://rasa.com/customers/>
- Rasa. (2020c, maart 29). *Language Support* (Rasa, Red.). <https://legacy-docs.rasa.com/docs/nlu/0.15.1/languages/>
- Rasa. (2020d, maart 29). *Messaging and Voice Channels* (Rasa, Red.). <https://rasa.com/docs/rasa/user-guide/messaging-and-voice-channels/>
- Rasa. (2020e, maart 30). *Rasa is essential in creating great AI assistants* (Rasa, Red.). <https://rasa.com/product/why-rasa/>
- Rasa. (2020f, maart 29). *Rasa offers flexible plans, from open source to enterprise subscriptions* (Rasa, Red.). <https://rasa.com/product/pricing/>
- Rasa. (2020g, maart 30). *Rasa SDK* (Rasa, Red.). <https://rasa.com/docs/rasa/api/rasa-sdk/>
- Reiters, E. & Dale, R. (2000, januari 28). *Building Natural Language Generation Systems*.
- Rouse, M. (2006, februari 13). *vector graphics* (M. Rouse, Red.). <https://searchwindevelopment.techtarget.com/definition/vector-graphics>
- Rouse, M. (2009, mei 13). *open source* (M. Rouse, Red.). <https://whatis.techtarget.com/definition/open-source>

- Rouse, M. (2017, oktober 24). *Amazon Lex* (M. Rouse, Red.). <https://searchaws.techtarget.com/definition/Amazon-Lex>
- Rouse, M. (2018, juli 30). *IBM Watson Assistant* (M. Rouse, Red.). <https://whatis.techtarget.com/definition/IBM-Watson-Assistant>
- Rouse, M. (2019, december 1). *chatbot* (M. Rouse, Red.). <https://searchcustomerexperience.techtarget.com/definition/chatbot>
- Shneiderman, B. (1996). The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. *Proceedings*.
- Stuart, R. & Peter, N. (2016, april 1). *Artificial Intelligence: A Modern Approach*.
- Tergan, T., Sigmar-Olaf & Keller. (2005, juni 28). *Knowledge and Information Visualization: Searching for Synergies*.
- The Bot Framework Team. (2017, maart 28). *Using custom channel data in bot messages* (The Bot Framework Team, Red.). <https://blog.botframework.com/2017/03/28/custom-channel-data/>
- The Data Visualisation Catalogue. (2020a, mei 2). *Arc Diagram* (The Data Visualisation Catalogue, Red.). https://datavizcatalogue.com/methods/arc_diagram.html
- The Data Visualisation Catalogue. (2020b, april 27). *Circle Packing* (The Data Visualisation Catalogue, Red.). https://datavizcatalogue.com/methods/circle_packing.html
- The Data Visualisation Catalogue. (2020c, mei 1). *Network Diagram* (The Data Visualisation Catalogue, Red.). https://datavizcatalogue.com/methods/network_diagram.html
- The Data Visualisation Catalogue. (2020d, april 28). *Sunburst Diagram* (The Data Visualisation Catalogue, Red.). https://datavizcatalogue.com/methods/sunburst_diagram.html
- The Data Visualisation Catalogue. (2020e, mei 2). *Treemap* (The Data Visualisation Catalogue, Red.). <https://datavizcatalogue.com/methods/treemap.html>
- the TOM agency. (2020, april 2). *Spreadsheets rarely tell a compelling story* (the TOM agency, Red.). <http://thetomagency.com/work/data-viz/>
- Vectr. (2020, april 2). *What Are Vector Graphics?* (Vectr, Red.). <https://vectr.com/tutorials/what-are-vector-graphics/>
- Zoovu. (2019, januari 1). Why your search experience is losing you money and what to do about it. (Zoovu, Red.).
- Zoss, A. (2019, september 19). *Data Visualization: About Data Visualization* (A. Zoss, Red.). <https://guides.library.duke.edu/c.php?g=289678&p=1930713>