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 mijn afstudeerrichting AI & *Data Engineering* was. Op die manier kon ik te weten komen of AI & *Data Engineering* eerder iets voor mij is, of toch eerder het programmeren zelf... (*spoiler*: het is het programmeren geworden.)

Het was best wel een interessant onderwerp! Aangezien er meer en meer vergrijzing komt, is er meer nood aan ouderenzorg. 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 hierdoor minder aan *elderspeak* zullen doen zodat ouderen niet behandeld worden als kleine kinderen. Hopelijk zal dit niet het geval zijn wanneer ik oud zal zijn en zorg zal nodig hebben...

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

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

Samenvatting

TODO: De äbstractöf samenvatting is een kernachtige (1 blz. voor een thesis) synthese van het document.

Deze aspecten moeten zeker aan bod komen: - Context: waarom is dit werk belangrijk? - Nood: waarom moest dit onderzocht worden? - Taak: wat heb je precies gedaan? - Object: wat staat in dit document geschreven? - Resultaat: wat was het resultaat? - Conclusie: wat is/zijn de belangrijkste conclusie(s)? - Perspectief: blijven er nog vragen open die in de toekomst nog kunnen onderzocht worden? Wat is een mogelijk vervolg voor jouw onderzoek?

LET OP! Een samenvatting is GEEN voorwoord!

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

Nu dit onderzocht werd, 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 is belangrijk zodat senioren minder het gevoel hebben dat ze als kleine kinderen behandeld worden. Op die manier worden zij in hun waarde gelaten tijdens het communiceren met zorgpersoneel ~~terwijl hun leven op zn einde loopt~~.

Om delen van *elderspeak* te herkennen in het spreken van een persoon, was de eerste stap het ijken van spraakherkenningssoftware met de kenmerken van *elderspeak*. Spraakher-

kenning gebruikt achterliggend een geavanceerd artificieel intelligentie-model en daarboven op wordt *natural language processing* of *NLP* uitgevoerd om de precieze te verbeteren. Die spraakherkenning kan alleen werken als er zo weinig mogelijk achtergrondlawaai is. Deze drie technieken worden beschreven in het literatuuronderzoek.

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

In het hoofdstuk daarna worden de resultaten van het testen beschreven. Zo werd er data verzameld en werden deze gegeven aan een Python-script dat automatisch *confusion matrixes* gebruikt om de accuraatheid van de applicatie weer te geven. Daarbij wordt er ook beschreven of alle *requirements* voldaan zijn om de applicatie af te leven.

Als voorlaatste hoofdstuk wordt er beschreven hoe dit eindwerk kan verder leven en wat een mogelijk vervolg kan zijn in een eventuele interdisciplinaire omgeving. Via GitHub kan de code gedeeld worden en kan de applicatie gemakkelijk geïnstalleerd worden op een andere computer. Zo kan de applicatie draaien op een computer, maar via een parameter kunnen ook andere apparaten aan de website.

De conclusie van deze bachelorproef is dat het mogelijk is om *elderspeak* te detecteren via AI, specifiek via spraakherkenning en Python-methoden. Dit is in een mooi jasje gegoten op een website zodat studenten in de zorg op dit fenomeen kunnen 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 hun 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 die de samenleving moet organiseren 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 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." 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 eerdere stappen rond 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?”

De inleiding moet de lezer net genoeg informatie verschaffen om het onderwerp te begrijpen en in te zien waarom de onderzoeksvraag de moeite waard is om te onderzoeken. In de inleiding ga je literatuurverwijzingen beperken, zodat de tekst vlot leesbaar blijft. Je kan de inleiding verder onderverdelen in secties als dit de tekst verduidelijkt. Zaken die aan bod kunnen komen in de inleiding (Pollefliet, 2011):

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

1.1 Probleemstelling

Dit onderzoek heeft als doel een meerwaarde te betekenen voor de oudere mensen in rusthuizen, homes, ziekenhuizen, maar ook nog de ouderen die zelfstandig thuis wonen. Deze mensen vinden het namelijk helemaal niet leuk om aangesproken te worden als kleine kinderen. Deze applicatie zal *elderspeak* herkennen en aangeven welke elementen die *elderspeak* opmaken, het model gevonden heeft. Het zijn dan vooral de verpleegkundigen, dokters, maar ook familieleden die zich bewust moeten zijn hoe ze tegen ouderen praten.

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

1.2 Onderzoeksvraag

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 alleen dit beantwoorden zal uiteraard niet genoeg zijn. Het beantwoorden van de volgende deelvragen zullen wel een duidelijker en uitgebreider antwoord geven op de algemene probleemstelling:

- 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 om zo deze applicatie in gebruik te nemen voor zorgkundigen?

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

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 gewoon te praten zoals je zou doen tegen je vrienden. Nadien wordt er gevraagd om te praten zoals je zou doen bij slechthorende senioren in een rusthuis. Om dat gevoel te versterken zal er een foto getoond van iemand uit een rusthuis. De applicatie analyseert dan de audiosamples en geeft aan welke kenmerken er aanwezig waren van *elderspeak*. Die kenmerken van *elderspeak* en waarom het model 'denkt' dat die eigenschappen aanwezig zijn, zijn vergaard in het literatuuronderzoek.

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

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

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

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

In Hoofdstuk 4 worden de resultaten beschreven van deze bachelorproef zoals de data die verzameld zijn, de resultaten na het testen, hoe goed een rollenspel werkt en of 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, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2. Stand van zaken

Dit hoofdstuk bevat je literatuurstudie. De inhoud gaat verder op de inleiding, maar zal het onderwerp van de bachelorproef *diepgaand* uitspitten. De bedoeling is dat de lezer na lezing van dit hoofdstuk helemaal op de hoogte is van de huidige stand van zaken (state-of-the-art) in het onderzoeksdomein. Iemand die niet vertrouwd is met het onderwerp, weet nu voldoende om de rest van het verhaal te kunnen volgen, zonder dat die er nog andere informatie moet over opzoeken (Pollefliet, 2011).

Je verwijst bij elke bewering die je doet, vakterm die je introduceert, enz. naar je bronnen. In LaTeX kan dat met het commando `\textcite{}` of `\autocite{}`. Als argument van het commando geef je de "sleutel" van een "record" in een bibliografische databank in het BibLaTeX-formaat (een tekstbestand). Als je expliciet naar de auteur verwijst in de zin, gebruik je `\textcite{}`. Soms wil je de auteur niet expliciet vernoemen, dan gebruik je `\autocite{}`. In de volgende paragraaf een voorbeeld van elk.

