1. Inleiding

De veroudering van de bevolking in de Vlaamse steden en gemeenten zet zich in de komende decennia verder (StatistiekVlaanderen, 2018). Volgens de voorspellingen zou tegen 2033 25% van de bevolking een 65-plusser zijn.

Tegelijk is het woord 'waardigheid' actueler dan ooit. Na de schrijnende omstandigheden van de Tweede Wereldoorlog stond dat woord centraal bij het opstellen van het Verdrag van de Verenigde Naties (1945), de Universele Verklaring van de Rechten van de mens (1948) en de grondrechten van de Europese Unie (2000). Die basiswaarde vinden we ook terug bij het Europese en Belgische zorgbeleid. Ouderen mogen niet gediscrimineerd worden op vlak van leeftijd. Tevens mogen ze ook niet op een kinderlijke, betuttelende of onvriendelijke wijze aangesproken worden en moeten ze met respect bejegend worden (Campens, 2021).

Hoe meer ouderen er in de samenleving zijn, hoe groter het zorgaanbod dat de samenleving moet organiseren voor deze leeftijdscategorie. Die zorgverleners, maar evengoed familie, weten niet altijd goed hoe ze moeten omgaan met ouderen. Wanneer een jonger persoon op een andere manier spreekt tegen een senior dan tegen een leeftijdsgenoot, spreken we over *elderspeak*. K. N. Williams (2011) omschrijft *elderspeak* als volgt: "Elderspeak is a common intergenerational speech style used by younger persons in communication with older adults in a variety of community and health care settings. Based on negative stereotypes of older adults as less competent communicators, younger speakers (in this case nursing home staff) modify their communication with nursing home residents by simplifying the vocabulary and grammar and by adding clarifications such as repetitions and altered prosody." Kortom, jongere mensen passen hun communicatie aan, door hun woordenschat te vereenvoudigen en verduidelijkingen toe te voegen zoals herhalingen, evenals hun prosodie (de fonologie, het ritme, de klemtoon en de intonatie van de

stem). Om *elderspeak* te bestrijden, gaven Wick en Zanni (2007) een paar tips mee in hun onderzoek. Enkele van die tips gingen als volgt: spreek mensen aan zoals ze wensen aangesproken te worden, vraag om ze aan te spreken met de voornaam, vermijd troetelnamen, wees bewust van non-verbaal gedrag, verhoog uw stemvolume enkel wanneer uw gesprekspartner hardhorig is, herhaal alleen uw zin als uw gesprekspartner het niet begrepen heeft, vermijd korte, langzame en makkelijke zinnen, vermijd verkleinwoorden en hanteer beleefd taalgebruik.

Naast *elderspeak* vormt het fenomeen *nursery tone* een extra uitdaging. Dit verwijst naar de situatie waarbij iemand de toonhoogte aan het einde van de zin standaard verhoogt zoals bij communicatie met jonge kinderen.

Beeckman (2021) en Standaert (2021) werkten vorig jaar in hun bachelorproef al aan de opzet van dit onderwerp en dit onderzoek zal verder werken op hun gelegde basis. Sommige stukken programmacode van hen zullen gebruikt worden om zo een beter model op te stellen. Zij haalden zelf ook verbeterpunten aan en moeilijkheden die, hopelijk, op te lossen zijn. Wat het verschil zal zijn tussen hun eindwerken en dit eindwerk wordt toegelicht in A.2.

De nog steeds relevante onderzoeksvraag van dit onderwerp is: “Kan *elderspeak* gedetecteerd worden door Artificiële Intelligentie en is het nuttig om AI toe te passen in de praktijk?”. Een bijkomende onderzoeksvraag is: “Kan *nursery tone* gedetecteerd worden door Artificiële Intelligentie?”. De laatste onderzoeksvraag moet een antwoord bieden op: “Kan de applicatie nuttig zijn voor zorgkundigen?”

1.1 Probleemstelling

Dit onderzoek heeft als doel een meerwaarde te betekenen voor de oudere mensen in woonzorgcentra en ziekenhuizen, maar ook nog de ouderen die zelfstandig thuis wonen. Ze vinden het namelijk helemaal niet aangenaam om aangesproken te worden als kleine kinderen. Deze applicatie zal *elderspeak* herkennen en aangeven welke elementen van *elderspeak* aanwezig zijn. Het zijn vooral de verpleegkundigen, artsen en ander zorgpersoneel, maar ook familieleden die zich bewust moeten zijn van de manier waarop ze tegen ouderen praten.

1.2 Onderzoeksvraag

De onderzoeksvraag die bij dit eindwerk hoort, is: “Kan *elderspeak* gedetecteerd worden door Artificiële Intelligentie en kan dit toegepast worden in de praktijk?”. Een bijkomende onderzoeksvraag is: “Kan *nursery tone* gedetecteerd worden door Artificiële Intelligentie?”. Maar enkel dit beantwoorden, is uiteraard niet genoeg. Aan de hand van de volgende deelvragen kan er een duidelijker en uitgebreider antwoord geformuleerd worden:

- Welk type Artificiële Intelligentie past het beste bij deze opstelling?
- Welk type model van *machine learning* of *deep learning* werkt het beste per eigenschap?
- Kan je achtergrondlawaai wegfilteren en hoe precies?
- Zal spraakherkenning lukken met de gratis beschikbare softwarebibliotheken?
- Hoe zet je een "Flask" server op waar je *webrequests* naar stuurt? En hoe verbind je daar een model mee?
- Is het nuttig voor zorgkundigen om deze applicatie in gebruik te nemen?

1.3 Onderzoeksdoelstelling

Deze bachelorproef heeft als doel om een basiswebsite aan te bieden die dient als *PoC* of *proof-of-concept*. Die basisapplicatie vraagt eerst om te praten zoals je zou doen tegen je vrienden. Nadien wordt er gevraagd om te praten zoals je zou doen bij slechthorende ouderen. Ter ondersteuning zal er een foto getoond worden van iemand uit een woonzorgcentrum. De applicatie analyseert dan de audiosamples en geeft aan welke kenmerken er aanwezig waren van *elderspeak*. Die kenmerken en de reden waarom het model oordeelt dat die aanwezig zijn, zijn vergaard in het literatuuronderzoek.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

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

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

In Hoofdstuk 4 worden de resultaten beschreven van deze bachelorproef zoals de verzamelde data, de resultaten na het testen, hoe goed een rollenspel werkt en of er aan de *requirements* voldaan zijn.

In Hoofdstuk 5 wordt er beschreven hoe de programmacode gedeeld kan worden, hoe de applicatie gehost kan worden en hoe men in het vervolg beter kan testen.

In Hoofdstuk 6 ten slotte, 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

"Kan *elderspeak* gedetecteerd worden door Artificiële Intelligentie en kan dit toegepast worden in de praktijk?", is de centrale onderzoeksvraag. Om te kunnen staven of dit wel degelijk mogelijk is, moeten twee begrippen uitgelegd en begrepen worden. Er zal dus eerst beschreven worden wat *elderspeak* precies is, hoe het ontstaat, wat de schadelijke effecten zijn, maar ook wat de eigenschappen zijn en hoe kan met het voorkomen? Ten tweede moeten we ook begrijpen wat Artificiële Intelligentie is, daarbij alle verschillende types, vormen en gradaties ervan.

2.1 Elderspeak

2.1.1 Definitie van elderspeak

Het begrip *elderspeak*, ook *secondary babytalk* genoemd, kent verschillende definities. Kemper e.a. (1998) omschrijft het begrip als volgt: "Elderspeak is a simplified speech register with exaggerated pitch and intonation, simplified grammar, limited vocabulary and slow rate of delivery."

Daarnaast beschrijft K. N. Williams (2011) het begrip als volgt: "Elderspeak is a common intergenerational speech style used by younger persons in communication with older adults in a variety of community and health care settings. Based on negative stereotypes of older adults as less competent communicators, younger speakers (in this case nursing home staff) modify their communication with nursing home residents by simplifying the vocabulary and grammar and by adding clarifications such as repetitions and altered prosody." Het omvat dus de stereotiepe communicatie tussen ouderen en jongeren.

2.1.2 Ontstaan van elderspeak

Er zijn verschillende communicatietheorieën die het ontstaan van *elderspeak* kunnen verklaren.

De Communicatie Accommodatie Theorie (CAT) beschrijft hoe mensen hun gespreksstijl aan hun gesprekspartner aanpassen en stelt dat deze aanpassingen een manier zijn om waarden en attitudes te uiten, aldus Campens (2021). Dat verschil in communiceren word convergeren genoemd. Dit is wanneer een gesprekspartner in de communicatie tracht af te stemmen op de communicatiestijl van de andere, waardoor twee gesprekspartners op de zelfde golflengte komen te zitten. Dit brengt mee dat de gesprekspartner beter luistert naar de spreker en de spreker voor de gesprekspartner voorspelbaarder wordt.

Bij *elderspeak* convergeert die stijl echter in een extreme vorm. De gesprekspartner laat zich te veel leiden door het stereotiepe beeld van ouderen. Het CAT-model stelt dat de jongere zich in zo'n geval niet aanpast aan de oudere gesprekspartner, maar wel aan dat stereotiepe beeld. Hierdoor zal *elderspeak* versterkt worden tijdens een gesprek (Hehman & Bugental, 2015).

De tweede communicatietheorie, namelijk het Communication Predicament of Ageing Model (CPA), beschrijft dat het stereotiepe beeld gestimuleerd kan worden. Dit wordt aangemoedigd door de omgeving, zoals een woon-zorgcentrum, en kenmerken, zoals een hoorapparaat of wandelstok (Hehman & Bugental, 2015).

In tegenstelling tot de twee vorige communicatiemodellen, probeert het Communicatie Enhancement of Ageing model (CEA) de vicieuze cirkel van het CPA-model te doorbreken. Campens (2021) beschrijft dit als volgt: "Het model stelt voor om de gespreksstijl op de individuele (oudere) gesprekspartner af te stemmen en een person-centered-approach te hanteren, eerder dan een category-based-approach (zoals bij het CPA-model het geval is).". Dit kan bekomen worden door systematische beoordeling van de individuele kenmerken van de gesprekspartner bij het begin van een conversatie. Wanneer er aanpassingen nodig zijn, moet er opnieuw een beoordeling plaatsvinden. Het model stelt dat er op die manier positieve resultaten kunnen bekomen worden voor beide gesprekspartners zoals empowerment, toegenomen competenties, tevredenheid, effectieve communicatie etc.˜(**\egroup**)Hehman2015.

2.1.3 Gevolgen van elderspeak

Zorgpersoneel probeert ouderen op hun gemak te stellen, vriendelijk over te komen, de boodschap van een gesprek op een duidelijke en effectieve wijze over te brengen en een betere samenwerking te verkrijgen. Hierdoor past men (on)bewust *elderspeak* toe, maar met de beste bedoelingen (Grimme e.a., 2015).

Door *elderspeak* toe te passen komen er negatieve gevolgen tevoorschijn bij ouderen. Zo krijgen ze het gevoel dat ze hulpeloos en afhankelijk zijn. Het bedreigt hun zelfbeeld, zelfrespect en het welzijn. Bovendien kan het depressieve gevoelens opwekken die zowel

fysieke als cognitieve achteruitgang in de hand kunnen werken. Hierdoor zullen sommige ouderen sociale interactie vermijden en mogelijk sociaal geïsoleerd geraken (K. Williams & Herman, 2011).

