HO
GENT

Faculteit Bedrijf en Organisatie

Nursery Tone Monitor: detecteren van elderspeak via AI

Sibian De Gussem

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

Promotor:
Geert Van Boven
Co-promotor:
Jorrit Campens

Instelling: —

Academiejaar: 2021-2022

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Nursery Tone Monitor: detecteren van elderspeak via AI

Sibian De Gussem

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

Promotor:
Geert Van Boven
Co-promotor:
Jorrit Campens

Instelling: —

Academiejaar: 2021-2022

Tweede examenperiode

Woord vooraf

Ik wilde een onderwerp kiezen dat een impact heeft op de maatschappij en waarbij ik zowel artificiële intelligentie als websiteontwikkeling kon gebruiken. Enerzijds om te kunnen aantonen dat AI niet altijd een negatieve connotatie moet hebben en anderzijds omdat ik AI & *Data Engineering* had gekozen als afstudeerrichting. Op die manier kon ik ondervinden of dit iets voor mij is, of toch eerder het programmeren an sich... (*spoiler*: het is programmeren geworden.)

Het was echt een interessant onderwerp! Aangezien er onze maatschappij aan het vergrijzen is, neemt de nood aan ouderenzorg toe. Er is daarbij een duidelijk verschil tussen zorg op papier en kwalitatieve zorg in het echte leven. Ik hoop dat studenten in de opleiding verpleegkunde dankzij mijn bachelorproef minder *elderspeak* zullen gebruiken, zodat ouderen niet behandeld worden als kleine kinderen.

Als laatste puntje wil ik iedereen bedanken die me geholpen heeft met dit eindwerk. Zo heeft mijn zus Shauni de tekst meermaals gelezen en verbeterpuntjes aangehaald. Ook vrienden en familie die mijn scriptie hebben nagelezen wil ik bedanken! Als laatste bedank ik mijn promotor, Geert van Boven, en mijn co-promotor, Jorrit Campens, om feedback, maar ook tips en uitleg te geven doorheen het hele proces!

Dankjewel aan iedereen, zonder jullie was dit niet zo goed gelukt!

Samenvatting

Deze bachelorproef behandelt het detecteren van *elderspeak* a.d.h.v. een webapplicatie. *Elderspeak* is het fenomeen waarbij jongere mensen tegen ouderen spreken op een betuttelende manier, vandaar dat *elderspeak* ook wel als *secondary baby talk* wordt benoemd.

Hopelijk kan deze applicatie gebruikt worden bij opleidingen in de zorgsector. De studenten kunnen dit leren in een practicum waarbij ze situaties moeten naspelen. De website zal dan aangeven of er *elderspeak* aanwezig is.

Dit optimaliseert de communicatie tussen zorgpersoneel en ouderen, waardoor er kan voorzien worden in een kwaliteitsvolle zorg. Op die manier is er de mogelijkheid om een waardige levenseindefase te voorzien.

Om aspecten van *elderspeak* te kunnen herkennen in een gesprek, was de eerste stap het ijken van spraakherkenningssoftware met de kenmerken van *elderspeak*. Spraakherkenning gebruikt achterliggend een geavanceerd artificieel intelligentie-model en daarnaast wordt *natural language processing* of *NLP* toegepast om de precisie te verbeteren. De spraakherkenning kan enkel werken bij een minimum aan achtergrondlawaai. Deze drie technieken worden beschreven in het literatuuronderzoek.

Nadien beschrijft deze bachelorproef hoe de applicatie van scratch opgebouwd is, welke methoden en *frameworks* gebruikt worden en hoe alles met elkaar verweven werd. Zo werd een *micro-framework* Flask in de programmeertaal Python gebruikt om snel een webapplicatie op te zetten. Via de Google *Speech Recognition*-API en andere Python-bibliotheken kan *elderspeak* gedetecteerd worden.

In het volgende hoofdstuk worden de resultaten van het testen beschreven. Zo werd er data verzameld en werden geïmplementeerd in een Python-script dat automatisch *confusion*

matrixes gebruikt om de accuraatheid van de applicatie weer te geven. Daarbij wordt er ook beschreven of er aan alle *requirements* voldaan is om de applicatie af te leveren.

Als voorlaatste hoofdstuk wordt er beschreven hoe dit eindwerk kan verder evolueren en wat een mogelijk vervolg kan zijn in een multidisciplinaire omgeving. Via GitHub kan de code gedeeld worden en kan de applicatie gemakkelijk geïnstalleerd worden op een andere computer. Zo kan de applicatie geraadpleegd op een computer, maar via een parameter kunnen ook andere apparaten gebruik maken van de website.

Deze bachelorproef besluit dat het mogelijk is om *elderspeak* te detecteren via AI, specifiek via spraakherkenning en Python-methoden. Dit werd gevisualiseerd op een website zodat studenten in de zorgsector op beroep kunnen doen op deze tool om te oefenen.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

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*. 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 dat 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 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 gaat over stereotiepe communicatie tussen ouderen en jongeren.

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.