Knuth (1998) schreef een van de standaardwerken over sorteer- en zoekalgoritmen. Experten zijn het erover eens dat cloud computing een interessante opportuniteit vormen, zowel voor gebruikers als voor dienstverleners op vlak van informatietechnologie (Creeger, 2009).

2.1 Verkennend onderzoek

"Kan *elderspeak* gedetecteerd worden door Artificiële Intelligentie en kan dit toegepast worden in de praktijk?", is de centrale onderzoeksvraag, maar daarvoor moeten we twee begrippen goed uitleggen en begrijpen om te kunnen staven of dit wel degelijk mogelijk is.

Er zal dus eerst beschreven worden wat *elderspeak* precies is. Waarom vinden ouderen dat niet leuk? Wat zijn de eigenschappen en hoe kan je het voorkomen? Daarnaast moeten we ook begrijpen wat Artificiële Intelligentie is met daarbij alle verschillende types, vormen en maten ervan. De reden hiervoor is dat er verstaan moet worden welke en hoe AI gebruikt werd in het eindresultaat.

2.2 Elderspeak

2.2.1 Wat is *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 het volgende: "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." Dus het gaat over stereotype communicatie tussen ouderen en jongeren zoals het vereenvoudigen van zinnen en het toevoegen van herhalingen.

2.2.2 Wat zijn de kenmerken?

Deze kenmerken komen overeen met een communicatiestijl die men hanteert wanneer men tegen (afhankelijke) kinderen praat. Vandaar dat *elderspeak* ook wel als *secondary baby* talk wordt benoemd. Campens (2021) volgende 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

Tenslotte is het belangrijk om te weten 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).

2.2.3 Wat zijn de tips om *elderspeak* te voorkomen?

De onderstaande tips zijn enkele van de tips die Wick en Zanni (2007) meegeven ter voorkoming van *elderspeak*:

- 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 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 zinnen en 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).

2.3 Artificiële Intelligentie

2.3.1 Wat is AI?

Er zijn bijzonder veel beschrijvingen van wat AI is, maar alleen uit verschillende bronnen kan je een accuraat beeld krijgen van wat AI precies is.

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.

2.3.2 Welke vormen zijn van AI?

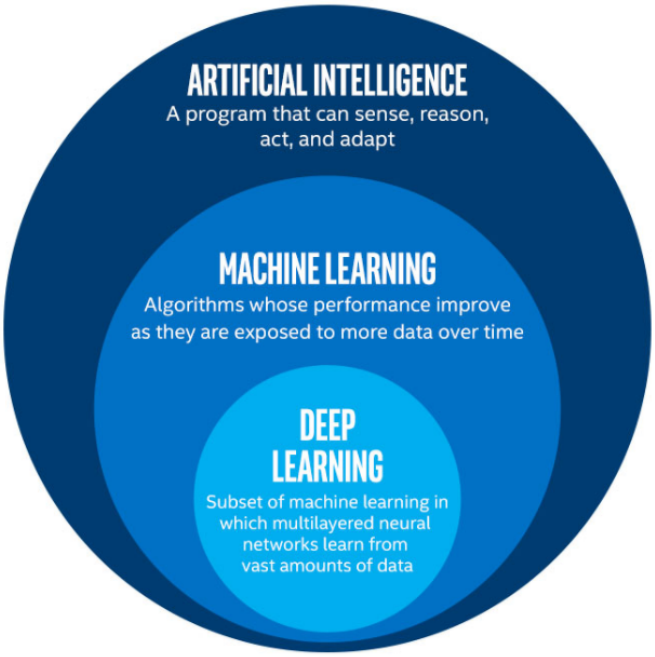
Artificiële Intelligentie is een verzamelnaam voor twee soorten algoritmes. Zo is er enerzijds *machine learning* en anderzijds *deep learning* (Kavlakoglu, 2020). Dit kan mooi geïllustreerd worden a.d.h.v. figuur 2.1. Daarop kan er afgeleid worden dat Artificiële Intelligentie een verzamelnaam is. Specifieker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

2.3.3 *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 types namelijk *supervised learning* of gesuperviseerd leren, *unsupervised learning* of leren zonder toezicht en als laatste type is er *reinforcement learning* of leren door bekrachtiging.

Supervised learning

Een defenitie gegeven door Lievens (2021) over *supervised learning* 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



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

een (nieuwe) input te voorspellen. Wanneer het label een reëel getal is, spreekt men van 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.

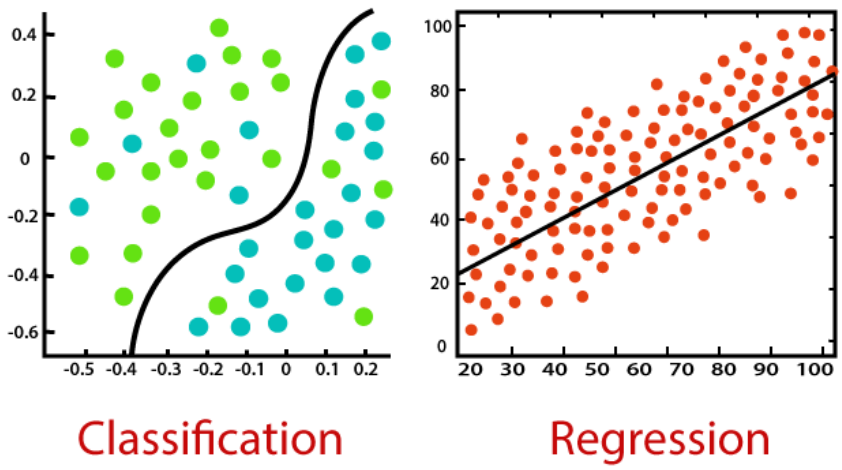
Voorbeelden van het classificatieprobleem zijn spamdetectie, nummerherkenning of diabetesdetectie. Elk hebben ze een beperkt aantal output klassen. Bij spamdetectie zijn er twee vooraf gedefinieerde klassen: spam en geen spam. Bij het herkennen van nummers zijn er 10 klassen: 0 t.e.m. 10 en bij diabetesdetectie kunnen er drie klassen aanwezig zijn namelijk geen diabetes, diabetes en pre-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 voorbeelden van clustering is marktsegmentatie waarbij klanten worden opgedeeld in



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

verschillende segmenten zoals trouwe klant, mogelijke vertrekkende klant, niet tevreden klant etc. Op basis van aankoopdata, tijdstippen en online activiteit kan men de klanten gaan indelen in clusters. Die informatie kan dan weer gebruikt worden om bepaalde clusters meer informatie te geven of korting te geven. Er zijn algoritmen die fraude opsporen op een website die anders gedrag dan normaal gaan detecteren, wat dan een voorbeeld van anomalie detectie is. Bij PCA worden de data gereduceerd zodat de minder relevante data afneemt in de dataset, maar de nodige data juist duidelijker maakt.