Het is dus belangrijk dat elderspeak vermeden wordt, want het ervaren van sociale betrokkenheid gaat gepaard met een toename van de tevredenheid over het leven (K. Williams & Herman, 2011).

2.1.4 Kenmerken van elderspeak

De kenmerken komen overeen met de communicatiestijl die men hanteert wanneer men tegen (afhankelijke) kinderen praat. Vandaar dat *elderspeak* ook wel als *secondary baby talk* wordt benoemd. Campens (2021) somt de kenmerken als volgt op:

- Langzaam spreken
- Verhoogde toonhoogte
- Verhoogd stemvolume
- Overdreven intonatie
- Vereenvoudigd woordgebruik, gebruik van verkleinwoorden en/of ongepaste bijnamen of troetelnamen
- Verminderde grammaticale complexiteit (bv. voornamelijk enkelvoudige zinnen)
- Gebruik van collectieve voornaamwoorden (bijvoorbeeld “we” in plaats van “jij”)
- Veelvuldig gebruik van (bevestigende) tussenwerpsels (zoals “hé” of “voilà”)
- Gewijzigd non-verbaal gedrag (bv. langdurig oogcontact, extra gebaren, te dichtbij komen)
- Veelvuldige verduidelijking en herhaling

Ten slotte is het belangrijk om te beseffen dat bij *elderspeak* de inhoud van de boodschap, die de zorgverlener wil overbrengen, niet wijzigt. Wel verandert de wijze waarop de boodschap wordt overgebracht door het gebruik van een infantiliserende communicatiestijl, aldus Campens (2021).

2.1.5 Tips om elderspeak te voorkomen?

De onderstaande tips zijn er enkelen die Wick en Zanni (2007) meegeven om *elderspeak* te voorkomen:

- Spreek personen aan met de naam waarmee ze willen aangesproken worden. Gebruik geen collectieve voornaamwoorden als die niet van toepassing zijn.
- Als een persoon een zorgverlener toelaat om hem of haar met zijn voornaam aan te spreken, ga er dan niet van uit dat deze toestemming voor alle zorgverleners geldt. De mate van intimiteit varieert, waardoor elke zorgverlener moet nagaan hoe hij of zij zijn of haar gesprekspartner mag aanspreken.
- Vermijd het gebruik van troetelnamen en overmatige intieme liefkozingen, tenzij de gesprekspartner uitdrukkelijk aangeeft dat hij of zij zo wil aangesproken worden.

- Wees je bewust van je non-verbaal gedrag.
- Verhoog je stemvolume (in beperkte mate) enkel en alleen wanneer de gesprekspartner hardhorig is. Verhoging van je stemvolume betekent geen verhoging van je stemhoogte. Wees je ervan bewust dat niet elke oudere gesprekspartner hardhorig is.
- Herhaling en verminderde grammaticale complexiteit hebben een plaats als de gesprekspartner je niet begrepen heeft.
- Vermijd korte, langzaam en met overdreven intonatie uitgesproken zinnen.
- Vermijd het gebruik van verkleinwoorden, aangezien die het gevoel van afhankelijkheid kunnen versterken en denigrerend kunnen geïnterpreteerd worden.
- Vermijd overdreven directieve boodschappen en bied keuzevrijheid.
- Hanteer een beleefd taalgebruik en beleefde omgangsvormen (bv. op de kamerdeur kloppen alvorens de kamer binnen te gaan).

2.2 Artificiële Intelligentie

2.2.1 Definitie van AI?

Er zijn bijzonder veel beschrijvingen van AI, maar een combinatie van bronnen geeft het meest accurate beeld.

Volgens Oracle (2014) heeft AI (kunstmatige intelligentie) betrekking op systemen of machines die onze eigen intelligentie nabootsen om taken uit te voeren en die zichzelf tijdens dat proces kunnen verbeteren op basis van de vergaarde informatie.

Het Europees Parlement (2020) geeft de volgende definitie aan artificiële intelligentie (of kunstmatige intelligentie): "AI is de mogelijkheid van een machine om mensachtige vaardigheden te vertonen - zoals redeneren, leren, plannen en creativiteit. AI maakt het voor technische systemen mogelijk om hun omgeving waar te nemen, om om te gaan met deze waarnemingen en problemen op te lossen om een specifiek doel te bereiken. De computer ontvangt data - reeds voorbereid en verzameld via eigen sensoren, zoals een camera - verwerkt deze en reageert erop. AI-systemen zijn in staat om hun gedrag in zekere mate aan te passen door het effect van vorige acties te analyseren en autonoom te werken."

Samengevat is AI een technisch systeem dat opereert via complexe/grootschalige informatievergaring, maar waarbij het systeem vooral ook zichzelf aanpast op basis van die vergaarde informatie en op die manier menselijke intelligentie nabootst.

2.2.2 Vormen van AI

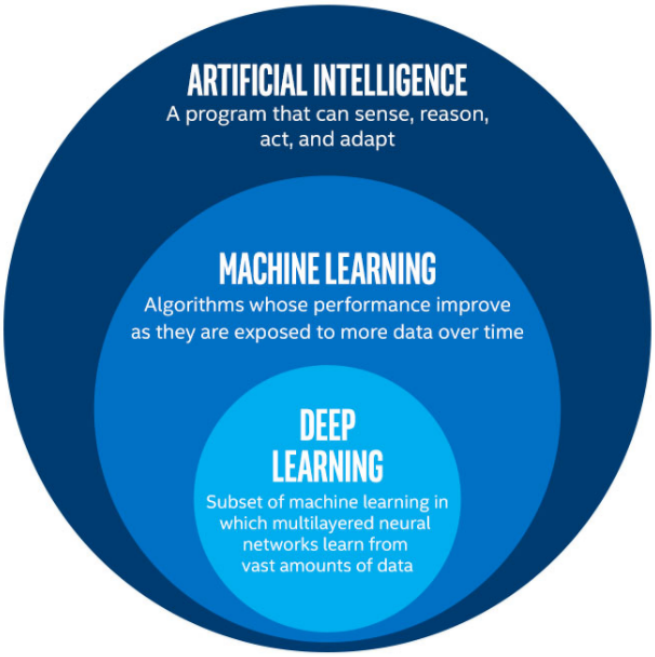
Artificiële Intelligentie is een verzamelnaam voor twee soorten algoritmes. Zo is er enerzijds *machine learning* en anderzijds *deep learning* (Kavlakoglu, 2020). Dit kan mooi geïllustreerd worden a.d.h.v. figuur 2.1.

2.2.3 Machine learning

Machine learning of machinaal leren is het deelgebied van kunstmatige intelligentie dat computers het vermogen geeft te leren zonder expliciet geprogrammeerd te zijn, aldus Lievens (2021). Machinaal leren heeft drie type, namelijk *supervised learning* of gesuperviseerd leren, *unsupervised learning* of leren zonder toezicht en *reinforcement learning* of leren door bekrachtiging.

Supervised learning

Een definitie over *supervised learning*, gegeven door Lievens (2021), gaat als volgt: "De taak van *supervised learning* is een hypothese op te bouwen op basis van een reeks gelabelde trainingsgegevens. Deze hypothese kan dan worden gebruikt om het label voor een (nieuwe) input te voorspellen. Wanneer het label een reëel getal is, spreekt men van



Figuur 2.1: Soorten AI in diagram (Bansal, 2019)

een regressieprobleem; wanneer het label beperkt is tot een (beperkt) aantal vooraf gedefinieerde klassen, wordt het probleem een classificatieprobleem genoemd." De visuele voorstelling van beide problemen is te vinden in figuur 2.2.

Een voorbeeld van het regressieprobleem is het voorspellen van huisprijzen. De input van het model is dan een reeks vectoren die de eigenschappen van het huis voorstellen zoals aantal slaapkamers, oppervlakte, bouwjaar etc.

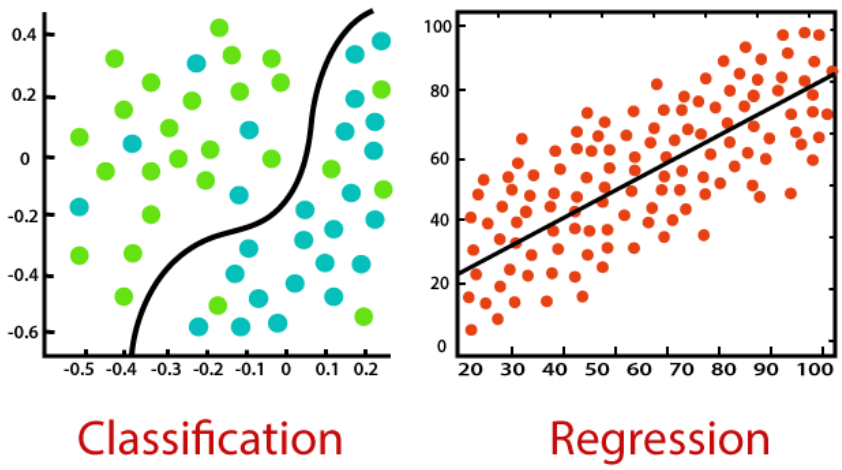
Voorbeelden van het classificatieprobleem zijn spamdetectie, nummerherkenning of diabetesdetectie. Elk hebben ze een beperkt aantal outputklassen. Bij spamdetectie zijn er twee vooraf gedefinieerde klassen: spam en geen spam. Bij nummerherkenning zijn er 10 klassen (0 t.e.m. 9) en bij diabetesdetectie kunnen er drie klassen aanwezig zijn (geen diabetes, pre-diabetes en diabetes).

Unsupervised learning

Lievens (2021) omschrijft *unsupervised learning* als de taak van het ontdekken van structuur in een ongelabelde gegevensreeks.

De meest prominente taak bij *unsupervised learning* is *clustering*, d.w.z. de ontdekking van coherente groepen. Andere mogelijke taken zijn anomaliedetectie en hoofdcomponentenanalyse (PCA).

Een voorbeeld van clustering is marktsegmentatie waarbij klanten worden opgedeeld in verschillende segmenten zoals trouwe klant, mogelijke vertrekkende klant, ontevreden



Figuur 2.2: Classification vs. Regression (JavaTpoint, 2021).

klant etc. Op basis van aankoopdata, tijdstippen en online activiteit kan men de klanten indelen in clusters. Die informatie kan dan weer gebruikt worden om mensen die behoren tot bepaalde clusters korting te geven of te contacteren i.v.m. hun ontevredenheid. Daarnaast zijn er algoritmen die fraude opsporen door het analyseren van abnormaal gedrag op een website. Dit is dan een voorbeeld van anomaliedetectie. Bij PCA wordt de data gereduceerd zodat de minder relevante data weggefilterd wordt uit de dataset.