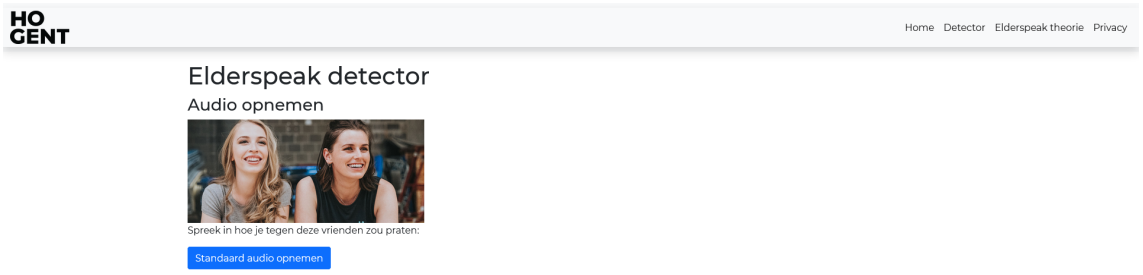
Figuur 3.1: Home pagina website

3.1 Berekeningen in back-end

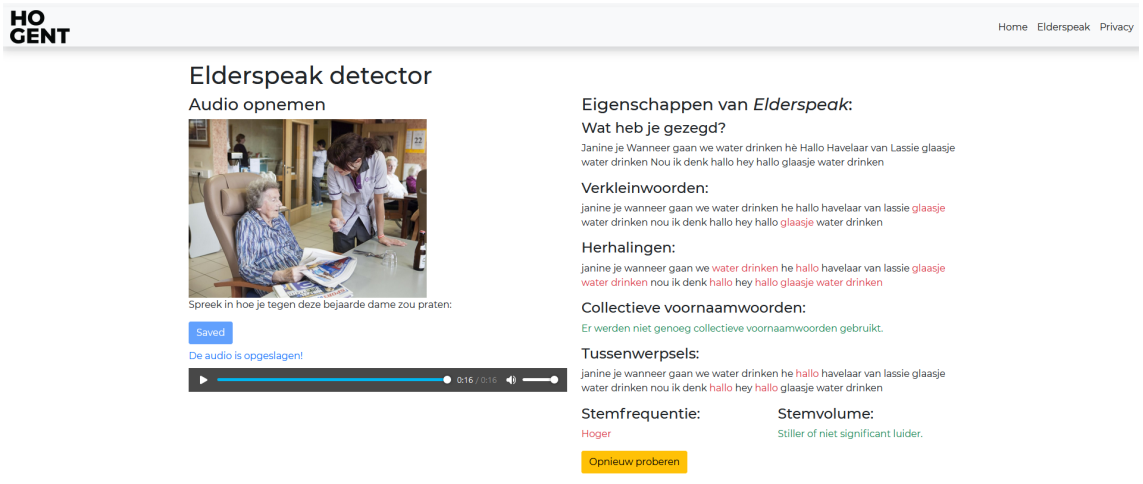
Deze voorbeeldapplicatie, geschreven in Python en gebruikmakend van het *micro-framework* Flask, is een webapplicatie die verschillende webpagina's bevat. De code van de volledige Flask-applicatie is te vinden in bijlage B.1. Naast de inleidende pagina, zie schermafbeelding 3.1, bevat deze ook een detector. Voor het begin zie schermafbeelding 3.3, en na het analyseren ziet de webpagina er als volgt uit, zie schermafbeelding 3.4. Er kan ook bestudeerd worden wat *secondary baby talk* is in de vorm van een oplijsting, zie schermafbeelding 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 jongen 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". Hoe de conversie 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.

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. 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 dat 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.

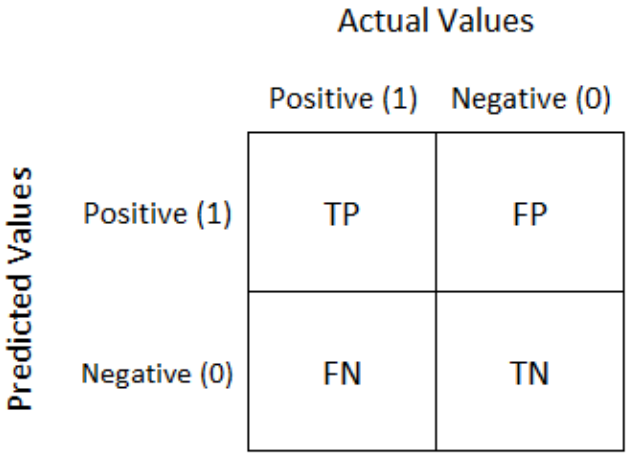
3.2 Front-end

3.2.1 Geluid opnemen

Het geluid opnemen gebeurt volledig aan de kant van de *client* of de gebruiker. 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

Vanuit de *front-end* worden er *API-requests* gestuurd met het audiobestand als bijlage naar de *back-end*. De server analyseert dan de verschillende methodes. Wanneer alles onderzocht is, wordt alle data verzameld en via een JSON-formaat teruggestuurd naar de *client*. Daar worden de resultaten ingevuld in de voorziene html-stukken.



Figuur 3.5: Confusion Matrix (Jain, 2020)