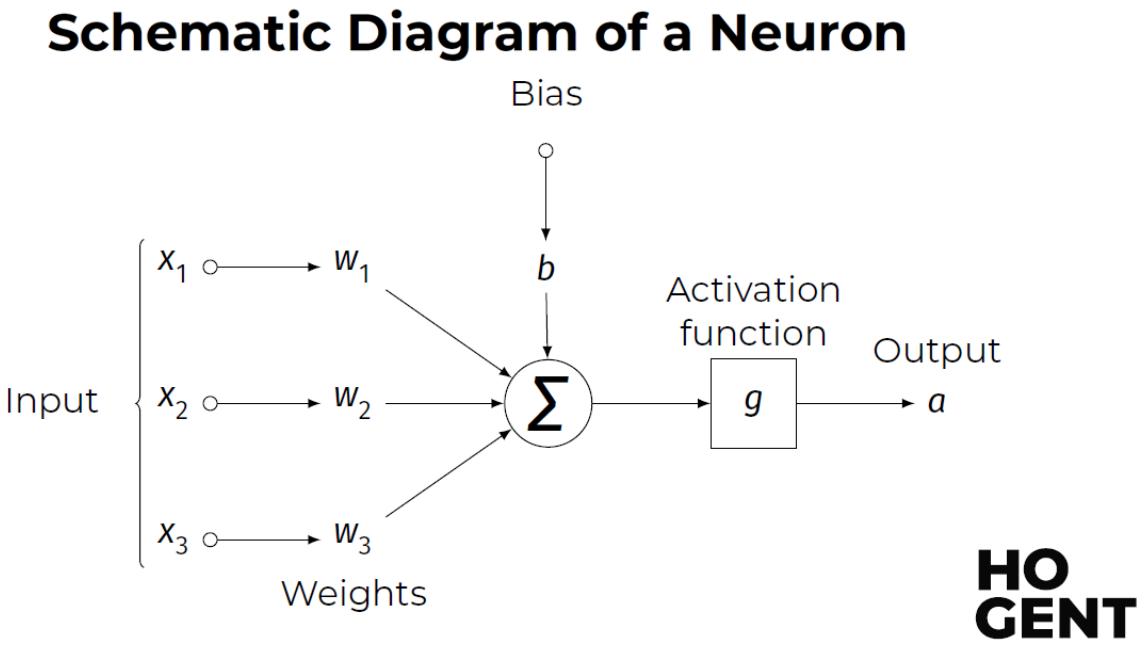
Reinforcement learning

Lievens (2021) beschrijft leren door bekrachtiging door een techniek die niet echt een dataset gebruikt, maar eerder via een identiteit is die leeft in een (on)bekende wereld. Die identiteit krijgt dan beloningssignalen. De opdracht is dan om uit te zoeken wat de regels zijn die leiden tot een grote beloning.

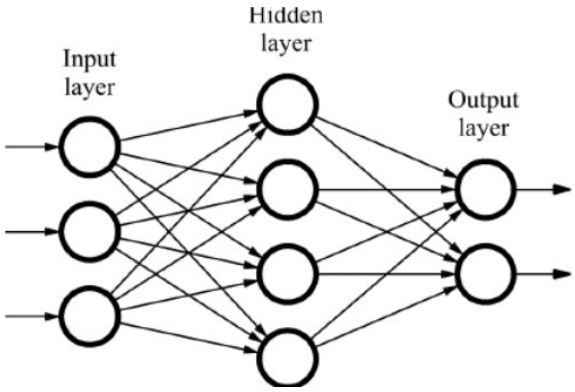
Het bekendste voorbeeld van *reinforcement learning* is zelfrijdende wagens waarbij het model alles zelf moet aanleren zoals het remmen en gas geven in welke hoeveelheid en hoe er gestuurd moet worden. Dit is een lang en iteratief proces waarbij het model dus de hele tijd zichzelf bijstuurt en bijleert van wat het net gezien heeft.

2.3.4 *Deep Learning*

Deep learning is een onderdeel van *machine learning* die neurale netwerken bevat. “Een artificieel neuraal netwerk bestaat uit een groot aantal eenvoudige rekende eenheden, units of neuronen die de volgende eigenschap heeft: ‘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*.



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



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

De systematische voorstelling van een neuron is te vinden in figuur 2.3. Een volledig neuraal netwerk kan er uit zien 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 die 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.

2.4 AI-technieken die gebruikt worden om elderspeak te detecteren

2.4.1 NLP of *natural language processing*

Om te kunnen detecteren of men verkleinwoorden, troetelnamen, collectieve voornaamwoorden of 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 kort NLP. Maar wat is NLP precies?

Natural language processing of vertaald "natuurlijke taalverwerking" zit in de tak van de kunstmatige intelligentie. NLP zit een stuk in *machine learning* en een stukje in *deep learning*. (Kleinings, 2022) De visuele voorstelling van die verhoudingen zijn 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.