Reinforcement learning

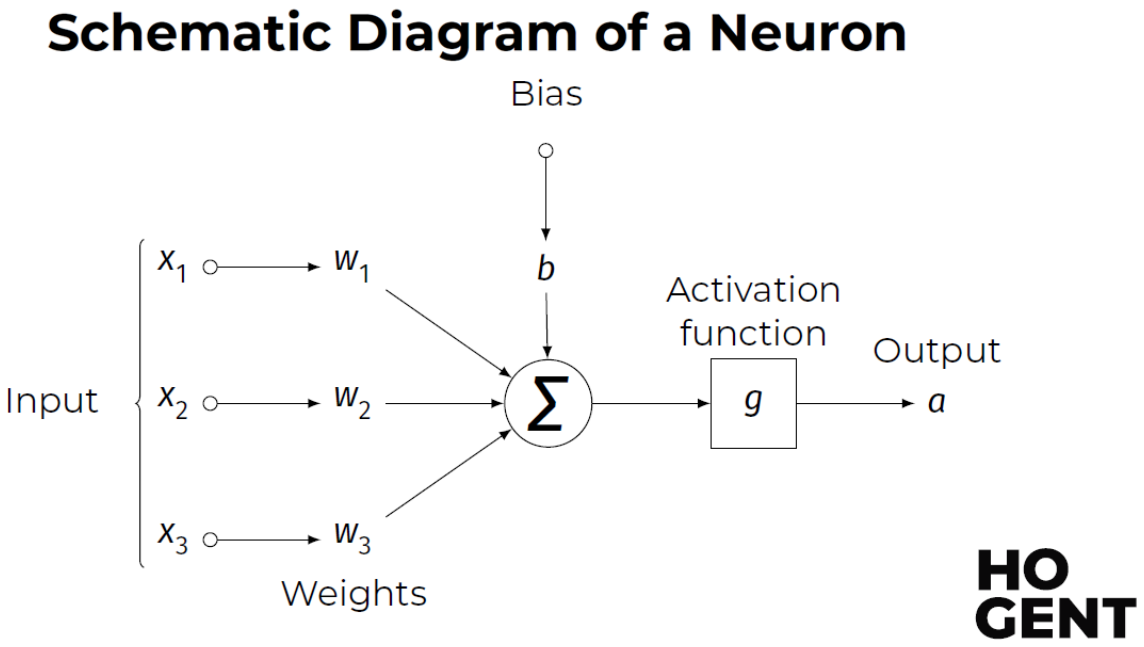
Lievens (2021) beschrijft *reinforcement learning* als een techniek die niet echt een dataset gebruikt, maar wel een identiteit die leeft in een (on)bekende wereld. Die identiteit krijgt dan beloningssignalen. De opdracht is dan om uit te zoeken welke regels leiden tot een grote beloning.

Het bekendste voorbeeld van *reinforcement learning* is dat van zelfrijdende wagens. Hierbij moet het model zelf aanleren in welke mate het moet remmen, gas geven en sturen. Dit is een lang en iteratief proces waarbij het model zichzelf de hele tijd bijstuurt en leert van wat het net ervaren heeft.

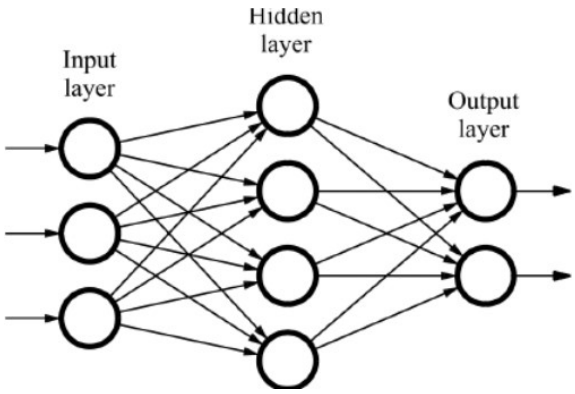
2.2.4 Deep Learning

Deep learning is het onderdeel van *machine learning* dat neurale netwerken bevat. “Een artificieel neuraal netwerk bestaat uit een groot aantal eenvoudige rekenende eenheden, units of neuronen die de volgende eigenschap hebben: ‘Als de (gewogen) input van een neuron groot is, zal het ’vuren’ en dit neuron zal een grote waarde op zijn axon zetten. Bovendien zijn deze eenheden verbonden door middel van gerichte links waarbij een reëel getal de sterkte van elke verbinding aangeeft.”, aldus Lievens (2021) in zijn cursus *Distributed Databases*.

De systematische voorstelling van een neuron is te vinden in figuur 2.3. Een volledig



Figuur 2.3: Systematische voorstelling van een Neuron (Lievens, 2021).



Figuur 2.4: Layers van een artificieel neuraal netwerk (Lievens, 2021).

neuraal netwerk zoals in figuur 2.4. De data wordt gevoed aan de *input layer*. Daarna komen een x-aantal *hidden layers* die dan de bewerkingen uitvoeren en als laatste stap heb je de *output layer*. Het aantal *nodes* of knopen in de *output layer* is gelijk aan het aantal klassen dat een probleem heeft. Als er een *deep learning*-model gemaakt wordt van het detecteren van spam, dan zou de *output layer* twee knopen moeten hebben.

2.3 AI-technieken die gebruikt worden om elderspeak te detecteren

2.3.1 NLP of natural language processing

Om te kunnen detecteren of de spreker verkleinwoorden, troetelnamen, collectieve voornaamwoorden of te veel tussenwerpsels gebruikt, moet het model, of eerder gezegd de Python-bibliotheek die het model bevat, weten of deze eigenschappen voorkomen. Hiervoor moeten we gebruik maken van *natural language processing*-bibliotheken of afgekort NLP.

Natural language processing of "natuurlijke taalverwerking" situeert zich in de tak van de kunstmatige intelligentie. NLP behoort deels tot *machine learning* en deels tot *deep learning* (Kleinings, 2022). De visuele voorstelling van die verhoudingen is te vinden in figuur 2.5.

Volgens Kleinings (2022) is die natuurlijke taalverwerking een technologie die gebruikt wordt om computers te helpen om natuurlijke menselijke taal te begrijpen en te interpreteren.

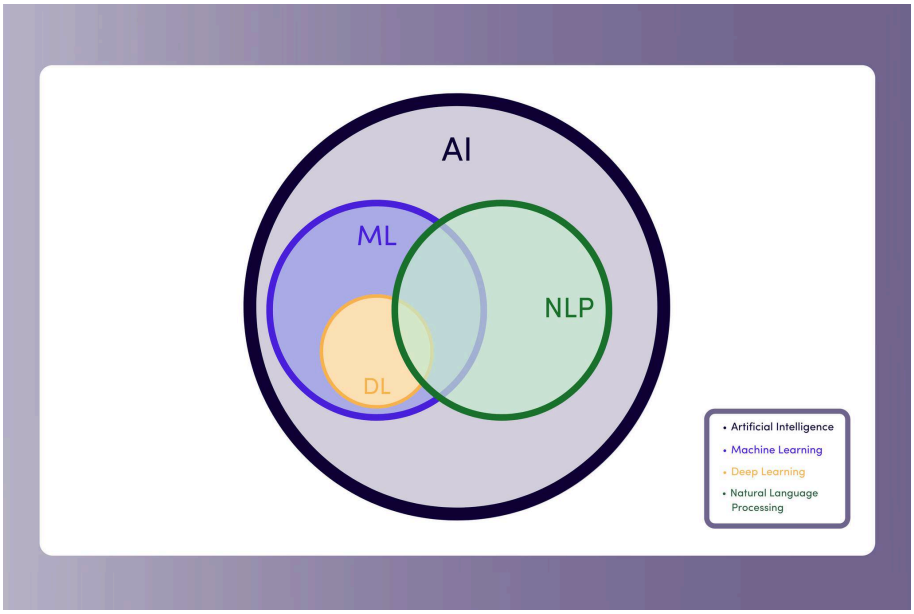
NLP combineert computationele linguïstiek (op regels gebaseerde modellering van menselijke taal) met statistische, *machine learning*- en *deep learning*-modellen. Samen stellen deze technologieën computers in staat menselijke taal in de vorm van tekst of spraakgegevens te verwerken en de volledige betekenis ervan te "begrijpen", met inbegrip van bedoeling en het sentiment van de spreker of schrijver. (IBM Cloud Education, 2021)

NLP stuurt computerprogramma's aan die tekst van de ene taal naar de andere kunnen vertalen, die reageren op gesproken opdrachten en die grote hoeveelheden tekst samenvatten, zelfs in realtime. Enkele voorbeelden van NLP-systemen in het dagelijkse leven zijn: spraakgestuurde GPS-systemen, digitale assistenten, spraak-naar-tekst dicteersoftware (o.a. in smartphones of Microsoft Word), chatbots voor de klantenservice en andere 'gemakken' voor de consument. Maar NLP speelt ook een steeds grotere rol in bedrijfsoplossingen die helpen bij het stroomlijnen van de bedrijfsvoering, het verhogen van de productiviteit van werknemers en het vereenvoudigen van bedrijfskritische bedrijfsprocessen, aldus IBM Cloud Education (2021).

Waarom is het analyseren van taal moeilijk?

Volgens Kleinings (2022) zijn er vele redenen waarom het verwerken van taal door een computer of AI-systeem nog steeds moeilijk is en zal blijven.

Eerst en vooral heb je alle linguïstische regels per taal. Zo zijn er meer dan 6500 talen die gesproken worden en die elke hun eigen regels hebben. Daarnaast is er het probleem van de uniformiteit. Om taal te kunnen verwerken moet die eerst omgezet worden in een systeem of formaat dat de computer kan begrijpen. Door middel van machinaal leren identificeert het model ongestructureerde taal, en zet deze om naar bruikbare, niet-dubbelzinnige



Figuur 2.5: Situering van NLP binnen het AI-veld (Kleinings, 2022).

informatie. De computer kan dit begrijpen en er verder mee rekenen. Dit onderdeel wordt de *pre-processing* genoemd.

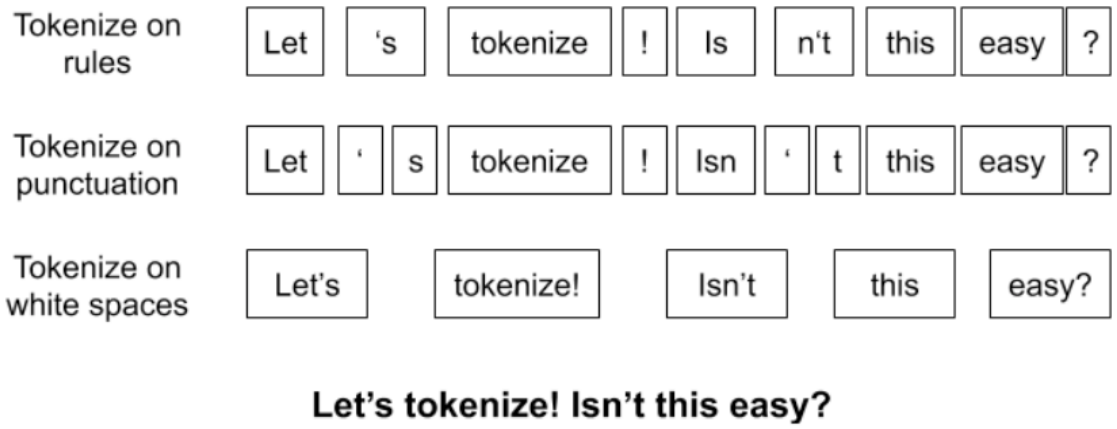
Bovendien speelt de context NLP ook vaak parten. *Natural Language Processing* werkt immers fundamenteel door de hiërarchie van linguïstische dictie tussen elk woord te begrijpen en deze om te zetten in een vorm die computers kunnen interpreteren. Onze talen zijn niet eenvoudig, zo hebben woorden meerdere betekenissen die alleen begrepen worden door het verschil in context. (Kleinings, 2022) Een voorbeeld hiervan is het woord "bank". In de ene context is het de financiële instelling en in een andere context is het de rustbank in het park.