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 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
```

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.

3.4.2 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 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.

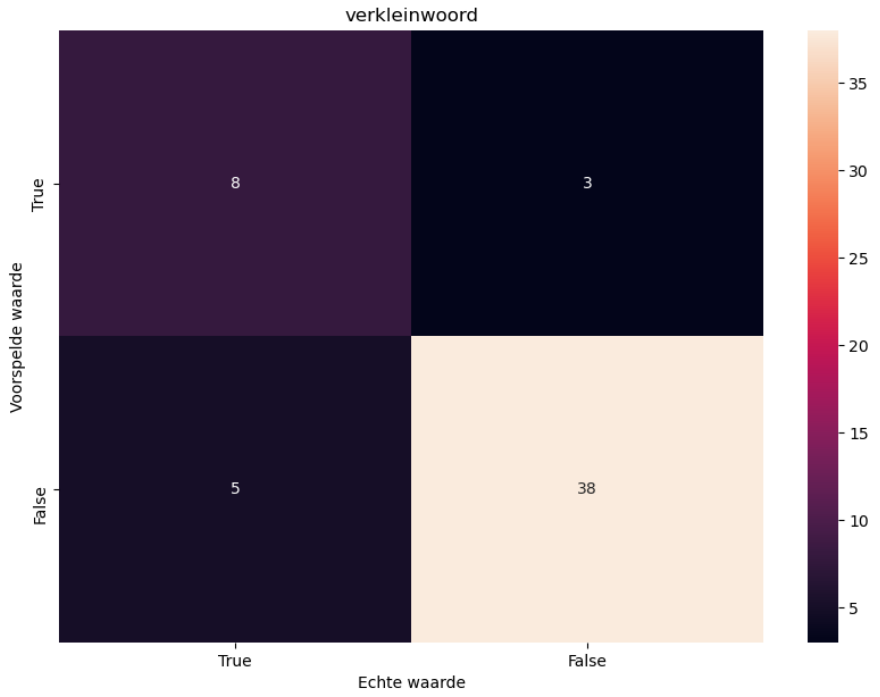
Toch gaven deze 13 personen 54 audiobestanden ter beschikking. Op die geluidsbestanden kon er gecontroleerd worden of er *elderspeak* aanwezig was.

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 er op de x-as de echte waarden 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

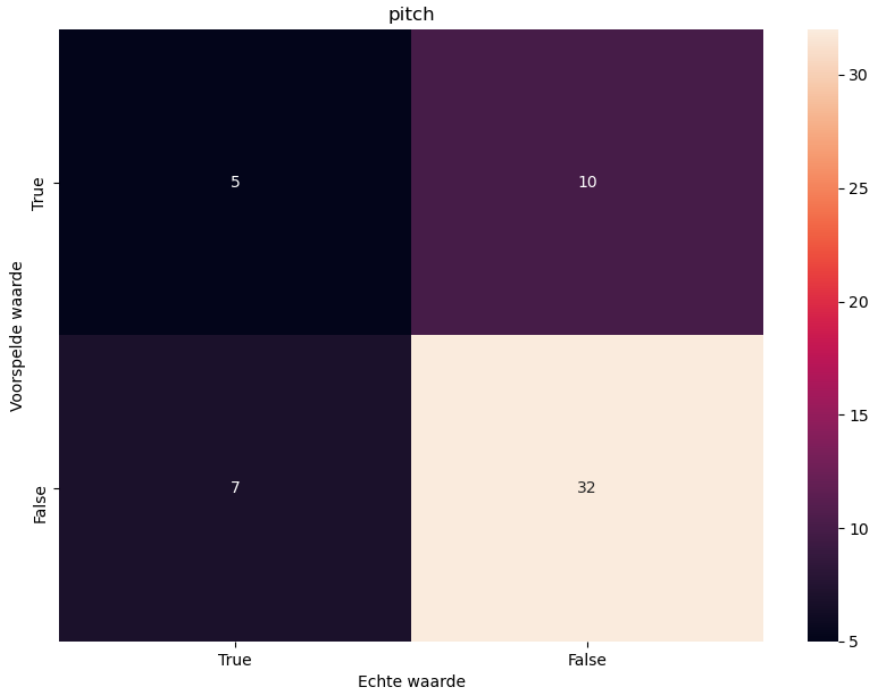
De testgevallen van de verkleinwoorden zijn te vinden in *confusion matrix* 4.1. Daarbij is te zien dat er 8 correcte positieven zijn, 3 vals positieven, 5 valse negatieve en 38 correct negatieven. Hieruit kunnen we afleiden dat de applicatie goed de verkleinwoorden weet te vinden.

4.2.2 Toonhoogte

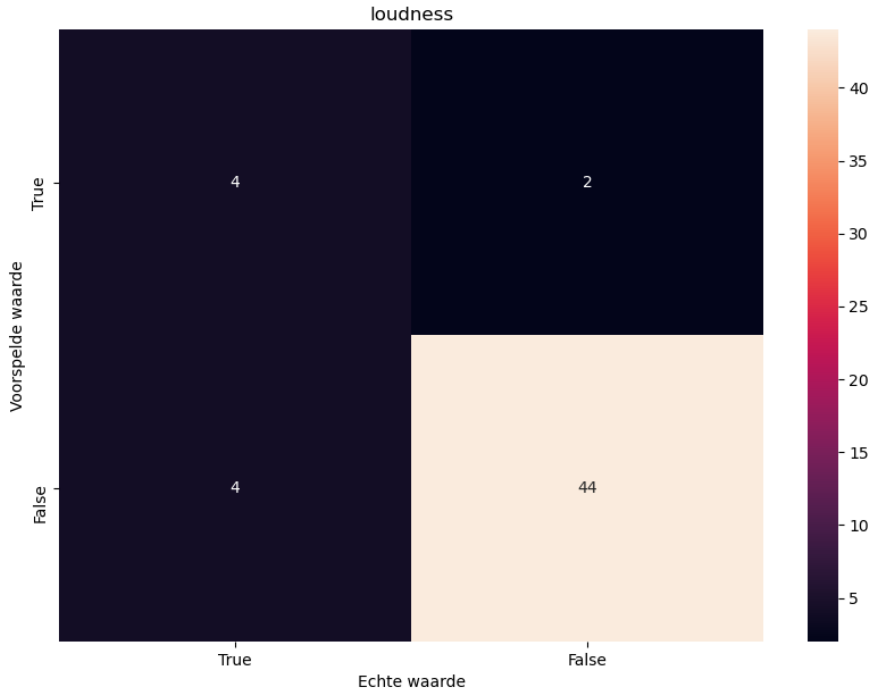
De *confusion matrix* van de testgevallen van de toonhoogte is te vinden in figuur 4.2. Daar is te zien dat er 5 correcte positieven zijn, 10 vals positieven, 7 valse negatieve en 32 correctie negatieven. Hieruit kunnen we afleiden dat de applicatie een beetje werkt voor de toonhoogte, maar dat hij er soms wel naast durft te zitten.

4.2.3 Stemvolume

De *confusion matrix* van de testgevallen van het stemvolume is te vinden in figuur 4.2. Daar is te zien dat er 4 correcte positieven zijn, 2 vals positieven, 4 valse negatieve en 44 correctie negatieven. Hieruit kunnen we afleiden dat de applicatie wel goed werkt, maar toch een zekere foutenmarge heeft.



Figuur 4.2: Resultaten *confusion matrix* toonhoogte



Figuur 4.3: Resultaten *confusion matrix* stemvolume

4.2.4 Algemeen besluit na testen

We kunnen wel goed zien a.d.h.v. deze *confusion matrixes* dat er te weinig testgevallen zijn om een mooi beeld te vormen of de applicatie werkt. We moeten meer gevallen verzamelen waarbij de toonhoogte of stemvolume hoger zijn 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 staan 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 gebruikt worden om conversaties te oefenen.

4.4 Zijn de *requirements* voldaan?

De *requirements* zijn weldegelijk voldaan. Zo kan de applicatie de *elderspeak* detecteren, weliswaar met een foutenmarge die te lezen is in de resultaten na het testen. Daarnaast is de webapplicatie te draaien op een computer en is alles in een mooie lay-out gegoten. Zo waren de applicatie en de paper meer dan genoeg op tijd klaar. Daarnaast is er ook een vervolg hoofdstuk geschreven in Hoofdstuk 5 zodat dit onderzoek niet voor niets was.

5. Vervolg

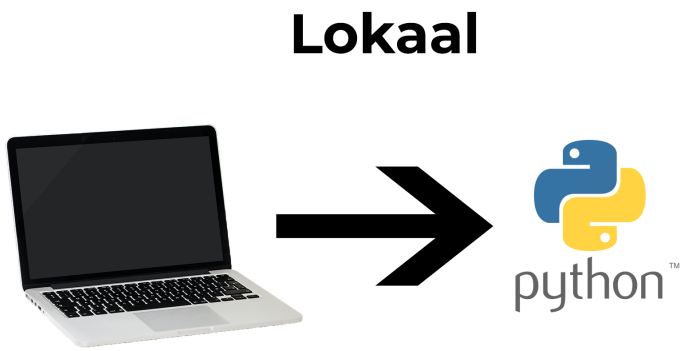
In dit hoofdstuk zullen mogelijke vervolgen en toepassingen van deze bachelorproef beschreven worden. Hoe kunnen anderen toegang krijgen tot de programmeercode, de website raadplegen of hoe kan de *webserver* geïnstalleerd worden binnen het HOGENT-netwerk. Ook zal er besproken worden hoe de data beter 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 makkelijk mogelijk uitgelegd worden.

5.1 Applicatie-code delen

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

Om GitHub te begrijpen is het eerst nodig te begrijpen 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 staan 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 men een e-mail sturen naar Jorrit Campens die u dan in contact brengt met de schrijver, Sibian De Gussem.

5.2 Hosting en bereikbaarheid van de applicatie

Om te kunnen spreken hoe we het beste de applicatie beschikbaar maken, is het eerst noodzakelijk dat iedereen weet wat alles rond hosting precies betekent.

De huidige opstelling van dit eindwerk is geïllustreerd in figuur 5.1. Daarbij is alle code geïnstalleerd, of eerder gezegd aanwezig op die lokale computer zelf. 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. Als er naar de website gegaan wordt ziet u de applicatie.

Wanneer we de website beschikbaar willen maken voor meerdere computers moeten we de code op een server installeren, maar een server kan ook een gewone computer zijn. Zo kunnen er meerde *clients* een aanvraag of *request* sturen naar de server die alles afhandelt zoals de webpagina tonen en de berekeningen uitvoeren. Een illustratie is te vinden op figuur 5.2. Daarbij is te zien dat wanneer een gebruiker of *user* naar een website surft, er een verzoek verstuurd wordt naar het internet. Het internet weet op zijn beurt aan welke webserver die informatie kan vragen of sturen. De server handelt de website zelf af, de html-, en JavaScript-pagina's als websitebestanden, en doet ook de berekeningen.

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

5.2.1 Installeren op een computer

Hoe kan men alles installeren op een enkele 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 wel nog een extra stap gebeuren en dat is dat FFmpeg apart geïnstalleerd moet worden. De stappen daarvan staan beschreven op de volgende website: https://www.wikihow.com/Install-FFmpeg-on-Windows voor een Windows besturingssysteem.

Mogelijkheden

- Snelle opzet voor IT’ers
- Snelle verwerking omdat het op dezelfde computer gebeurt en er maar een persoon aan kan
- Gratis oplossing

Beperkingen

- Andere computers hebben geen toegang
- Niet-IT’ers kunnen het mogelijk minder makkelijk installeren

- Alles zou geïnstalleerd moeten worden op elke computer opnieuw

5.2.2 Hosten in een lokaal netwerk