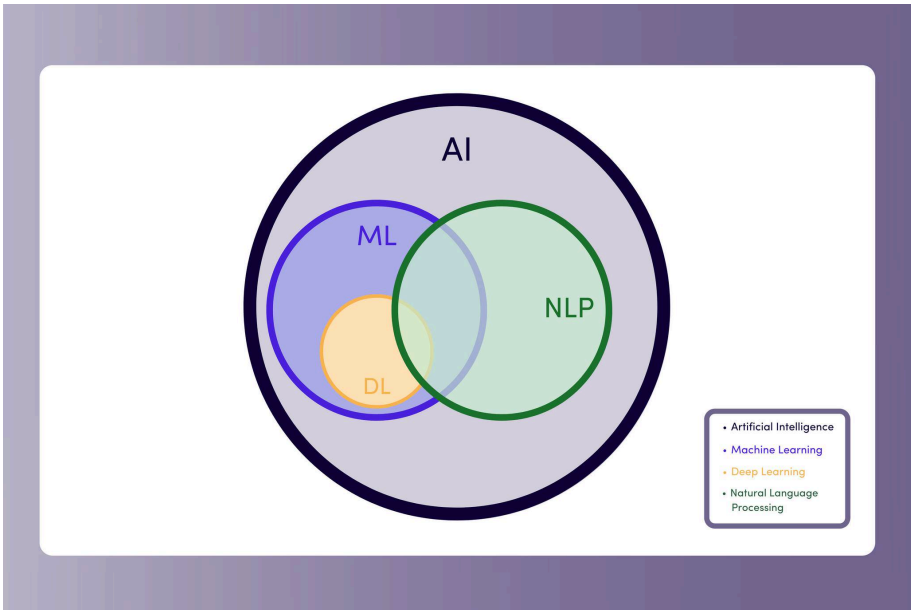
NLP combineert computationele linguïstiek - op regels gebaseerde modellering van menselijke taal - met statistische, *machine learning*- en *deep learning*-modellen. Samen stellen deze technologieën computers in staat menselijke taal - in de vorm van tekst of spraakgegevens - te verwerken en de volledige betekenis ervan te "begrijpen", compleet met de 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 vertalen, reageren op gesproken opdrachten en grote hoeveelheden tekst kan samengevat worden, zelfs in realtime. Enkele voorbeelden van NLP-systemen in het dagelijkse leven zijn: spraakgestuurde GPS-systemen, digitale assistenten, spraak-naar-tekst dicteersoftware in bijvoorbeeld de smartphone 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 vandaag de dag, die elke hun aparte 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



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

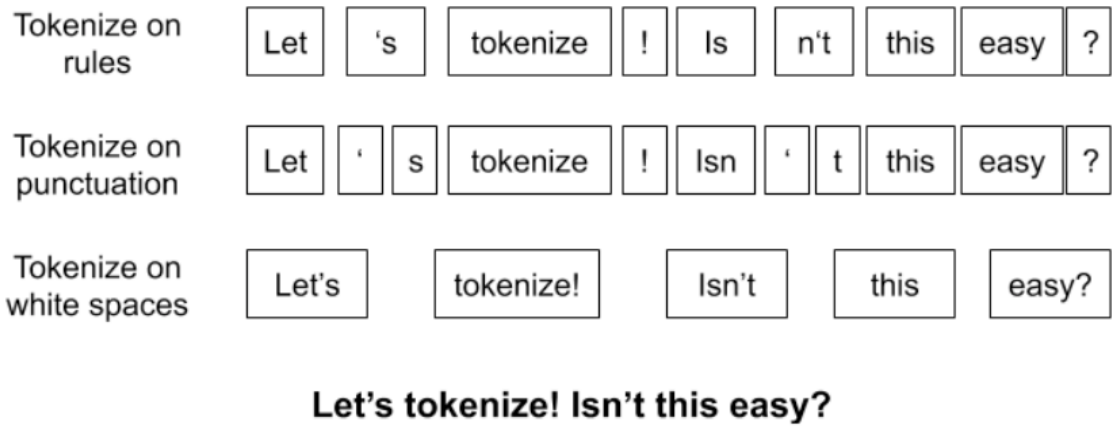
bruikbare, niet dubbelzinnige informatie die de computer dan kan begrijpen en er verder mee kan rekenen. Dit stukje wordt de *pre-processing* genoemd.

Bovendien speelt de context ook vaak parten. *Natural Language Processing* werkt 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 sarcasme of ironie in hun taal steken zodat het een andere betekenis krijgt. Om deze andere betekenis uit een tekst te halen is er heel veel moeite voor nodig.

Hoe werkt 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: morfologie, syntaxis, semantiek en pragmatiek, schrijft Kleinings (2022), wat te lezen is op een *low-code/no-code* platform genaamd Levity die dus veel kennis hebben 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. Denk hierbij aan het voorbeeld met de bank.



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

Om de syntaxis aan te leren zijn er twee manieren. Enerzijds kunnen er woorden gelabeld worden waarbij er gezegd 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 met dezelfde logica.

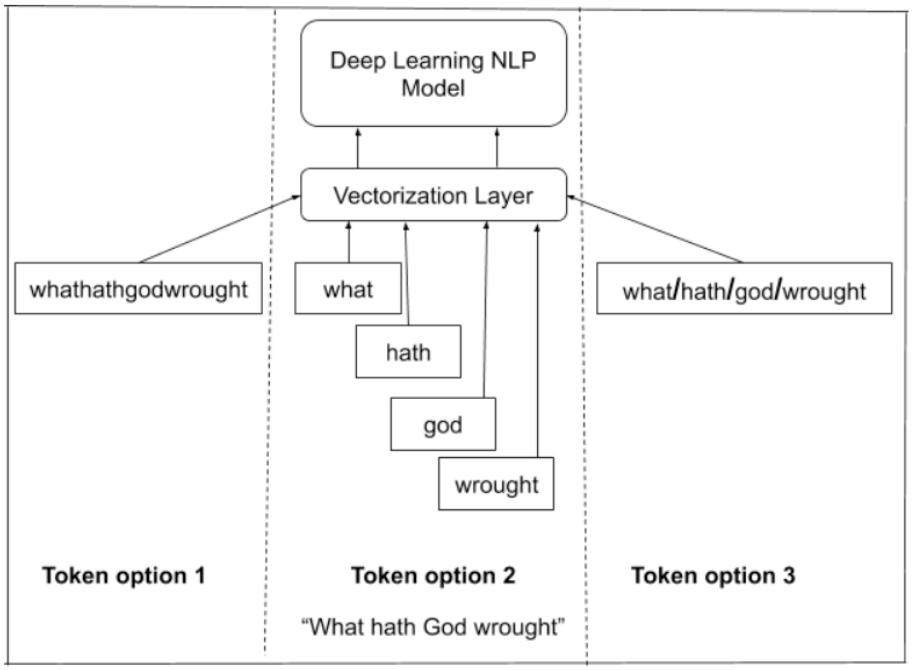
Daarnaast is het ook aan te raden dat stopwoorden verwijderd worden als het model de tekst wil begrijpen. Stopwoorden zoals “een, de, euh, inderdaad, echt, oké, eigenlijk, allez, etc.” moeten verwijderd worden. Daarbij aansluitend zijn er twee andere methoden. Enerzijds lemmatiseren waarbij werkwoorden naar de tegenwoordige tijd veranderd worden, en anderzijds stemming, waarbij er prefixen of affixen verwijderd worden, ook nodig om de conversie te maken. Een voorbeelden van het converteren 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.