Als laatste kan de toon van de stem ook nog verschillen. Mensen kunnen d.m.v. intonatie en toonhoogte sarcasme of ironie uiten, zodat de zin een andere betekenis krijgt. Het kost veel moeite om om dit correct te interpreteren.

Werking van NLP

NLP is niet één statische methode, maar een ketting van manipulaties van de tekst zodat er meer lagen informatie tevoorschijn komen. Dit wordt gerealiseerd met neurale netwerken waarbij elke node een bepaalde functie uitvoert. Er zijn vier grote stappen i.v.m. de taalverwerking namelijk morfologie, syntaxis, semantiek en pragmatiek, schrijft Kleinings (2022). Dit is te lezen is op een *low-code/no-code* platform, genaamd Levity, dat dus veel kennis heeft over AI. Daar kan men blokken slepen en gebruiken om een AI-model op te stellen.

Met behulp van morfologie kan er per woord een type geclassificeerd worden zoals een zelfstandig naamwoord, bijvoeglijk naamwoord, voornaamwoord, lidwoord enzovoort.



Figuur 2.6: Manieren om een zin om te delen volgens een *token* (Horan, 2020).

Denk hierbij aan het voorbeeld met de bank.

Om de syntaxis aan te leren zijn er twee manieren. Enerzijds kunnen er woorden gelabeld worden waarbij er aangeduid wordt dat woord A de stam is, woord B de 2e persoon enkelvoud, woord C het voltooid deelwoord etc. Anderzijds kan er een *unsupervised machine learning* model gemaakt worden waarbij de computer de regels afleidt uit verschillende gelabelde teksten, en die logica gebruikt om nieuwe, niet-gelabelde, teksten te begrijpen.

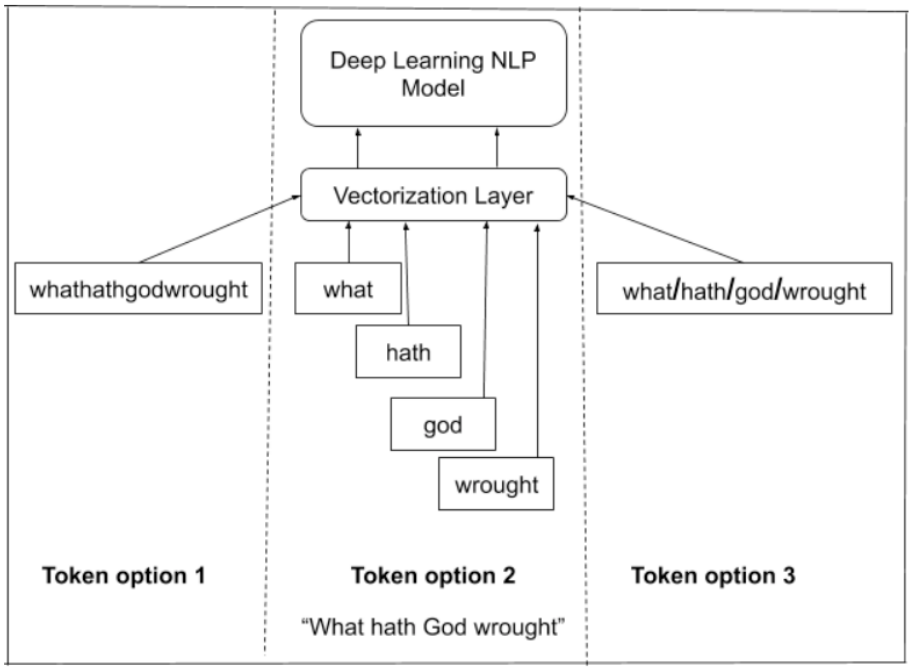
Daarnaast is het ook aan te raden dat stopwoorden verwijderd worden als het model de tekst wil begrijpen. Stopwoorden zoals "een, de, euh, inderdaad, echt, oké, eigenlijk, allez, etc." moeten verwijderd worden. Daarbij aansluitend zijn er twee andere methoden. Enerzijds lemmatiseren, waarbij werkwoorden naar de tegenwoordige tijd veranderd worden. Anderzijds is er stamming, waarbij er prefixen of affixen verwijderd worden. Een voorbeeld van conversie is dan: "wij dachten" naar "denk". Woorden worden daarnaast ook vervangen door meer gebruikte synoniemen zoals van "enorm" naar "groot".

Als laatste systeem of layer bestaat er *tokenization*. Dit is het proces waarbij het systeem de zin indeelt in verschillende eenheden waaruit informatie kan gehaald worden. Er kan een *token* gekozen worden op basis van regels, leestekens, spaties, maar dit geeft wel steeds een ander resultaat, wat geïllustreerd is in figuur 2.6. Hoe die *tokenization* zich verhoudt met die *layer* van het *deep learning*-NLP-model is te vinden in figuur 2.7.

2.3.2 Filteren van achtergrondlawaai

Volgens Jung e.a. (2021) helpen *convolutional neural networks* of CNN's bijzonder veel om achtergrondlawaai weg te filteren. Een CNN is een specifieke opstelling van een *deep learning*-model waarbij een data-array van twee of meer dimensies, zoals een afbeelding of geluidsfragment, wordt gestapeld door een veelvoud van tweedimensionale filters.

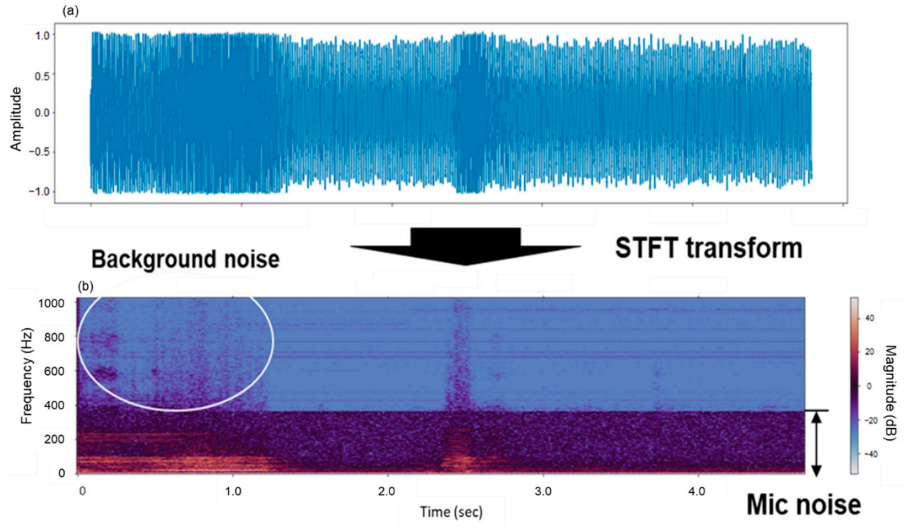
Om achtergrondlawaai weg te werken zijn AI-gebaseerde filters, zoals de *short-time Fourier transform* of STFT, ideaal. De STFT-filter verbetert in het algemeen de kwaliteit van



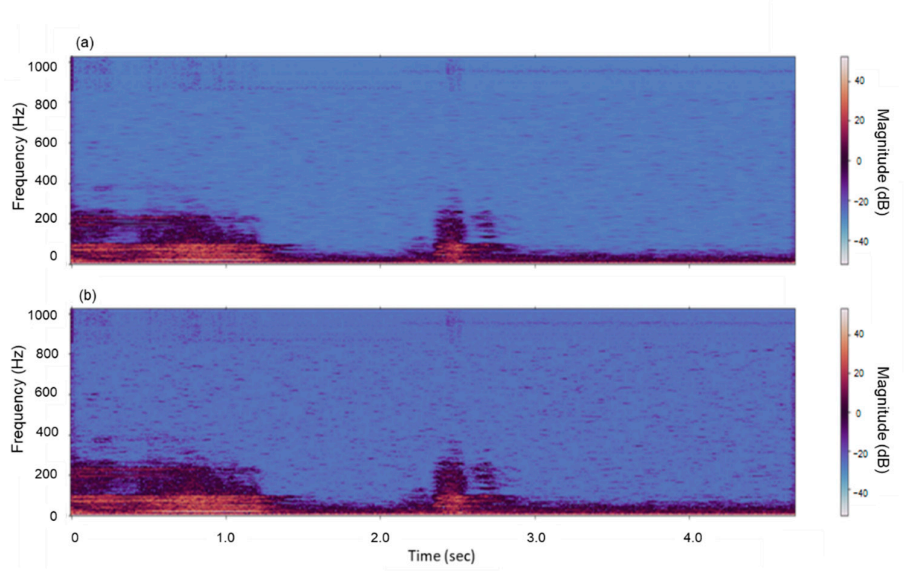
Figuur 2.7: Verhouding *deep learning*-NLP-model met de layers en de verschillende *tokens* (Horan, 2020).

het inkomende geluidssignaal door ruis uit het geluidssignaal te verwijderen. De STFT is een Fourier-gerelateerde transformatie die wordt gebruikt om de sinusoïdale frequentie- en fase-inhoud van plaatselijke delen van een signaal te bepalen terwijl dit in de tijd verandert, zie figuur 2.8. In de praktijk worden STFT's berekend door een geluidssignaal op te delen in kortere segmenten van gelijke lengte en de Fouriertransformatie afzonderlijk te berekenen voor elk korter segment. Aldus wordt het Fourier-spectrum voor elk korter segment onthuld. Gewoonlijk zet men dan de veranderende spectra uit in functie van de tijd; de grafiek staat bekend als een spectrogram of watervalgrafiek, zie figuur 2.9. (Jung e.a., 2021)

In de opstelling van het onderzoek van Jung e.a. (2021) gebruikt men de *librosa* en de *noisereduce* Python-bibliotheken. Deze twee bibliotheken zullen ook gebruikt worden in het praktische gedeelte van deze bachelorproef.



Figuur 2.8: Tweedimensionale bron van de spraakgegevens van de oestrusoproep bij runderen (a) en de omzetting naar korte-tijd Fourier transform (STFT) gebied (b) (Jung e.a., 2021).



Figuur 2.9: De originele audio met ruisonderdrukking (a) en spraakinformatie gecorrigeerd door ruismasker afvlakking (b) (Jung e.a., 2021).

3. Methodologie

Na het literatuuronderzoek is het doel van deze bachelorproef om een website te ontwikkelen waarmee *elderspeak* kan gedetecteerd worden via een detector. In het literatuurgedeelte werd ook theoretisch beschreven wat AI is en welke types er zijn, wat en hoe *natural language processing* werkt en hoe men achtergrondlawaai filtert.