Het is ook mogelijk om de software op een computer te installeren, deze te laten aanstaan, maar met een speciale parameter. Wanneer er in de Flask-applicatie de volgende parameter wordt meegegeven: " host='0.0.0.0' ", zal de applicatie opengesteld worden in het lokale netwerk. Dit laat het mogelijk naar het lokale IP-adres te surfen op een ander toestel en ook de website met de detector te kunnen gebruiken. Als men het argument *threaded* op *true* zet, dan is Flask *multithreaded* (Vieira, 2013). Hierdoor kunnen er zeker meerdere verbindingen 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.".

Mogelijkheden

- Een keer installeren op een computer van bijvoorbeeld de docent in kwestie, en nadien kan het programma altijd opgestart worden tijdens een les.
- Meerdere connecties mogelijk
- Een gratis oplossing

Beperkingen

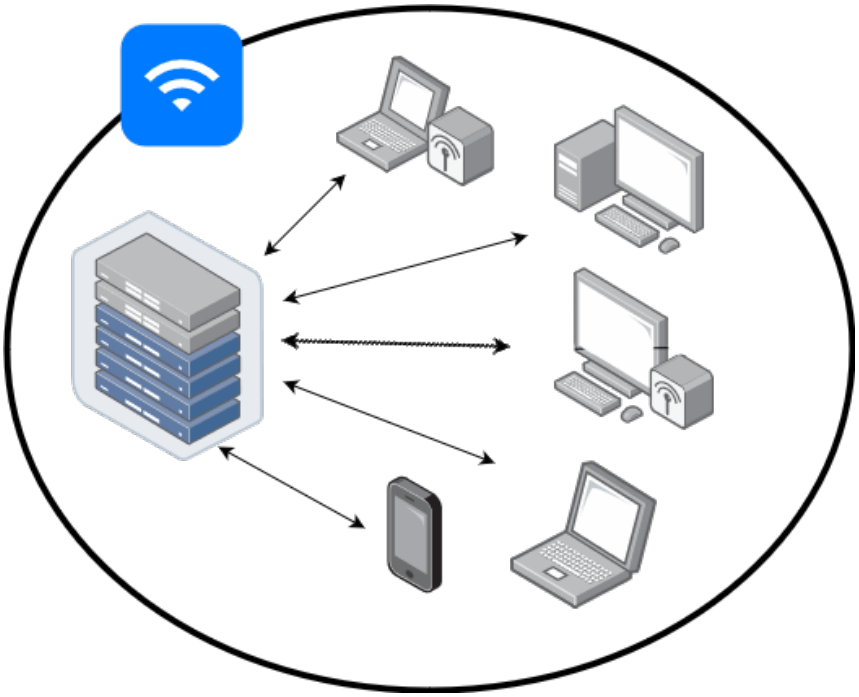
- Het maximale aantal connecties is afhankelijk van de computer waar de applicatie wordt opgestart. Hoe beter en sneller de computer, hoe meer connecties mogelijk en hoe sneller.
- Er moet een SS- certificaat gemaakt en geïnstalleerd worden voor een beveiligde HTTPS-verbinding.
- Niet mogelijk op het 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 een website om de drie jaar op de volgende link: https://www.academicsoftware.eu/software/298/857.

Mogelijkheden

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



Figuur 5.3: Lokaal netwerk

- Heel snel

Beperkingen

- Grote financiële kost

5.3 Testen

De testen die verzameld zijn, 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 en/of communicatie en eventueel samen met de richting toegepaste informatica. Hoe meer testen, hoe accurater de productiviteit van de applicatie.

Een opdracht bij de richting verpleegkunde kan het volgende zijn: de studenten maken en/of klasseren audio-opnames van eventuele *elderspeak*. Hierna kan de gelabelde data gevoed worden aan het Python-script dat er geschreven is voor deze bachelorproef. Hierna kan er achterhaald worden hoe goed de applicatie presteert.

In een richting communicatie kan alle data opnieuw gelabeld worden, waar men veel kennis heeft rond dit fenomeen. Zo kan de data zeer nauwkeurig en uitgebreid geanalyseerd worden, terwijl er in dit eindwerk de data gelabeld is door een persoon die geen expert is.

Ook kan dit een opdracht vormen voor studenten toegepaste informatica om de gelabelde

data naar de server door te sturen en een resultaat terug te krijgen. Dit is wel al beschikbaar in deze bachelorproef, maar studenten die dat nog moeten leren kunnen zo een echt voorbeeld en een duidelijk doel om dat te programmeren. Wanneer de data uitgebreider geanalyseerd wordt, moeten de testen lichtjes herschreven worden, dus dit kan ook een voorbeeld zijn van een opdracht.

6. Conclusie

Met deze bachelorproef kan besloten worden dat *elderspeak* of *nursery tone* onrechtstreeks kan gedetecteerd worden met artificiële intelligentie. Het verkozen type kunstmatige intelligentie zit verwerkt in de Google *Speech Recognition*-API. Op basis van die spraakherkenning wordt er gedetecteerd of er verkleinwoorden, herhalingen, collectieve voornaamwoorden en 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 te vinden omdat anders andere mensen dit kunnen kopiëren. Wat er wel geweten is, is dat Google aan *natural language processing* doet. Het achtergrondlawaai kan makkelijk 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 te hebben van de audio. Met andere formaten kan de bibliotheek niet overweg. Er is ook een limiet om de software gratis te gebruiken. Wanneer het geluidsbestand langer is dan 2 - 3 minuten, geeft de API een foutboodschap. Wanneer het geluid opgedeeld wordt in deelbestandjes of *chuncks* die dan elk op hun beurt de audio doorsturen, 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 eerder de methoden aanroepen, is ook gemakkelijk. Er kan gemakkelijk een ander bestand geïmporteerd worden die alle berekeningen heeft.

Om te kunnen staven dat *elderspeak* goed gedetecteerd kan worden, zullen er meer testgevallen moeten gemaakt worden. 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 er besproken of het nuttig is om deze applicatie in gebruik te nemen voor zorgkundigen.

Het valt te beargumenteren dat deze applicatie zeker en vast gebruikt kan worden voor studenten in de zorgsector. Op die manier kunnen studenten *elderspeak* beter ontdekken wat dat fenomeen is. Ze kunnen dat zowel actief, door de detector, als passief leren door de lijst van eigenschappen en tips te lezen.

De accuraatheid is misschien niet ideaal, maar dat hoeft ook niet. Wanneer er iemand de detector wil gebruiken, dan moet hij of zij ook aan zelfreflectie doen. Op die manier blijft het begrip en de eigenschappen langer in het geheugen zitten.

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

A. Onderzoeksvoorstel

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

A.1 Introductie

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

Het woord 'waardigheid' is 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 meer zorg zij nodig hebben en hoe meer zorgverleners instaan voor deze leeftijdscategorie. Die zorgverleners, maar evengoed familie, weten niet altijd even goed hoe ze moeten omgaan met senioren. Wanneer een jonger persoon op een andere manier spreekt tegen een senior dan tegen een leeftijdsgenoot, spreken we over *elderspeak*. Williams (2011) omschrijft *elderspeak* als volgt: "Elderspeak is a common intergenerational speech style used by younger persons in com-

munication 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." 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* heb je ook nog *nursery tone*. Dit verwijst naar de situatie waarbij iemand de toonhoogte aan het einde van de zin standaard verhoogt zoals bij communicatie met jonge kinderen.