2.4.2 Filteren van achtergrond lawaai

Volgens Jung e.a. (2021) helpen *convolutional neural networks* of CNN’s - 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 - bijzonder veel om achtergrondlawaai weg te filteren.

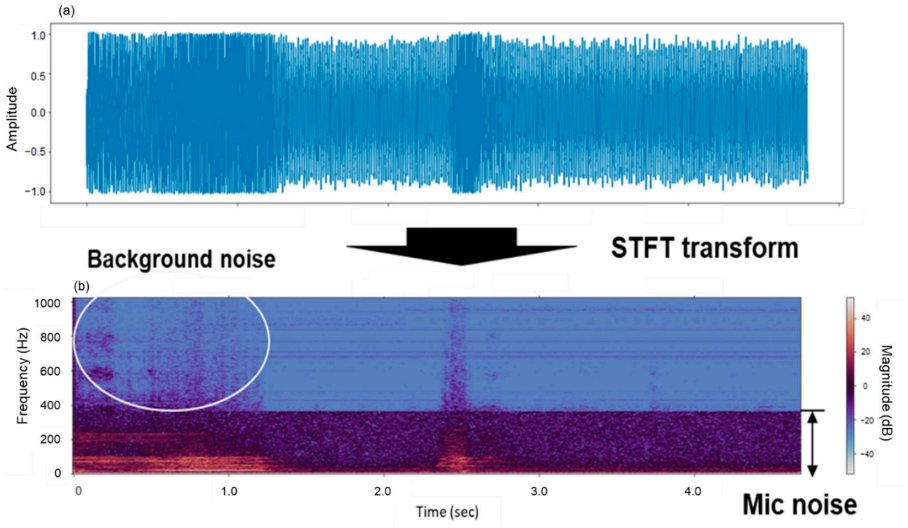
Om achtergrond lawaai weg te werken zijn AI gebaseerde filters zoals de *short-time Fourier transform* of STFT ideaal. De STFT-filter verbetert in het algemeen de kwaliteit van



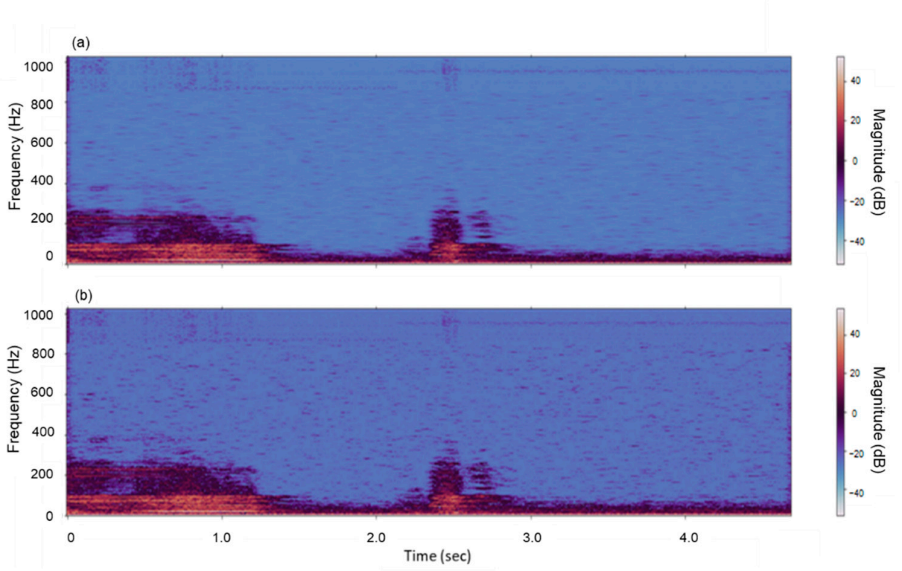
Figuur 2.7: Verhouding *deep learning* NLP model met de layers en de verschillende *tokens*. (Horan, 2020)

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 tijdsignaal 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 als 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) gebruikte 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 dat uitgeschreven is in de stand van zaken over wat *elderspeak* en *nursery tone* precies zijn, is het doel van deze bachelorproef om een website te ontwikkelen waarmee ~~er~~ *elderspeak* kan gedetecteerd worden via een detector. In het literatuur werd er ook theoretisch beschreven wat AI is en welke types er zijn, wat en hoe *natural language processing* werkt en hoe men achtergrond lawaai filtert.

Alle eigenschappen van *elderspeak* worden herkend a.d.h.v. Python-bibliotheken en niet op basis van een zelf gemaakt AI-model. De reden hiervoor is dat men het warm water niet opnieuw moet uitvinden. Zo staafde Beeckman (2021) het volgende over zelf een AI-model te maken voor deze opstelling: “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.”. Door deze bewering is de denkpiste in combinatie met de kleine dataset, die in dit onderzoek verzameld werd, over zelf een AI-model opstellen al snel van tafel geveegd. Ook de promotor Van Boven haalde aan dat er genoeg Python-bibliotheken ter beschikking waren 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 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. Daardoor zal er in Hoofdstuk 5 een deeltje aanbod komen over waarom het belangrijk is dat code gedeeld wordt.