Alle eigenschappen van *elderspeak* worden herkend a.d.h.v. Python-bibliotheken en niet op basis van een zelfgemaakt AI-model. Men moet immers het warm water nieuw opnieuw uitvinden. Zo stelde Beeckman (2021) het volgende: “Deze bachelorproef heeft geen meerwaarde kunnen aantonen voor het gebruik van een CNN. Wegens omstandigheden was het niet mogelijk te beschikken over een grote dataset wat leidt tot slechte voorspellingen van het model. Het model voorspelt een classificatie aan dezelfde accuraatheid dan dat gokken zou teweegbrengen. Dit maakt het huidige model onbruikbaar in de praktijk.”. De denkpiste om zelf een AI-model te maken, werd door deze stelling snel ontkracht. Ook de promotor Van Boven haalde aan dat er genoeg Python-bibliotheken ter beschikking zijn om de opdracht op die manier tot een goed einde te brengen.

Sommige methoden konden gekopieerd worden van de bijlagen van de studenten die hiervoor aan dit project gewerkt hebben, maar niet alle code stond beschreven in die bijlage. Daarnaast had Standaert (2021) geen GitHub-*repository*, waardoor de code niet snel kon hergebruikt worden. Ook zaten er af en toe kleine foutjes in de code waarbij het nodig was om die op te lossen. Om die reden zal er in Hoofdstuk 5 een deeltje aan bod komen over belang van het delen van code.

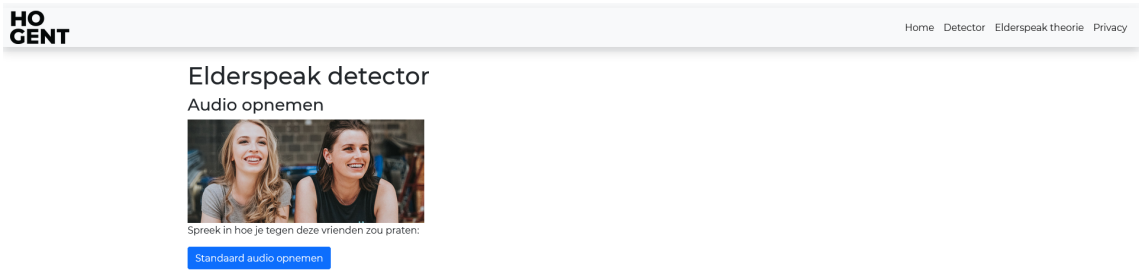
Figuur 3.1: Home pagina website

3.1 Berekeningen in back-end

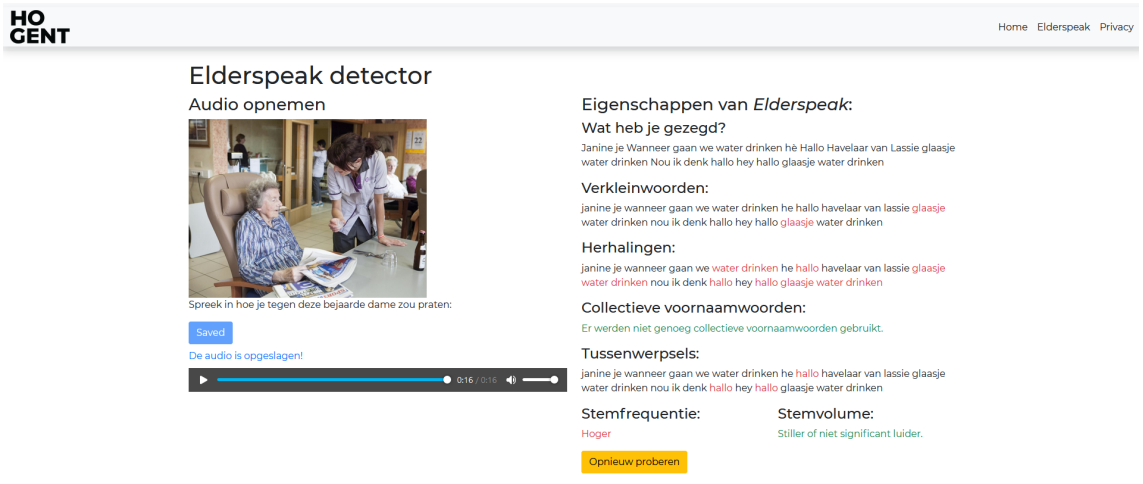
Deze voorbeeldapplicatie, geschreven in Python en gebruikmakend van het *micro-framework* Flask, is een webapplicatie die verschillende webpagina's bevat. De code van de volledige Flask-applicatie is te vinden in bijlage B.1. Naast de inleidende pagina, zie figuur 3.1, bevat deze ook een detector. Voor het begin zie figuur 3.3, en na het analyseren ziet de webpagina er als volgt uit, zie figuur 3.4. Er kan ook bestudeerd worden wat *secondary baby talk* is in de vorm van een oplijsting, zie figuur 3.2. Op die manier kan iedereen, maar specifiek studenten in de zorg, actief en passief leren wat *elderspeak* precies is. Enerzijds kunnen ze de eigenschappen leren herkennen door zelf actief stukjes spraak op te nemen. Die worden dan geanalyseerd zodat men kan zien welke eigenschappen er aanwezig waren. Anderzijds kunnen ze passief leren wat de eigenschappen zijn van *elderspeak* en hoe men dat kan voorkomen.

De applicatie bestaat uit twee onderdelen. Eerst wordt een afbeelding van twee jonge vrouwen getoond, dit is een foto van HOGENT voor copyrightrechten. Er wordt dan gevraagd om tegen hen te spreken en audio op te nemen. Dat fragment wordt dan geanalyseerd voor de eigenschappen van toonhoogte en stemvolume, wat gebruikt kan worden als vergelijkingsmateriaal voor het volgende fragment. Vervolgens wordt er een foto van een oudere vrouw getoond waarbij er gevraagd wordt om haar aan te spreken. Na dit tweede fragment worden de aanwezige *elderspeak*-eigenschappen weergegeven. Ook indien er geen kenmerken aanwezig waren, wordt dit getoond op de website.

Figuur 3.2: Eigenschappen *elderspeak* op de website.



Figuur 3.3: Detector bij het begin.



Figuur 3.4: Detector na het analyseren.

3.1.1 Speech recognition

De basis van de applicatie is het herkennen van de spraak die wordt opgenomen op de website. Uiteraard moet dit niet van nul gebouwd worden, maar kan er gebruikgemaakt worden van een bestaande Python-bibliotheek. Standaert (2021) bemerkte in zijn bachelorproef dat de Google Speech Recognition-API de beste optie is om te gebruiken, omdat deze het beste presteert. Hij vatte dit samen in zijn conclusie: “Uit verschillende onderzoeken of studies waar men verschillende ASR-systemen met elkaar vergeleken [sic] kwam de Google Speech-API er altijd als beste uit. En dit in alle aspecten.”. Daarnaast gebruikt de Google Speech-API ook *natural language processing* of NLP om beter te kunnen begrijpen wat er precies gezegd werd (Google Cloud, 2022). Hoe dit precies werkt, is na te lezen in het literatuuronderzoek, namelijk in Hoofdstuk 2.

Bij de spraakherkenning kwam er wel een groot probleem aan het licht. De Google *Speech Recognition*-API kan alleen overweg met *wav*- en *flac*-bestanden én kan maar een bepaalde periode, ongeveer 2 tot 3 minuten, gratis herkennen. Het probleem is dat de audio gecapteerd werd in een *mp3*-formaat. Het was dus noodzakelijk om eerst een conversie te maken van *mp3* naar *flac*, en om het audiobestand op te delen in deelbestandjes of *chuncks*, zodat er geen betalende versie voor nodig was. De gebruikte technologie hierbij was “ffmpeg”. Hierbij was het nodig dat de lengte van het bestand berekend werd, in hoeveel chunks het bestand opgedeeld moet worden a.d.h.v. de limiet die de gratis Google *Speech Recognition*-API toeliet. Hoe de conversie heel precies gebeurt, is gedetailleerd te vinden in bijlage B.2.

Het gebruiken van die API is relatief gemakkelijk. Een extra optie is dan ook ter beschikking om achtergrondlawaai een beetje weg te filteren. Door de optie ‘ajust_for_ambient_noice(source)’ aan te zetten, zal de bibliotheek zich aanpassen aan de geluidsbron om de klanken beter op te nemen alvorens die worden doorgestuurd naar de spraakherkenning-API. Deze methode gebruikt de techniek die beschreven is in het literatuuronderzoek over achtergrondlawaai.

Eens dit ingesteld is, worden de audiobestanden meegegeven en krijgt het systeem de tekst terug waarvan het AI-model van Google de tekst herkend heeft. Uiteraard werkt dit niet feilloos, maar het is wel redelijk goed voor een gratis versie. Dit vormt de basis voor de volgende methoden. Die zijn te vinden in de tussentitels hieronder:

3.1.2 Verkleinwoorden

Om verkleinwoorden te herkennen werd de methode van Standaert (2021) gehanteerd. Daarbij wordt er gekeken of woorden langer zijn dan 3 letters en ze niet voorkomen in de lijst die geen verkleinwoorden zijn. Enkele voorbeelden die wel eindigen op “-je”, “-ke”, “-kes” of “-jes”, maar geen verkleinwoorden zijn, zijn: poffertje, meisje, koopje, etentje, dutje, toetje, mannelijke, vrouwelijke etc. Die herkende verkleinwoorden worden dan bijgehouden in een lijst. Op het einde wordt ofwel bepaald dat er geen verkleinwoorden aanwezig waren, of worden alle verkleinwoorden in de tekst in het rood aangeduid. De code voor deze methode kan gevonden worden in bijlage B.3.

3.1.3 Herhalingen

Om herhalingen te herkennen werd er eveneens gebruik gemaakt van de methode die Standaert (2021) beschreven heeft. Daarbij worden de voorbije 25 woorden bijgehouden waarin eventuele herhalingen bewaard worden in een lijst. Die zullen later gebruikt worden om een mooie weergave te maken van de aanwezige herhalingen. De code om herhalingen te detecteren is te vinden in bijlage B.4.

3.1.4 Collectieve voornaamwoorden

Een voorbeeld van een collectief voornaamwoord is het gebruiken van “we” / “wij”.Volgende zinnen verduidelijken dit voorbeeld: “Gaan we onze patatjes opeten?”, “Kunnen we alleen naar de wc?”, “Awel, wat zijn we aan het doen?”.

Er wordt bijgehouden hoeveel keer er collectieve voornaamwoorden gebruikt worden in de tekst. Als een woord meer dan één keer voorkomt, dan zal de applicatie dit weergeven. Dit is zo ingesteld omdat het niet mag worden weergegeven wanneer er iemand eenmalig het woord “we” gebruikt. Wanneer er geen of minder dan twee collectieve voornaamwoorden gebruikt worden, zal de applicatie zeggen dat er geen of niet genoeg aanwezig waren om als *elderspeak* vast te stellen.

Natuurlijk duiden twee of meer collectieve voornaamwoorden niet direct op *elderspeak*, maar het geeft wel al een richting. De persoon in kwestie moet natuurlijk de theorie over *elderspeak* kennen en moet daarna ook kritisch zijn over het resultaat, onder meer met behulp van een zelfreflectie.