Dit onderwerp was vorig jaar al een onderzoeksonderwerp voor Glenn Beeckman (2021) en Victor Standaert (2021). Zij hebben al een basis gelegd in de goede richting om dit project tot een goed einde te brengen. 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 kan dit toegepast worden in de praktijk?". Een bijkomende onderzoeksvraag is: "Kan *nursery tone* gedetecteerd worden door Artificiële Intelligentie?".

Met dit eindwerk zal ik alle mogelijkheden en capaciteiten van mezelf inzetten om een applicatie én AI-model te maken zodat dit kan getest en gebruikt worden in de opleiding verpleegkunde. Ik hoop ook dat ik ouderen op deze manier een betere levenskwaliteit kan bieden door de communicatie met zorgverleners, en misschien zelfs hun familie, te optimaliseren.

A.2 State-of-the-art

A.2.1 Literatuuronderzoek

Omdat voorgaande studenten al uitgezocht hebben wat *elderspeak* precies is, zal dit niet herhaald worden in dit onderzoek. Wel zal er op basis van de beschikbare literatuur onderzocht worden welk soort machinaal leren of *deep learning* het meest geschikt is voor deze specifieke taken. Zowel *machine learning* als *deep learning* hebben elk verschillende onderlinge modellen. Er moet dan bekeken worden welke hypothese het beste past om bovenstaande parameters te integreren in het model of verschillende modellen.

Een extra obstakel kan verschijnen wanneer er te veel achtergrond lawaai aanwezig is. Mogelijks moet er dan eerst een filter worden toegepast op de audiobestanden om dit weg te filteren zodat deze wel gebruikt kunnen worden voor het herkennen van eigenschappen op *elderspeak*.

A.2.2 Stand van zaken

Zoals reeds vermeld in de inleiding werd dit bachelorproef-onderwerp vorig jaar al gekozen door twee studenten. Zij hebben zich gefocust op de *speech-to-text*, verkleinwoorden detecteren, een frequentiemeter, herhalende zinnen herkennen, emotie-herkenner en een basisapplicatie in 'Tkinter', een standaard *Graphical User Interface* (GUI) in Python.

Beeckman (2021) vermeldde dat er nog nood was aan een methode om herhaling en verkleinwoorden te detecteren. Standaert (2021) haalde aan dat er nog onderzoek nodig was voor de spraakherkenning en de frequentiemeter om de applicatie preciezer te maken.

A.2.3 Wat is mijn aandeel?

Beide studenten hebben niet echt Kunstmatige Intelligentie gebruikt om het resultaat te bekomen. Standaert (2021) heeft wel methoden beschreven om een paar kenmerken te herkennen, maar dit gebeurt op basis van vaste parameters. Mocht AI gebruikt kunnen worden om de nauwkeurigheid op te schalen, dan zou dat alvast een winst zijn. Het gebruik van Machinaal leren of *Deep Learning*, meer specifiek een *Convolutional Neural Network* (CNN) kan een positief effect hebben op het detecteren van alle parameters rond *elderspeak*. Mijn aandeel zal dus zijn om te onderzoeken welke modellen het beste gebruikt worden om die parameters te detecteren.

Een belangrijke stap zal zijn om de twee eerder vernoemde eindwerken samen te voegen en te verbeteren. Beeckman (2021) gebruikte "Tkinter" om de *front-end* te maken, maar haalde een paar redenen aan waarom dat toch niet te verkiezen is, zoals bijvoorbeeld het amateuristische uiterlijk en de beperkte mogelijkheden. Mijn voorkeur gaat eerder uit naar het gebruik van "Flask", een *micro-webframework* in Python, dat kan gebruikt worden om een webpagina te maken en te linken naar de *back-end*. Het voordeel hiervan is dat men sneller én mooier een website kan ontwerpen.

A.3 Methodologie

Om te verzekeren dat er genoeg data beschikbaar is, is het aan te raden dat er audiosamples verzameld worden voor het 2e semester.

Op basis van de resultaten van het literatuuronderzoek en beide eindwerken van vorig jaar, kunnen er methodes opgesteld worden die de belangrijkste kenmerken van *nursery tone* en *elderspeak* herkennen. Daarbij is het gebruik van Artificiële Intelligentie een handige

manier om het verschil te kennen tussen iemand die *elderspeak* gebruikt en iemand die dat niet doet. Met welk model en op welke wijze dit het beste gerealiseerd wordt, zal onderzocht worden in dit eindwerk.

Daarnaast moet alles omgezet worden naar een duidelijke webapplicatie via "Flask" zodat het in latere fases niet geïnstalleerd moet worden op een computer. Zo kan iedereen de applicatie gebruiken zonder vooraf iets te downloaden, wat het gebruiksgemak thuis en op verplaatsing, bijvoorbeeld in een rusthuis, optimaliseert.

Tot slotte zullen het beantwoorden van de volgende deelvragen hierbij moeten helpen:

- 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 achtergrond lawaai 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?

A.4 Verwachte resultaten

Het verwachte resultaat is een webapplicatie met "Flask" als *back-end*, waarbij men de optie heeft om het model te trainen, en waarbij het model aangeeft of er *elderspeak* of *nursery tone* aanwezig is. Bovendien geeft de applicatie weer op basis van welke parameters het model 'denkt' dat het om die twee taalregisters gaat.

A.5 Verwachte conclusies

De gehoopte resultaten houden in dat er een meetbaar verschil is tussen personen die *nursery tone* gebruiken t.o.v. mensen die normaal praten. Het model zal nooit 100% accuraat zijn: zo spreekt men in de praktijk vaker dialect tegen ouderen terwijl de algoritmes getraind zijn op Algemeen Nederlands (AN), en ook het verschil tussen de Nederlandse uitspraak en de Vlaamse uitspraak kunnen een obstakel vormen. Ook het filteren van achtergrondlawaai wordt een bijkomende uitdaging.

B. Bijlagen

B.1 Flask code