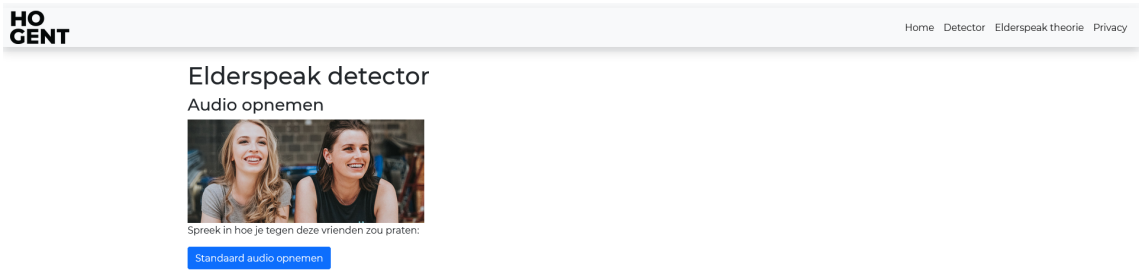
Figuur 3.1: Home pagina website

3.1 Berekeningen in *back-end*

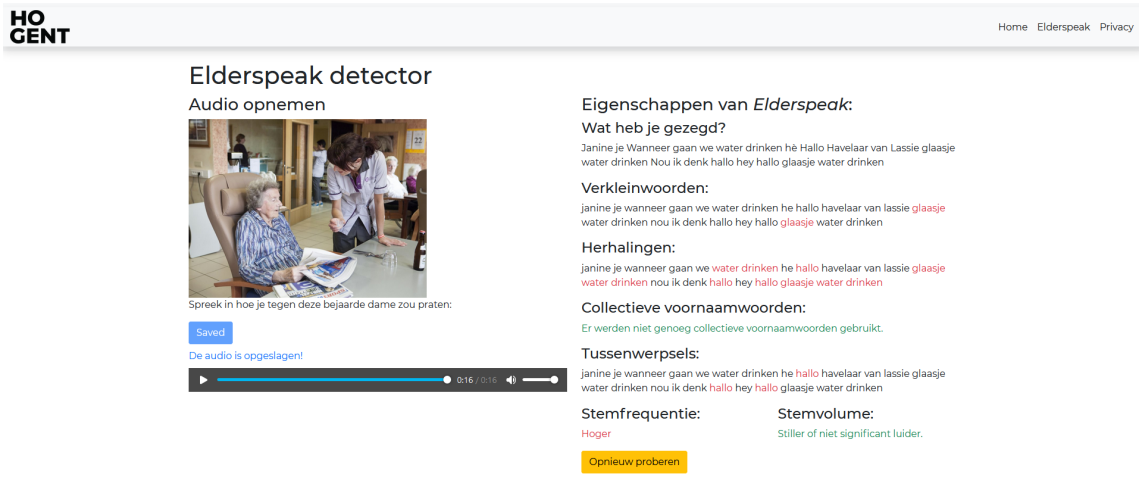
Deze voorbeeldapplicatie, geschreven in Python en gebruikmakend van het *micro-framework* Flask, is een webapplicatie die verschillende webpaginas bevat. De code van de volledige Flask-applicatie is te vinden in bijlage B.1. Naast de inleidende pagina, zie schermafbeelding 3.1, bevat deze ook een detector. Voor het begin zie schermafbeelding 3.3 en na het analyseren ziet de webpagina er het volgende 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 kunnen mensen, maar specifiek studenten in de zorg, actief en passief leren wat *elderspeak* precies is. Enerzijds kunnen ze de eigenschappen te weten komen door zelf actief stukjes spraak op te nemen. Die wordt dan geanalyseerd zodat men kan zien welke eigenschappen er aanwezig waren. Anderzijds kunnen ze passief bekijken wat de eigenschappen zijn van *elderspeak* en hoe men dat kan voorkomen.

De applicatie werd zo gemaakt waarbij men eerst twee jonge vrouwen ziet, een foto van HOGENT voor copyrightrechten, waarbij er gevraagd wordt om spraak in te spreken zoals men zou doen tegen hen. Dat geluid wordt dan geanalyseerd is voor de eigenschappen van toonhoogte en stemvolume. Daarna zal er een foto van een oudere vrouw getoond wordt in het rusthuis waarbij er gevraagd wordt om tegen haar te spreken. De twee parameters van daarvoor worden dan meegegeven zodat de correcte conclusies dan getrokken kunnen worden.

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 een groot deel van de applicatie is het herkennen van de spraak die wordt opgenomen op de website. Uiteraard moet dit niet van nul gemaakt worden, maar kan er gebruik gemaakt worden van een bestaande Python-bibliotheek. Standaert (2021) bestudeerde in zijn bachelorproef afgelopen jaar dat de beste optie is om de Google Speech Recognition-API 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 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 weten wat er precies gezegd is geweest (Google Cloud, 2022). Hoe dit precies in elkaar zit is te lezen in het literatuuronderzoek, namelijk in Hoofdstuk 2.

Bij de spraakherkenning kwam er wel een groot probleem de kop op steken. De Google *Speech Recognition*-API kon alleen overweg met *wav*- en *flac*-bestanden én kan maar een bepaalde periode, ongeveer 2 tot 3 minuten, gratis herkennen. Het probleem is dus dat de audio gecapteerd is in een *mp3*-formaat. Hiervoor was het 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 te vinden tot in detail 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 wat aanpassen aan de geluidsbron zodat het beter de klanken kan horen 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 mee gegeven 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. Met deze grotere methode is de basis gelegd voor andere methoden die steunen op de tekst die herkend werd.

3.1.2 Verkleinwoorden

Voor de methode om verkleinwoorden te herkennen is er gefundeerd op de methode van Standaert (2021). 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 voorbeelden die 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 evenals gebruikt gemaakt van de methode die Standaert (2021) geschreven had. Daarbij wordt er bijgehouden wat de vorige 25 woorden waren en bij herhalingen worden de woorden die herhaald worden bewaard in een lijst. Die zullen we later gebruiken om een mooie weergave te maken. 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”. Om het voorbeeld te verduidelijken zijn de volgende zinnen gegeven: “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 een keer voorkomt, dan beschouwt de applicatie dit dat het voorkomt. Dit is zo ingesteld omdat het niet mag worden weergegeven wanneer er iemand een keer 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 gebruikt werden.

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* weten en moet daarna ook kritisch zijn over het resultaat en dit kan via 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: “o”, “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 bij de methode van de collectieve voornaamwoorden die te vinden is in 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 zal praten, zal de andere persoon direct voelen dat hij/zij behandeld wordt als een kind. Het is dan ook belangrijk dat deze functie goed werkt zodat de

gebruikers direct weten wanneer ze (on)bewust hoger praten.

Deze methode werd al opgesteld door Standaert (2021) in zijn eindwerk van het vorige jaar. Wat hij berekende was de gemiddelde toonhoogte, uitgedrukt in Hz, van het gegeven audiobestand. Wat er aan deze applicatie toegevoegd is, is het berekenen of de toonhoogte hoger ligt bij de casus met de oudere vrouw dan de casus bij de twee jonger vrouwen. Hoe dit gerealiseerd werd in de code is te vinden in bijlage B.7.

3.1.7 Stemvolume

De laatste eigenschap die geanalyseerd wordt in de applicatie is om te controleren 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 significant verder van de microfoon staat, zal de applicatie dit detecteren dat het niet luider zal zijn. Daarnaast is een gevoelige hoeveelheid van de oudere mensen slechthorend, waardoor men wel luider moet praten. Ondanks deze twee zaken vond ik het toch belangrijk om deze eigenschap te implementeren 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.