3.1.5 Tussenwerpsels

Het veelvuldig gebruik van tussenwerpsels is een eigenschap van *elderspeak* en ook dit wordt herkend. Enkele voorbeelden van tussenwerpsels die herkend worden zijn: “oh”, “oeps”, “helaas”, “hallo”, “hey”, “voila” etc. De Google *Speech Recognition*-API geeft soms verschillende varianten op het woord “hey”. Zo worden de volgende vormen soms gegeven: “hé”, “hè”, “he”, “hey”. Om te voorkomen dat dit foute resultaten oplevert, worden al deze varianten herleid naar “hey”.

De werkwijze om dit te detecteren is ongeveer dezelfde als de methode voor de collectieve voornaamwoorden, bijgevoegd als bijlage B.6.

3.1.6 Toonhoogte

De toonhoogte is een bijzonder belangrijke eigenschap van *elderspeak*. Deze eigenschap is ook volledig onafhankelijk van de gesproken tekst die herkend werd. Wanneer een persoon merkbaar hoger praat, zal de andere persoon direct voelen dat hij/zij behandeld wordt als een kind. Het is dan ook zeer belangrijk dat deze functie goed werkt, opdat de

gebruikers onmiddellijk attent gemaakt worden op het feit dat ze (on)bewust hoger praten.

Deze methode werd al opgesteld door Standaert (2021) in zijn eindwerk. Hij berekende de gemiddelde toonhoogte, uitgedrukt in Hz, van het gegeven audiobestand. Deze applicatie zorgde voor uitbreiding namelijk door het berekenen of de toonhoogte hoger ligt bij de casus met de oudere vrouw dan in de casus met de twee jongere vrouwen. Hoe dit gerealiseerd werd in de code is te vinden in bijlage B.7.

3.1.7 Stemvolume

Ten slotte analyseert de applicatie of er luider gesproken wordt in het 2^{e} fragment dan in het 1ste. Toch moet er hier een duidelijke kanttekening bij gemaakt worden. Wanneer een persoon bij de 2^{e} opname luider praat, maar significant verder van de microfoon staat, zal de applicatie dit foutief detecteren dat dit niet luider is. Daarnaast is een significante hoeveelheid van de oudere mensen slechthorend, waardoor men wel luider moet praten. Ondanks deze twee beperkingen is het belangrijk dat deze eigenschap geïmplementeerd werd zodat men er wel eens bij stil staat dat niet iedereen slechthorend is of een hoorapparaat draagt.

Om een getal te verkrijgen dat het volume voorstelt, is er gebruik gemaakt van de pyln-bibliotheek die een BS.170 geluidsmeter aanmaakt in de code. Deze analyseert dan de audio en geeft een getal weer. Hoe dit precies geïmplementeerd werd, is te vinden in bijlage B.8.

3.2 Front-end

De *front-end* wordt verwerkt op de server zelf, dus het Flask-*framework* is een volledig *back-end-framework* en dus geen *front-end-framework*. Dit wil zeggen dat de volledige html-pagina op de server gemaakt wordt met alles er op en er aan en dat die daarna pas wordt doorgestuurd naar de browser. Hierdoor zal de server meer werk hebben om alles te kunnen verwerken. Maar de front-end precies gemaakt wordt, is hieronder te lezen:

3.2.1 Geluid opnemen

Het geluid opnemen gebeurt volledig aan de kant van de *client* of de gebruiker. Dit gebeurt volledig met ingebouwde functies van JavaScript, de programmeertaal op een website dynamisch te bewerken. Wanneer op de knop gedrukt wordt om de audio-opname te starten, zullen er *audiochunks* worden toegevoegd aan een lijst. Die worden na de opname allemaal samengevoegd tot een *blob*, of een *binary-large object*, die dan een *mp3*-file aanmaakt. Per casus wordt er ook een andere afbeelding en tekst getoond boven de opneemknop.

3.2.2 API-afhandeling

De *mp3*-file staat dan nog steeds op het toestel van de gebruiker zelf. Dat bestand moet verstuurd worden naar de server of de back-end kant van de applicatie. Vanuit de *front-end* worden er dus *API-requests*, of netwerkverzoeken, gestuurd met het audiobestand als bijlage naar de *back-end* of de server. De server analyseert dan de verschillende methodes. Wanneer alles onderzocht is, wordt alle data verzameld en via een JSON-formaat, een algemeen leesbaar formaat op het internet, teruggestuurd naar de *client*. Daar worden de resultaten ingevuld in de voorziene html-stukken.

3.3 Testen

Om de werking, specificiteit en de sensibiliteit van de applicatie te objectiveren, zijn er automatische testen opgezet. De data die verzameld is via een online formulier, is te vinden op: https://www.jotform.com/form/213524968382060. Deze werd in het begin van het tweede semester verzameld. Eens de data opgeslagen was, werd alle data beluisterd en handmatig gelabeld.

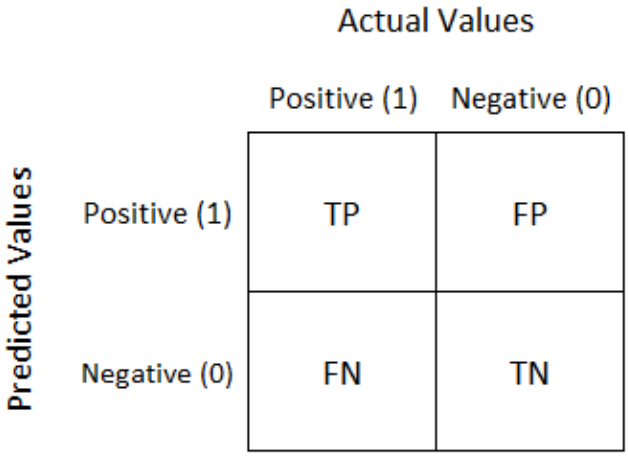
Nadien werd er een Python-script gemaakt dat het testen van die 54 bestanden automatiseerde. De audiobestanden waarbij er geen *elderspeak* aanwezig was, werden gebruikt zoals in de echte webapplicatie, om eerst een normaal stukje audio te hebben. Zo kan er toch vergeleken worden tussen een normale spraak en de spraak die erna komt. Nadien werden de geluidsopnames waarbij er wel *elderspeak* aanwezig was, gebruikt voor het testen van de applicatie zelf. De resultaten werden dan vergeleken met de gelabelde data.

Met deze resultaten kon er een *confusion matrix* gemaakt worden over de eigenschappen verkleinwoorden, hogere toonhoogte en hoger volume. Hierbij wordt er bepaald hoeveel correct-negatieve, correct-positieve, vals-positieve en vals-negatieve resultaten er aanwezig waren in de testset. Dit wordt dan gevisualiseerd in een matrix, te vinden in figuur 3.5. De resultaten daarvan zijn te vinden in het resultatenhoofdstuk, namelijk Hoofdstuk 4.

3.4 Hosting

3.4.1 Hosten op lokaal netwerk

Wanneer er de parameter “host” wordt meegegeven aan het programma zelf met als waarde “0.0.0.0”, dan zal de applicatie opengesteld worden in het lokaal netwerk. De applicatie zal dan draaien op het lokaal IP-adres van de computer of server. Op Windows kan men dit opzoeken door “ipconfig” in te typen in *command prompt*. Het IP-adres is dan terug te vinden bij ofwel Ethernet-adapter of WLAN-adapter.



Figuur 3.5: Confusion Matrix (Jain, 2020)

3.4.2 Installatie op host

Om de website te kunnen hosten, moet de nodige software eerst geïnstalleerd worden. Zo kunnen alle Python-bibliotheken geïnstalleerd worden via het volgende commando:

```
pip install -r requirements.txt
```

Dit commando zorgt er voor dat alle Python-bibliotheken uit dat bestand geïnstalleerd worden zoals het tijdens de *proof-of-concept* geïnstalleerd was.

Daarnaast moet ook FFmpeg apart geïnstalleerd worden. De stappen daarvan staan beschreven op de volgende website: https://www.wikihow.com/Install-FFmpeg-on-Windows voor een Windows besturingssysteem.

3.4.3 SSL-certificaat

Zodat andere toestellen aan de laptop of server zou kunnen moet er een SSL-certificaat geïnstalleerd worden. Dit kan bekomen worden door de volgende stappen uit te voeren:

- Installeer “pyopenssl”; voorbeeld via *installer* “chocolatey”.
- “openssl req -x509 -newkey rsa:4096 -nodes -out cert.pem -keyout key.pem -days 365”
- Kopieer “cert.pem” en “key.pem” naar de plaats van de Flask-applicatie.
- Voeg de opties toe:

```
import ssl
  context = ssl.SSLContext()
context.load_cert_chain("cert.pem", "key.pem")
app.run(ssl_context=context, host='0.0.0.0', ...)
```

4. Resultaten

4.1 Resultaten van de verzamelde data

Via het formulier van "JotForm" waren er 13 inzendingen. Dat lijkt niet veel, maar belangrijk om te weten is dat het invullen ervan behoorlijk lang duurt. Er werd telkens gevraagd om een eigenschap van *elderspeak* na te bootsen door een geluidsfragment in te spreken. Dit is veel werk, waardoor mensen sneller afhaakten.

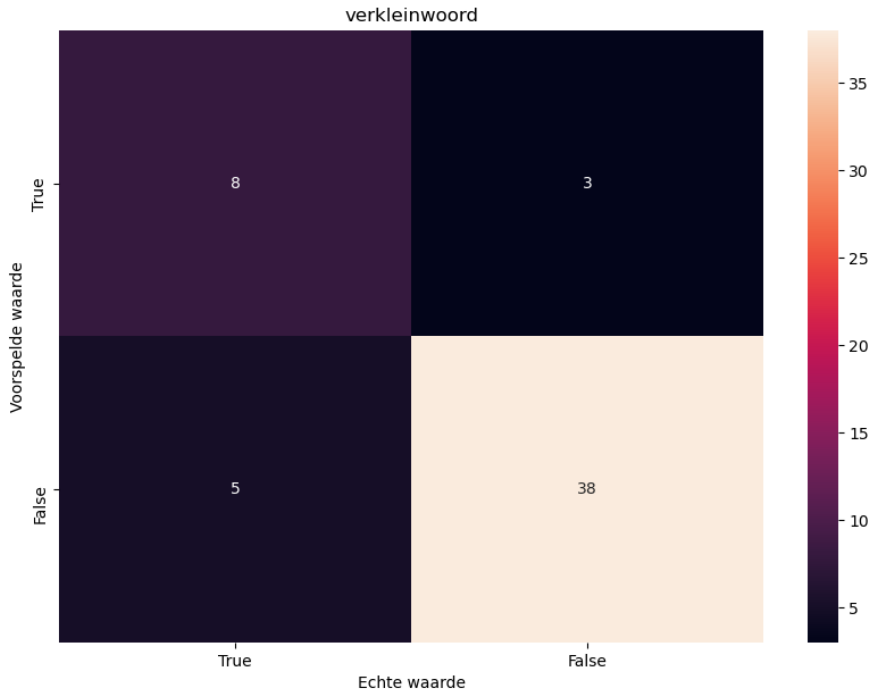
Toch gaven deze 13 personen 54 audiobestanden ter beschikking. Op die geluidsbestanden kon er gecontroleerd worden of er *elderspeak* aanwezig was.