```
from flask import Flask, render_template, url_for, request,
                                    redirect, jsonify
import os
from datetime import datetime
import Calculations
from Calculations import remove_uploads
import shutil

app = Flask(__name__)
app.config['UPLOAD_EXTENSIONS'] = ['.wav', '.mp3']


@app.route('/', methods=['POST', 'GET'])
def index():
    if request.method == 'POST':
        print("POST")
    else:
        return render_template('index.html')


@app.route('/detector', methods=['POST', 'GET'])
def detector():
    if request.method == 'POST':
        print("POST")
    else:
        return render_template('detector.html')


@app.route('/picture_old_woman', methods=['GET'])
```

```
def picture_old_woman():
    url = url_for('static', filename='img/rusthuis.jpg')
    print(url)
    return url


@app.route('/privacy', methods=['GET'])
def privacy():
    return render_template('privacy.html')


@app.route('/elderspeak', methods=['GET'])
def elderspeak():
    return render_template('elderspeak.html')


@app.route('/receive_elderspeak', methods=['POST'])
def receive_elderspeak():
    now = datetime.now()
    d1 = now.strftime("%Y%m%d%H%M%S")
    data = request.files['audio_data'].read()
    extra_data = request.form.get('extra_data', "0,0")
    extra_data = extra_data.split(',')
    extra_data = list(map(float, extra_data))
    pitch_normal, loudness_normal = extra_data
    file = f'./uploads/{d1}.wav'

    response_data = {"Hello": "World"}
    with open(os.path.abspath(file), 'wb') as f:
    f.write(data)

    total_text = Calculations.speech_recognition(file)
    total_text = Calculations.replace_hey(total_text)
    verkleinwoorden = Calculations.verkleinwoorden(total_text)
    herhalingen = Calculations.herhalende_zinnen(total_text)
    pitch = Calculations.make_text_compare(pitch_normal,
    Calculations.calculate_pitch(Calculations.maketempfile_wav(file)
                                 ),
         100,
        '<span class="text-danger">Hoger</span>',
        '<span class="text-success">Lager of niet significant hoger
                                    .</span>')
    loudness = Calculations.make_text_compare(pitch_normal,
        Calculations.loudness(Calculations.maketempfile_wav(file)),
        4,
        '<span class="text-danger">Luider</span>',
        '<span class="text-success">Stiller of niet significant
                                    luider.</span>')
    collectieve_voornaamwoorden = Calculations.
                                  collectieve_voornaamwoorden(
                                  total_text)
    tussenwerpsels = Calculations.tussenwerpsels(total_text)

    response_data["speech_recognition"] = total_text
    response_data["verkleinwoorden"] = verkleinwoorden
    response_data["herhalingen"] = herhalingen
```

```
    response_data["pitch"] = pitch
    response_data["loudness"] = loudness
    response_data["collectieve_voornaamwoorden"] =
                                        collectieve_voornaamwoorden
    response_data["tussenwerpsels"] = tussenwerpsels

    # return render_template('results.html', text=total_text)
    # laatste stap!

    shutil.rmtree('./uploads/chunks')
    if os.path.exists(file):
        os.remove(file)

    try:
        remove_uploads()
    except Exception as e:
        print(f"Fout bij het verwijderen van de tempfiles: {e}")

    response = jsonify(response_data)
    response.headers.add('Access-Control-Allow-Origin', '*')
    # remove_uploads()
    return response


@app.route('/receive_normal', methods=['POST'])
def receive_normal():
    now = datetime.now()
    d1 = now.strftime("%Y%m%d%H%M%S")
    file = f'./uploads/{d1}.wav'
    data = request.files['audio_data'].read()

    response_data = {"Hello": "World"}
    with open(os.path.abspath(file), 'wb') as f:
    f.write(data)

    print("PITCH BEREKENEN")
    pitch = Calculations.calculate_pitch(Calculations.
                                        maketempfile_wav(file))
    loudness = Calculations.loudness(Calculations.maketempfile_wav(
                                        file))

    response_data["pitch"] = pitch
    response_data["loudness"] = loudness

    if os.path.exists(file):
        os.remove(file)

    print(response_data)
    response = jsonify(response_data)
    response.headers.add('Access-Control-Allow-Origin', '*')
    # remove_uploads()
    return response


if __name__ == "__main__":
    try:
```

```
        app.run(
            debug=False,
            host='0.0.0.0'
        )
    finally:
        remove_uploads()
```

B.2 Speech Recognition

```
def speech_recognition(file):
    try:

        print(file, ' to chunks')
        AudioSegment.converter = which("ffmpeg")
        myaudio = AudioSegment.from_file(file)
        channel_count = myaudio.channels  # Get channels
        sample_width = myaudio.sample_width  # Get sample width
        duration_in_sec = len(myaudio) / 1000  # Length of audio in
                                               sec
        sample_rate = myaudio.frame_rate

        print("sample_width=", sample_width)
        print("channel_count=", channel_count)
        print("duration_in_sec=", duration_in_sec)
        print("frame_rate=", sample_rate)
        bit_rate = 16  # assumption , you can extract from
                                           mediainfo("test.wav")
                                           dynamically

        wav_file_size = (sample_rate * bit_rate * channel_count *
                                           duration_in_sec) / 20
        print("wav_file_size = ", wav_file_size)

        file_split_size = 25000000  # 10Mb OR 10, 000, 000 bytes
        total_chunks = wav_file_size // file_split_size

        # Get chunk size by following method #There are more than
                                           one ofcourse
        # for  duration_in_sec (X) -->  wav_file_size (Y)
        # So   whats duration in sec  (K) --> for file size of 10Mb
        #  K = X * 10Mb / Y

        chunk_length_in_sec = math.ceil((duration_in_sec * 20000000
                                           ) / wav_file_size)  # in
                                           sec
        chunk_length_ms = chunk_length_in_sec * 2000
        chunks = make_chunks(myaudio, chunk_length_ms)

        # Export all of the individual chunks as wav files

        if not os.path.exists('./uploads/chunks'):
            os.makedirs('./uploads/chunks')
```

```
        for i, chunk in enumerate(chunks):
            chunk_name = f"./uploads/chunks/chunck{i}.flac"
            print("exporting", chunk_name)
            chunk.export(chunk_name, format="flac")
    except Exception as error:
        error_message = f'Fout bij het bewerken van de audiofile: {
                                            error}.'
        print(error_message)
        return error_message

    DIR = './uploads/chunks/'

    numberOfItems = len([name for name in os.listdir(DIR) if os.
                                        path.isfile(os.path.join(DIR,
                                         name))])

    total_text = ""

    try:
        for i in range(numberOfItems):
        # Speech Recognition
        audio_file = sr.AudioFile(f'./uploads/chunks/chunck{i}.flac
                                            ')
        with audio_file as source:
            r.adjust_for_ambient_noise(source)
            audio = r.record(source)
            text = r.recognize_google(audio_data=audio, language="
                                                nl-BE")
            total_text += " " + text
            print("######## Google Recognize ####################")
            print(text)
            print("##############################################")
            return total_text.strip()
    except Exception as error:
        error_message = f'Fout bij de spraakherkenning: {error}.'
        print(error_message)
        return error_message
```

B.3 Verkleinwoorden

```
def verkleinwoorden(text):

    verkleinwoorden_array = []
    words = make_array_words(text)
    for word in words:
        if word is not None:
            if (len(word) > 3 and word not in geen_verkleinwoorden)
                                                and (
                word.endswith('je') or word.endswith('ke') or word.
                                                    endswith('kes')
                                                    or word.endswith(
                                                    'jes')):
                verkleinwoorden_array.append(word)
```

```
    if len(verkleinwoorden_array) == 0:
        return '<span class="text-success">Er zijn geen
                                            verkleinwoorden gevonden
                                            </span>'
    return highlight_words_in_text(text, set(verkleinwoorden_array)
                                        )
```

B.4 Herhalingen