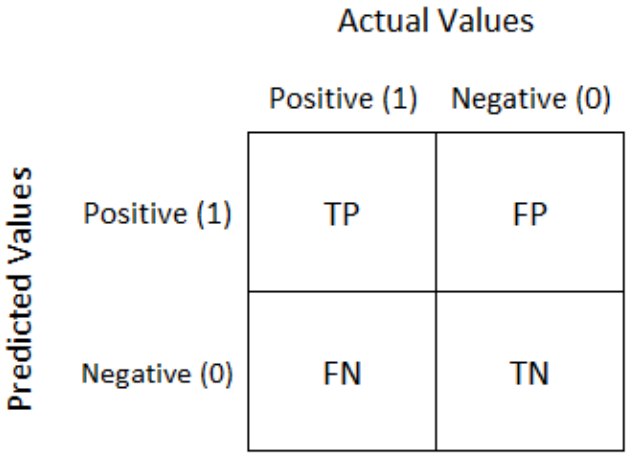
3.2 *Front-end*

3.2.1 Geluid opnemen

Het geluid opnemen gebeurt volledig aan de kant van de *client* of de gebruiker. Wanneer er op de knop geduwd 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 getond 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 terug gestuurd naar de *client*. Daar worden de resultaten ingevuld in de voorziene html-stukken.



Figuur 3.5: Confusion Matrix (Jain, 2020)

3.3 Testen

Om een objectief beeld te kunnen verkrijgen hoe goed de applicatie werkt, 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 twee semester verzameld. Eens de data opgeslagen was werd alle data beluisterd en handmatig gelabeld.

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

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

4. Resultaten

4.1 Resultaten van de data die verzameld is

Via het formulier van "JotForm" waren er 13 inzendingen. Dat lijkt niet veel, maar er moet wel geweten zijn dat dit niet zomaar snel een vragenlijst invullen is. Er werd telkens gevraagd om een eigenschappen van *elderspeak* na te bootsen door een geluidsfragment in te spreken. Dit is veel meer werk waardoor mensen sneller afhaakten.

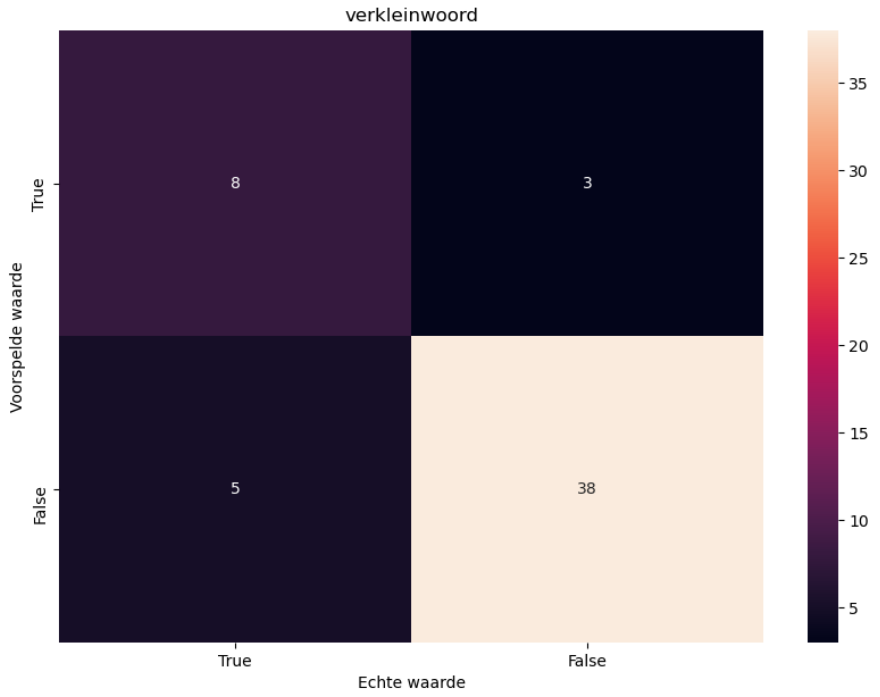
Toch gaven deze 13 personen 54 audiobestanden ter beschikking. Op die geluidsbestanden kon er gecontroleerd worden of er *elderspeak* aanwezig bij was.

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

Meer hierover wordt beschreven in het vervolg verhaal, namelijk 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

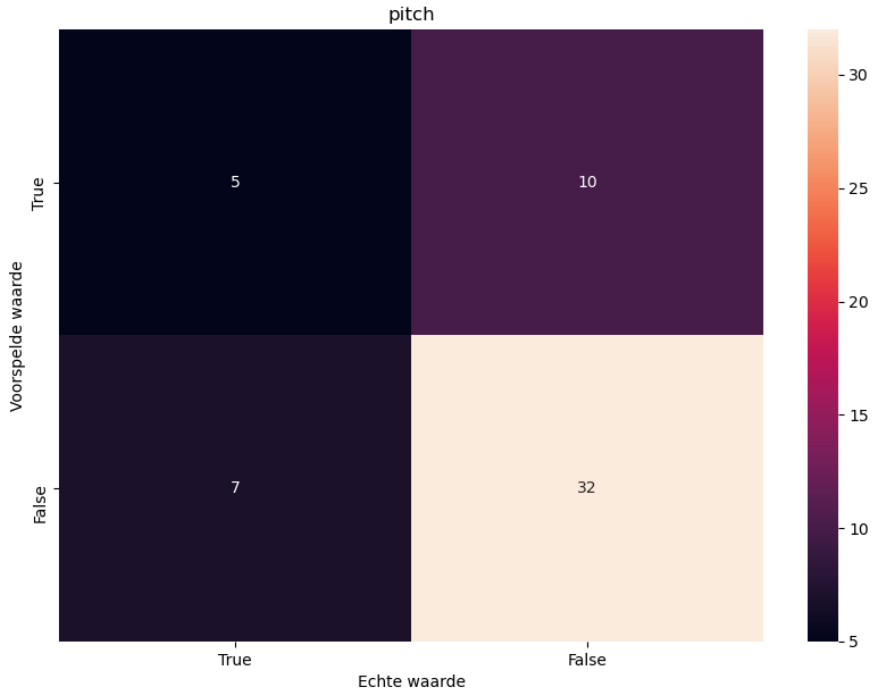
De testgevallen van de verkleinwoorden zijn te vinden in *confusion matrix* 4.1. Daarbij is te zien dat er 8 correcte positieven zijn, 3 vals positieven, 5 valse negatieve en 38 correctie negatieven. Hieruit kunnen we afleiden dat de applicatie goed de verkleinwoorden weet te vinden.