De data is gelabeld door een persoon, dus het kan zijn dat er fouten in de classificatie van de data slopen zijn. Dit eindwerk is natuurlijk van de richting toegepaste informatica, en niet van een communicatierichting. Een interessante en interdisciplinaire opdracht kan zijn, waarbij studenten verpleegkunde of communicatie nieuwe data labelen en studenten toegepaste informatica testen uitvoeren op de server.

Meer hierover wordt beschreven in wat volgt, namelijk in Hoofdstuk 5.

4.2 Resultaten na het testen

De testen worden weergegeven in een *confusion matrix*. Daarbij is te zien dat de echte waarden op de x-as staan, waarbij de echte waarden gelijk staan aan de waarden die gegeven werden bij het labelen van de data. De waarden op de y-as zijn de voorspelde waarden van de applicatie (*back-end* of berekeningen).



Figuur 4.1: Resultaten *confusion matrix* verkleinwoorden.

4.2.1 Verkleinwoorden

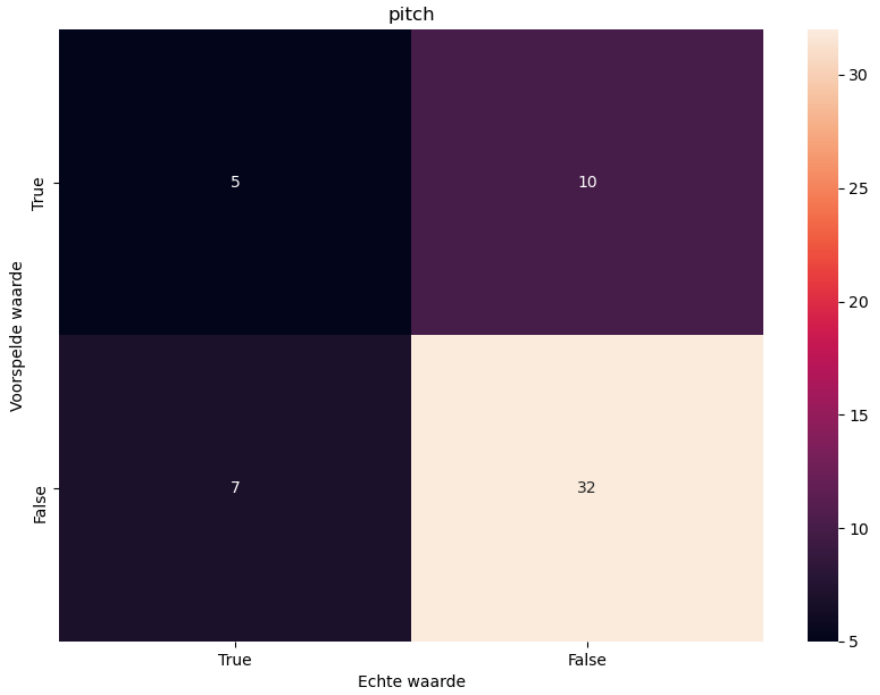
De testgevallen van de verkleinwoorden zijn te vinden in figuur 4.1. Daarbij is te zien dat er 8 correct-positieven zijn, 3 vals-positieven, 5 vals-negatieve en 38 correct-negatieven. Hieruit kunnen we afleiden dat de applicatie de verkleinwoorden correct herkend.

4.2.2 Toonhoogte

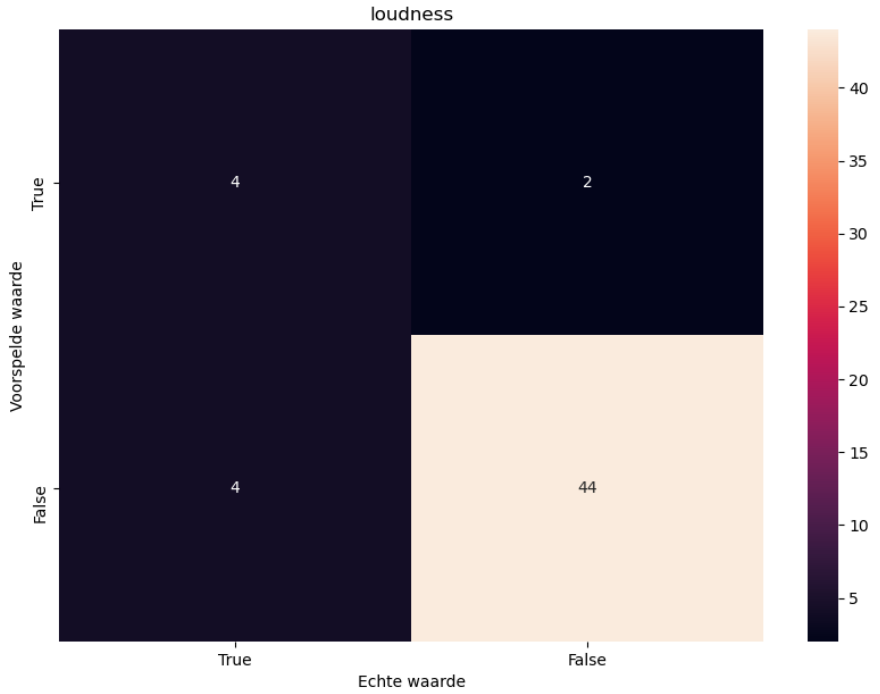
De testgevallen van de toonhoogte is te vinden in figuur 4.2. Daar is te zien dat er 5 correcte positieven zijn, 10 vals positieven, 7 valse negatieve en 32 correctie negatieven. Hieruit kunnen we afleiden dat de applicatie een beetje werkt voor de toonhoogte, maar zich ook regelmatig vergist.

4.2.3 Stemvolume

De testgevallen van het stemvolume zijn te vinden in figuur 4.2. Daar is te zien dat er 4 correct-positieven zijn, 2 vals-positieven, 4 vals-negatieve en 44 correct-negatieven. Hieruit kunnen we afleiden dat de applicatie goed werkt, maar toch een zekere foutmarge heeft.



Figuur 4.2: Resultaten *confusion matrix* toonhoogte.



Figuur 4.3: Resultaten *confusion matrix* stemvolume.

4.2.4 Algemeen besluit na testen

A.d.h.v. deze *confusion matrices* kan geconcludeerd worden dat er te weinig samples zijn om een beeld te vormen over de effectieve werking van de applicatie. Er moeten meer gevallen verzameld worden waarbij de toonhoogte of het stemvolume hoger is en waarbij er verkleinwoorden aanwezig zijn.

4.3 Rollenspel

Wanneer er een rollenspel gespeeld wordt, dan wordt de audio van beide personen geanalyseerd. Het is niet direct mogelijk om een stem weg te filteren als beide personen aanwezig zijn in de kamer. Alleen wanneer een van de twee personen significant verder staat van de microfoon, zal de applicatie die stem kunnen wegfilteren als achtergrondlawaai.

Toch kan er nog steeds geanalyseerd worden of er *secondary baby talk* aanwezig is, dus de applicatie kan ook gebruikt worden om conversaties te oefenen.

4.4 Is er aan de requirements voldaan?

De *requirements* zijn wel degelijk voldaan. Zo kan de applicatie de *elderspeak* detecteren, weliswaar met een foutmarge die te lezen is hierboven. Daarnaast kan de webapplicatie geïnstalleerd worden op een computer en heeft alles een een mooie lay-out gekregen. Zo waren de applicatie en de scriptie ruimschoots op tijd klaar. Daarnaast is er ook een vervolghoofdstuk geschreven in Hoofdstuk 5 zodat dit onderzoek niet tevergeeft werd uitgevoerd.

4.5 Live-demo

Een live-demo van de applicatie kan gevonden worden via de volgende link op Vimeo: TODO In die video wordt gedemonstreerd hoe de applicatie werkt en er uit ziet.

5. Vervolg

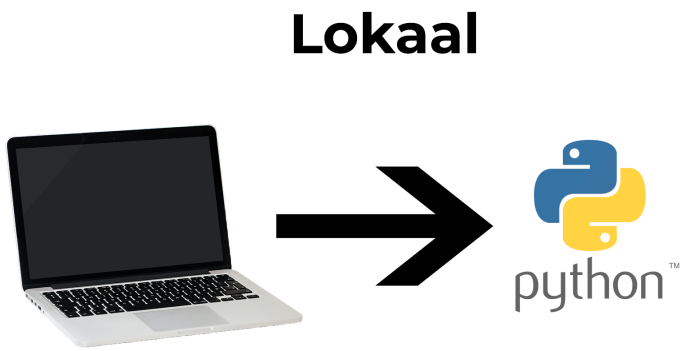
In dit hoofdstuk zullen mogelijke vervolgen en toepassingen van deze bachelorproef beschreven worden. Hoe kunnen anderen toegang krijgen tot de programmeercode, de website raadplegen of hoe kan de *webserver* geïnstalleerd worden binnen het HOGENT-netwerk. Ook zal er besproken worden hoe de data beter en objectiever gelabeld kan worden. Omdat het doelpubliek van dit hoofdstuk anders is dan het doelpubliek van het meer technische luik van dit onderzoek, zal alles zo eenvoudig mogelijk uitgelegd worden.

5.1 Applicatie-code delen

Om er zeker van te zijn dat deze bachelorproef niet onder het stof verdwijnt en de applicatie niet meer gebruikt zal worden, is het noodzakelijk dat de volledige programmacode gedeeld kan worden. Hiervoor zal "GitHub" gebruikt worden.

Om het principe van GitHub te begrijpen, is het eerst nodig te verstaan wat Git zelf is. Git is een open source versiebeheersysteem. Telkens ontwikkelaars met de eigen code aan de slag gaan en hun eigen wijzigingen aanbrengen, worden op die manier de verschillen in code opgeslagen. Versiebeheersystemen beheren deze revisies door alle wijzigingen op te slaan in een centrale opslagplaats. Hierdoor kunnen ontwikkelaars gemakkelijk samenwerken, omdat ze een nieuwe versie van de software kunnen downloaden, wijzigen en zelf nieuwe revisies kunnen uploaden. Elke programmeur van dat project kan dan op zijn of haar beurt opnieuw zien welke wijzigingen er doorgevoerd zijn en kan deze opnieuw downloaden (Brown, 2019).

GitHub is dus zo een voorbeeld van een centrale opslagplaats waar alle wijzigingen opge-



Figuur 5.1: Opstelling bachelorproef

slagen zijn in de *cloud*.

Alle code, bestanden en revisies van deze volledige bachelorproef zijn dan ook te vinden op de persoonlijke GitHub-link: https://github.com/SibianDG/BachelorProef. Wanneer iemand toegang wil, dan kan die een e-mail sturen naar Jorrit Campens die u dan in contact brengt met de schrijver, Sibian De Gussem.

De code zal ook bijgevoegd worden op de scriptietool van HOGENT als bijlage, maar er kan niet gegarandeerd worden dat dit de recentste code is.

5.2 Hosting en bereikbaarheid van de applicatie

Om te kunnen bespreken hoe men de applicatie het best beschikbaar maakt, is het nodig om de theoretische achtergrond van hosting te vermelden.