```
def herhalende_zinnen(text):

    words = make_array_words(text)

    cache = []
    toBeDeleted = []
    repetition = []

    for word in words:
        if word is not None:
            while len(cache) >= 25:
                cache.pop(0)
            if word not in nietzeggendewoorden:
                cache.append(word)

    sameequals = dict()

    sameequals = {word: cache.count(word) for word in cache}

    if sameequals is not None and len(sameequals) != 0:
        for word in sameequals:
            if sameequals[word] == 1:
                toBeDeleted.append(word)
            else:
                repetition.append(word)
        for word in toBeDeleted:
            del sameequals[word]

    if len(repetition) == 0:
        return '<span class="text-success">Er zijn geen herhalingen
                                            gevonden</span>'
    return highlight_words_in_text(text, set(repetition))
```

B.5 Collectieve voornaamwoorden

```
def collectieve_voornaamwoorden(text):
    collectieve_voornaamwoorden_array = []
    words = make_array_words(text)
    for word in words:
        if word is not None and word == "we":  # TODO uitbreiden?
            collectieve_voornaamwoorden_array.append(word)
```

```
    if len(collectieve_voornaamwoorden_array) == 0:
        return '<span class="text-success">Er werden geen
                                            collectieve
                                            voornaamwoorden gebruikt
                                            .</span>'
    c = dict(Counter(collectieve_voornaamwoorden_array))
    filtered_dict = {k: v for (k, v) in c.items() if v > 1}
    l = list(filtered_dict.keys())
    if len(l) == 0:
        return '<span class="text-success">Er werden niet genoeg
                                            collectieve
                                            voornaamwoorden gebruikt
                                            .</span>'
    return highlight_words_in_text(text, set(l))
```

B.6 Tussenwerpsels

```
def tussenwerpsels(text):
    tussenwerpsels_array = []
    words = make_array_words(text)
    for word in words:
        if word is not None and word in tussenwerpels_woorden:
            tussenwerpsels_array.append(word)
    if len(tussenwerpsels_array) == 0:
        return '<span class="text-success">Er werden geen
                                            tussenwerpsels gebruikt
                                            .</span>'
    c = dict(Counter(tussenwerpsels_array))
    filtered_dict = {k: v for (k, v) in c.items() if v > 1}
    l = list(filtered_dict.keys())
    if len(l) == 0:
        return '<span class="text-success">Er werden niet genoeg
                                            tussenwerpsels gebruikt
                                            .</span>'
    return highlight_words_in_text(text, set(l))
```

B.7 Toonhoogte

```
def calculate_pitch(wav_file):
    try:
        x, _ = librosa.load(wav_file, sr=16000)
        tmp_file = './uploads/tmp.wav'
        sf.write(tmp_file, x, 16000)

        chunk = 16384
        with wave.open(tmp_file, 'r') as wf:
            swidth = wf.getsampwidth()
            RATE = wf.getframerate()
            window = np.blackman(chunk)
            p = pyaudio.PyAudio()
```

```
            stream = p.open(format=
                p.get_format_from_width(wf.getsampwidth()),
                channels=wf.getnchannels(),
                rate=RATE,
                output=True)
            data = wf.readframes(chunk)
            freqlist = []
            while len(data) == chunk * swidth:
                # write data out to the audio stream
                stream.write(data)
                # unpack the data and times by the hamming window
                indata = np.array(wave.struct.unpack("%dh" % (len(
                                                    data) / swidth),
                data)) * window
                # Take the fft and square each value
                fftData = abs(np.fft.rfft(indata)) ** 2
                # find the maximum
                which = fftData[1:].argmax() + 1
                # use quadratic interpolation around the max
                if which != len(fftData) - 1:
                    y0, y1, y2 = np.log(fftData[which - 1:which + 2
                                                        :])
                    x1 = (y2 - y0) * .5 / (2 * y1 - y2 - y0)
                    # find the frequency and output it
                    thefreq = (which + x1) * RATE / chunk
                    print("The freq is %.0f Hz." % (thefreq))
                    freqlist.append(thefreq)
                else:
                    thefreq = which * RATE / chunk
                    print("The freq is %.0f Hz." % (thefreq))
                    freqlist.append(thefreq)
                # read some more data
                data = wf.readframes(chunk)
        if data:
            stream.write(data)
        freqlistavg = sum(freqlist) / len(freqlist)
        print("Average: %0.2f Hz." % (freqlistavg))
        stream.close()
        p.terminate()
        return round(freqlistavg, 2)
    except Exception as error:
        print(f'Fout bij het berekenen van de toonhoogte: {error}.'
                                            )
        return -10000
    finally:
        if tmp_file is not None and os.path.exists(tmp_file):
            os.remove(tmp_file)
```

B.8 Stemvolume

```
def loudness(wav):
    data, rate = sf.read(wav)  # load audio (with shape (samples,
                                    channels))
```

```
    meter = pyln.Meter(rate)  # create BS.1770 meter
    loudness_range = meter.integrated_loudness(data)  # measure
                                        loudness
    if float('inf') == loudness_range:
        loudness_range = 10000
    elif float('-inf') == loudness_range:
        loudness_range = -10000
    return loudness_range
```

B.9 Front-end

B.9.1 HTML code

```
<div class="row">
<div class="col-12">
<h1>Elderspeak detector</h1>
<div class="row">
<div class="col-md-6">
<h3>Audio opnemen</h3>
<img src="{{ url_for('static', filename='img/jonge_mensen.jpg') }}"
                                    alt="rusthuis foto" class="img-
                                    fluid img-medium" id="picture">
<p id="tekst_bij_foto">Spreek in hoe je tegen deze vrienden zou
                                    praten:</p>
<button type="button" id="btn-record" class="btn btn-primary">
                                    Standaard audio opnemen</button>
<div class="d-flex flex-column">
<p id="saved" class="text-primary hidden my-2">De audio is
                                    opgeslagen!</p>
<audio id=recordedAudio></audio>
</div>
</div>
<div class="col-md-6 hidden" id="eigenschappen_elderspeak">
<h3>Eigenschappen van <i>Elderspeak</i>:</h3>
<div id="loading_eigenschappen">
<div class="spinner-border" role="status">
<span class="visually-hidden">Loading...</span>
</div>
</div>
<div id="eigenschappen_content" class="hidden">
<div class="row" id="big-content"></div>
<div class="row" id="small-content"></div>
</div>
<div class="hidden text-danger" id="errors">Er zijn fouten in het
                                    verwerken van de data...</div>
<button class="btn btn-warning" id="reset">Opnieuw proberen</button
                                    >
</div>
</div>

</div>
</div>
```

B.9.2 Javascript code

```
navigator.mediaDevices.getUserMedia({audio:true})
.then(stream => {handlerFunction(stream)})

function handlerFunction(stream) {
    rec = new MediaRecorder(stream);
    rec.ondataavailable = e => {
        audioChunks.push(e.data);
        if (rec.state === "inactive"){
            let blob = new Blob(audioChunks,{type:'audio/mp3'});
            let audioUrl = URL.createObjectURL(blob);
            let recordedAudio = document.getElementById('
                                                recordedAudio')
            recordedAudio.src = audioUrl;
            recordedAudio.controls=true;
            recordedAudio.autoplay=true;
            sendData(blob, btn_record.innerText.toLowerCase())
        }
    }
}

let loading_eigenschappen = document.getElementById('
                                loading_eigenschappen');
const btn_record = document.getElementById('btn-record');
const picture = document.getElementById('picture');
const tekst_bij_foto = document.getElementById('tekst_bij_foto');
const text_under_pic = document.getElementById('saved');
const big_content = document.getElementById('big-content');
const small_content = document.getElementById('small-content');