De huidige opstelling van dit eindwerk is geïllustreerd in figuur 5.1. Daarbij is alle code geïnstalleerd, op die lokale computer. Wanneer het bestand 'website.py' wordt uitgevoerd, zal Python de code uitvoeren en een webserver opzetten op die lokale machine. In de output verschijnt er dan een website adres, langs waar de applicatie gebruikt kan worden.

Wanneer de website beschikbaar gemaakt moet worden voor meerdere computers moet de code op een server geïnstalleerd worden, maar een server kan ook een gewone computer zijn. Zo kunnen er meerde *clients* een aanvraag of *request* sturen naar de server. Die zal alles verder afhandelen, waaronder het tonen van de webpagina en het uitvoeren van de berekeningen. Een illustratie is te vinden in figuur 5.2. Daarbij is te zien dat een verzoek verstuurd wordt naar het internet wanneer een gebruiker of *user* naar een website surft. Het internet weet op zijn beurt aan welke webserver het informatie kan vragen of sturen. De rol van de server is het afhandelen van de html-, en JavaScript-pagina's, websitebestanden, en doet ook de berekeningen.

Figuur 5.2: Hoe werkt webhosting (Tamara, 2022).

5.2.1 Installeren op een computer

In de hoofdmap met alle code is er een bestand te vinden genaamd “requirements.txt”. Door het commando:

```
pip install -r requirements.txt
```

uit te voeren zullen alle Python-bibliotheken geïnstalleerd worden die nodig zijn om het project uit te voeren. Er dient wel nog een extra stap gebeuren, namelijk FFmpeg apart installeren. De instructies daarvan staan beschreven op de volgende website: https://www.wikihow.com/Install-FFmpeg-on-Windows voor een Windows besturingssysteem.

Voordelen

- Snelle opzet voor IT’ers.
- Snelle verwerking, omdat het op één computer gebeurt en er maar één persoon toegang heeft.
- Gratis oplossing.

Beperkingen

- Andere computers hebben geen toegang
- Niet-IT’ers kunnen het mogelijk minder makkelijk installeren
- De software zou op elke computer opnieuw geïnstalleerd moeten worden.

5.2.2 Hosten in een lokaal netwerk

Het is ook mogelijk om de software op een computer te installeren, met een speciale parameter. Wanneer in de Flask-applicatie de volgende parameter " host='0.0.0.0' " wordt meegegeven, zal de applicatie opengesteld worden in het lokale netwerk. Dit maakt het mogelijk om naar het lokale IP-adres te surfen met een ander toestel en ook de website met de detector te gebruiken. Als men het argument *threaded* op *true* zet, dan is Flask *multithreaded* (Vieira, 2013). Hierdoor kunnen er zeker meerdere toestellen tegelijk verbonden worden. Een systematische voorstelling is te vinden in figuur 5.3. Wanneer men verbonden is met het Eduroam-netwerk, het netwerk dat in elke hogeschool of universiteit aanwezig is, dan zal dit niet lukken volgens de IT-dienst van HOGENT: "Binnen het Eduroam-netwerk is het niet mogelijk dat andere toestellen aan jouw eigen toestel kunnen.".

Voordelen

- Een eenmalige installatie op een computer, bijvoorbeeld de docent, is voldoende om het programma altijd te kunnen opstarten in de les.
- Meerdere connecties zijn mogelijk.
- Het is een gratis oplossing

Beperkingen

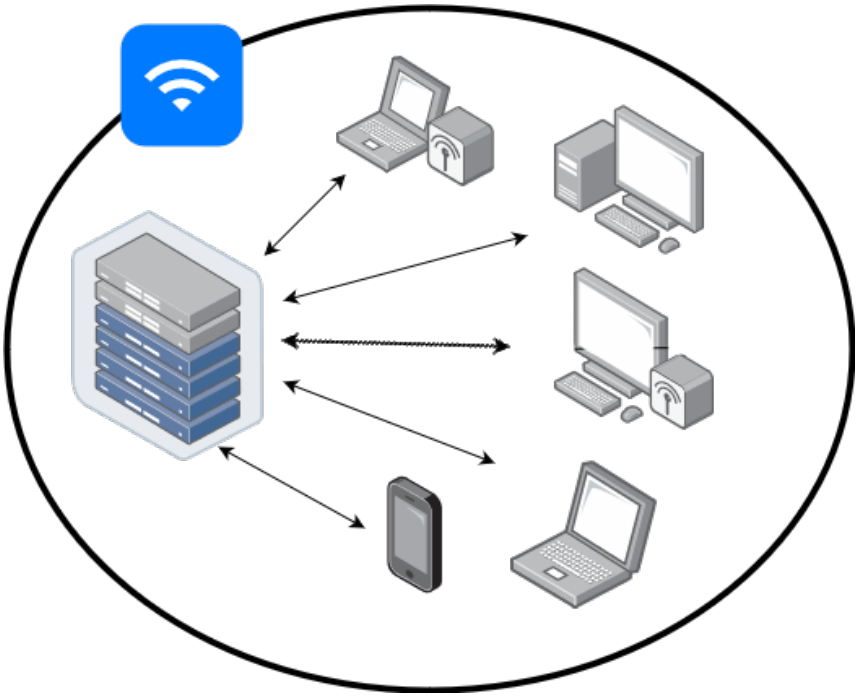
- Het maximale aantal connecties is afhankelijk van de computer waarop de applicatie wordt opgestart. Hoe beter en sneller de computer is, hoe meer connecties mogelijk zijn en hoe sneller de applicatie functioneert.
- Er moet een SSL-certificaat gemaakt en geïnstalleerd worden voor een beveiligde HTTPS-verbinding.
- Het is niet mogelijk op het Eduroam-netwerk.

5.2.3 Hosting online

De applicatie kan ook online gehost worden, maar dit zal natuurlijk geld kosten. Zo'n applicatie kan dan gehost worden op Microsoft Azure als webapplicatie, maar ook via Combell. Elke student en docent krijgt zo driejaarlijks een website op de volgende link: https://www.academicsoftware.eu/software/298/857.

Mogelijkheden

- Eens opgezet is het altijd en overal beschikbaar.
- Wanneer er veel verbindingen en berekeningen nodig zijn, zullen de servers van de aanbieder automatisch geüpgraded worden zodat ze alles kunnen bolwerken.
- Heel snelle bewerkingen.



Figuur 5.3: Lokaal netwerk

Beperkingen

- Dit zorgt voor een grote financiële kost.

Hosten via Docker

Een mogelijkheid is ook om de website te hosten via “Docker”. Docker is een container-technologie waar in één container draait. Maar deze optie lag buiten de scope van dit eindwerk. Toch kan dit eventueel dienen als vervolg voor de richting toegepaste informatica: “Host de *elderspeak*-website met een Docker-container.”.

5.3 Testen

Het aantal testen dat verzameld werd, is niet van een bijzonder grote grootteorde. Er kunnen extra testen, in de vorm van nieuwe audio-opnames, opgenomen worden in een vervolgopdracht. Dit kan een interdisciplinaire opdracht zijn met de richting verpleegkunde, communicatie en eventueel toegepaste informatica. Immers hoe meer testen, hoe groter de accuraatheid van de applicatie.

Een mogelijke opzet van deze opdracht wordt hier besproken. De studenten verpleegkunde maken en/of klasseren audio-opnames van eventuele *elderspeak*.

Studenten uit communicatierichtingen kunnen alle data dan opnieuw labelen, aangezien zij veel kennis hebben rond dit fenomeen. Zo kan de data zeer nauwkeurig en uitgebreid

geanalyseerd worden, terwijl in dit eindwerk de data gelabeld werd door een persoon die geen expert is.

Toch kunnen de studenten informatica ook betrokken worden. Ze kunnen zelf een programma schrijven om de gelabelde data te laten analyseren door de applicatie. Dit is evenwel al beschikbaar in de bachelorproef, maar beginnende studenten kunnen door deze opdracht gestimuleerd worden met een goed voorbeeld en een duidelijk doel. Wanneer de data uitgebreider geanalyseerd wordt, moeten de testen enigszins herschreven worden, wat ook een oefening kan zijn.

6. Conclusie

Uit deze bachelorproef kan besloten worden dat *elderspeak* of *nursery tone* onrechtstreeks kan gedetecteerd worden met artificiële intelligentie. Het verkozen type AI is verwerkt in de Google *Speech Recognition*-API. Op basis van die spraakherkenning wordt er gedetecteerd of verkleinwoorden, herhalingen, collectieve voornaamwoorden en/of tussenwerpsels aanwezig zijn. Daarnaast worden ook de toonhoogte en het stemvolume berekend via Python-bibliotheken die geen gebruik maken van kunstmatige intelligentie.

Het type model dat de Google *Speech Recognition*-API gebruikt, is natuurlijk niet zomaar online terug te vinden. Het is wel algemeen bekend dat Google aan *natural language processing* doet. Het achtergrondlawaai kan gemakkelijk worden weggefilterd via de volgende methode in de Google *Speech Recognition*-API: 'ajust_for_ambient_noice(source)'.

Spraakherkenning in Python is mogelijk via een gratis softwarebibliotheek, mits enkele aanpassingen. Zo is het noodzakelijk om *wav*- en *flac*-bestanden van de audio te hebben. Met andere formaten kan de bibliotheek niet overweg. Gratis gebruik van de software is ook gelimiteerd. Wanneer het geluidsbestand langer is dan 2 à 3 minuten, geeft de API een foutboodschap. Wanneer het geluid opgedeeld wordt in deelbestandjes of *chunks* lukt het wel.

Het opzetten van een "Flask applicatie" is bijzonder simpel. Het is een goede manier om snel een webserver op te zetten in Python. Het model ermee verbinden of, in ons geval, de methoden aanroepen, is ook eenvoudig. Er kan gemakkelijk een ander bestand dat alle berekeningen heeft, geïmporteerd worden.

Om te kunnen staven dat *elderspeak* goed gedetecteerd kan worden, is er nood aan meer testgevallen. Zo is het momenteel nog niet volledig duidelijk of de methoden die de

toonhoogte en het stemvolume bepalen, goed genoeg werken. Wat de mogelijke vervolgopdrachten of -studies zijn, is te vinden in Hoofdstuk 5.

6.1 Discussie

In deze discussie wordt besproken of het nuttig is voor verpleegkundigen om deze applicatie in gebruik te nemen.

Concluderend kunnen we stellen dat deze applicatie zeker en vast gebruikt kan worden voor studenten in de zorgsector. Op die manier kunnen ze *elderspeak* ontdekken en het fenomeen beter leren herkennen. Dat kan zowel actief, via de detector, als passief, door de lijst van eigenschappen en tips te lezen.

De accuraatheid is niet ideaal, maar dat hoeft ook niet. Wanneer iemand de detector gebruikt, is het belangrijk dat hij of zij ook aan zelfreflectie doet. Op die manier blijven het begrip en de eigenschappen langer aanwezig in het geheugen.

Met alle argumenten die aangehaald zijn in deze tekst, kan er toch wel besloten worden dat deze applicatie nuttig zal zijn in de toekomst. Hopelijk wordt de website wel degelijk ingezet tijdens de lessen *elderspeak* in de richting verpleegkunde.