//normal parameters
let pitch_normal = 0;
let loudness_normal = 0;


function sendData(blob, kind) {
    let data_to_send = new FormData();
    data_to_send.append('audio_data', blob);
    if (kind.toLowerCase().includes("saved")){
        data_to_send.append('extra_data', [pitch_normal,
                                            loudness_normal])
        fetch('http://127.0.0.1:5000/receive_elderspeak', {
            method: 'POST',
            body: data_to_send
        }).then(response => {
            console.log("||||||||||||||||||||||||||||||||||")
            console.log(response)
            return response.json();
        }).then(json => {
            console.log("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&")
            console.log(json)
            big_content.insertAdjacentHTML("beforeend", `<div><h4>
                                                Wat heb je gezegd?</
                                                h4><p>${json['
                                                speech_recognition']}
```

```
                                        </p></div>‘);
            big_content.insertAdjacentHTML("beforeend", ‘<div><h4>
                                        Verkleinwoorden:</h4>
                                        <p>${json[’
                                        verkleinwoorden’]}</p
                                        ></div>‘);
            big_content.insertAdjacentHTML("beforeend", ‘<div><h4>
                                        Herhalingen:</h4><p>$
                                        {json[’herhalingen’]}
                                        </p></div>‘);
            big_content.insertAdjacentHTML("beforeend", ‘<div><h4>
                                        Collectieve
                                        voornaamwoorden:</h4>
                                        <p>${json[’
                                        collectieve_voornaamwoorden
                                        ’]}</p></div>‘);
            big_content.insertAdjacentHTML("beforeend", ‘<div><h4>
                                        Tussenwerpsels:</h4><
                                        p>${json[’
                                        tussenwerpsels’]}</p>
                                        </div>‘);

            small_content.insertAdjacentHTML("beforeend", ‘<div
                                        class="col-6"><h4>
                                        Stemfrequentie:</h4><
                                        p>${json[’pitch’]}</p
                                        ></div>‘);
            small_content.insertAdjacentHTML("beforeend", ‘<div
                                        class="col-6"><h4>
                                        Stemvolume:</h4><p>${
                                        json[’loudness’]}</p>
                                        </div>‘);

            loading_eigenschappen.classList.add(’hidden’);
            document.getElementById(’eigenschappen_content’).
                                        classList.remove(’
                                        hidden’);
        }).catch((error) => {
            document.getElementById(’errors’).classList.remove(’
                                        hidden’);
            loading_eigenschappen.classList.add(’hidden’);
            console.log(error);
        });
    } else {
        fetch(’http://127.0.0.1:5000/receive_normal’, {
            method: ’POST’,
            body: data_to_send
        }).then(response => {
            return response.json();
        }).then(json => {
            btn_record.disabled = false;
            pitch_normal = json[’pitch’];
            loudness_normal = json[’loudness’];
        }).catch((error) => {
            console.log(error)
        });
```

```
    }

}

function start_audio(text_after){
    audioChunks = [];
    rec.start();
    btn_record.classList.remove("btn-primary");
    btn_record.classList.add("btn-warning");
    btn_record.innerText = text_after
}
function stop_audio(innertext, text_after){
    if (innertext === "stop standaard opname") {
        document.getElementById('saved').classList.remove('hidden')
                                                ;
        picture.src = "/static/img/rusthuis.jpg";
        tekst_bij_foto.innerText = "Spreek in hoe je tegen deze
                                                bejaarde dame zou praten:
                                                ";
    } else {
        loading_eigenschappen.classList.remove('hidden');
        document.getElementById('eigenschappen_elderspeak').
                                                classList.remove('hidden'
                                                );
        btn_record.disabled = true;
    }
    rec.stop();
    btn_record.classList.remove("btn-warning");
    btn_record.classList.add("btn-primary");
    btn_record.innerText = text_after;
}

btn_record.addEventListener('click', () => {
    if (btn_record.innerText.toLowerCase() === "standaard audio
                                            opnemen"){
        start_audio("Stop standaard opname")
    } else if (btn_record.innerText.toLowerCase() === "stop
                                            standaard opname"){
        stop_audio("stop standaard opname", "Elderspeak audio
                                                opnemen")
        btn_record.disabled = true;
    } else if (btn_record.innerText.toLowerCase() === "elderspeak
                                            audio opnemen"){
        start_audio("Stop elderspeak audio opname")
    } else if (btn_record.innerText.toLowerCase() === "stop
                                            elderspeak audio opname"){
        stop_audio("stop elderspeak audio opname", "Saved")
    }

});

document.getElementById('reset').addEventListener('click', () => {
    location.reload()
})
```

Bibliografie

Beeckman, G. (2021). *Nursery Tone Monitor: softwarematige detectie van elderspeak* (eindwerk). Hogeschool Gent. Verkregen 28 november 2021, van https://catalogus.hogent.be/catalog/hog01:000739720

Brown, K. (2019, november 13). *What Is GitHub, and What Is It Used For?* Verkregen 23 april 2022, van https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/

Campens, J. (2021). Cursus Elderspeak.

Google Cloud. (2022, januari 1). *Natural Language AI*. Verkregen 22 april 2022, van https://cloud.google.com/natural-language

Jain, A. M. (2020, juni 5). *Confusion Matrix*. Verkregen 25 april 2022, van https://medium.com/@anishajain2910/confusion-matrix-30249214041d

Kemper, S., Finter-Urczyk, A., Ferrell, P., Harden, T., & Billington, C. (1998). Using elderspeak with older adults. *Discourse Processes*, *25*(1), 55–73. https://doi.org/10.1080/01638539809545020

Standaert, V. (2021). *Nursery tone monitor* (Eindwerk). Hogeschool Gent. Verkregen 28 november 2021, van https://catalogus.hogent.be/catalog/hog01:000739598

StatistiekVlaanderen. (2018). Home Bevolking Economie Levensomstandigheden Omgeving Overheid Rapporten EvenementenDe vergrijzing zet zich verder, 3. Verkregen 28 november 2021, van https://www.statistiekvlaanderen.be/nl/de-vergrijzing-zet-zich-verder

Tamara, J. (2022, april 14). *How to Host a Website: 4 Simple Steps and Why You Need Web Hosting*. Verkregen 23 april 2022, van https://www.hostinger.com/tutorials/how-to-host-a-website

Vieira, S. (2013, april 12). *Can I serve multiple clients using just Flask app.run() as standalone?* (StackOverflow, Red.). Verkregen 25 april 2022, van https://stackoverflow.

com/questions/14814201/can-i-serve-multiple-clients-using-just-flask-app-run-as-standalone

Wick, J. Y., & Zanni, G. R. (2007). The Irony of Elderspeak: Effective but Condescending. *The Consultant Pharmacist*, *22*(2), 175–178. https://doi.org/10.4140/tcp.n.2007.175

Williams, K. N. (2011, juni 9). *Communication in Elderly Care*. Bloomsbury UK. Verkregen 28 november 2021, van https://www.ebook.de/de/product/21099510/communication_in_elderly_care.html