4.2.2 Toonhoogte

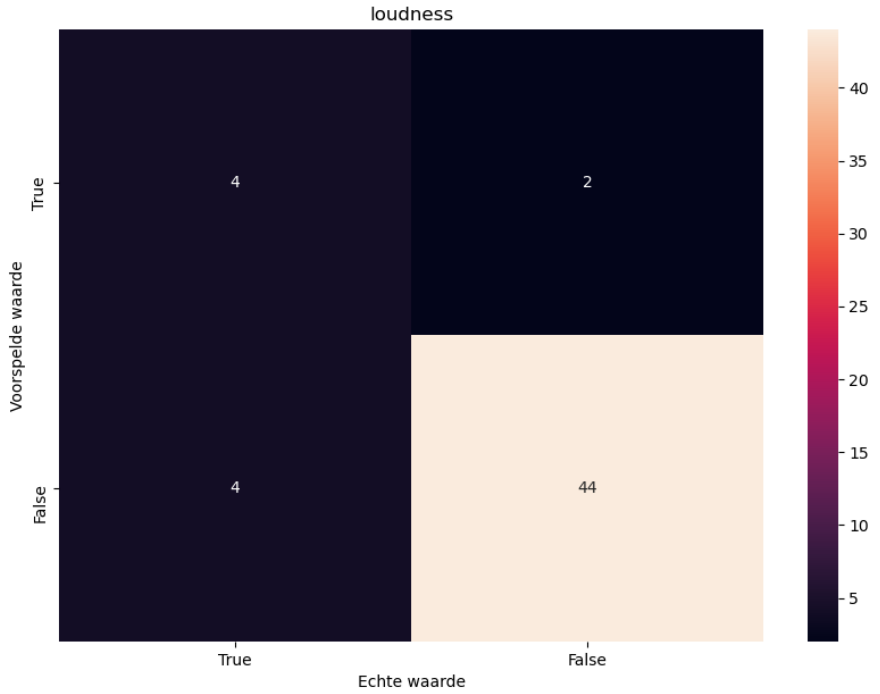
De *confusion matrix* van de testgevallen van de toonhoogte is te vinden in figuur 4.2. Daar is te zien dat er 5 correcte positieven zijn, 10 vals positieven, 7 valse negatieve en 32 correctie negatieven. Hieruit kunnen we afleiden dat de applicatie een beetje werkt voor de toonhoogte, maar dat hij er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

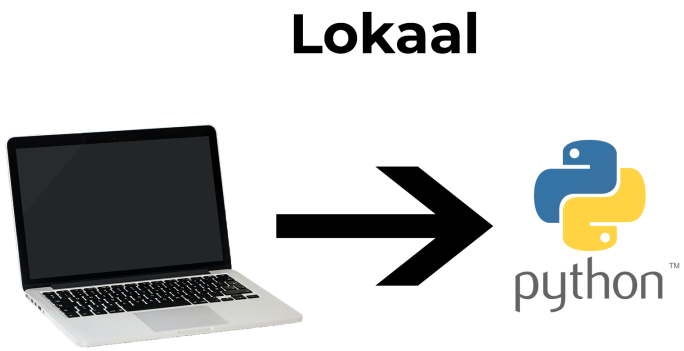
In dit hoofdstuk zullen mogelijke vervolgen en toepassingen van deze bachelorproef beschreven worden. Hoe kunnen anderen toegang krijgen tot de programmeercode, de website raadplegen of hoe kan de *webserver* geïnstalleerd worden binnen het HOGENT-netwerk. Ook zal er besproken worden hoe de data beter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

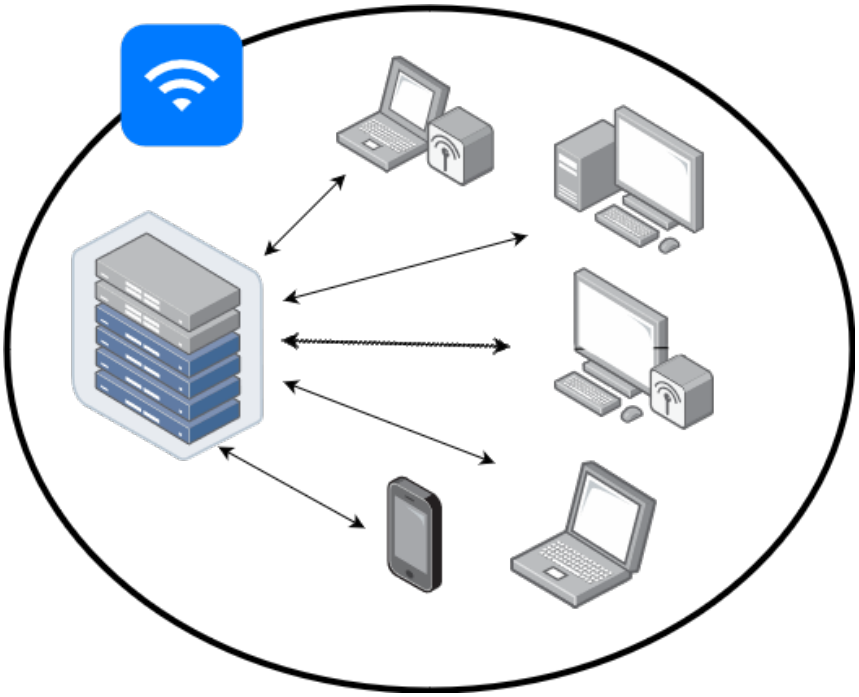
uit te voeren zullen alle Python-bibliotheken geïnstalleerd worden die nodig zijn om het project uit te voeren. Op Windows moet er wel nog een extra stap gebeuren en dat is FFmpeg apart installeren. De stappen daarvan staan beschreven op de volgende website: https://www.wikihow.com/Install-FFmpeg-on-Windows.

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



Figuur 5.3: Lokaal netwerk

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 de 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

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.

5.2.3 Gratis 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.

Mogelijkheden

- Eens opgezet is het altijd beschikbaar van overal
- Wanneer er veel verbindingen en berekeningen nodig zijn, zal Microsoft de servers *upgraden* zodat ze alles kunnen bolwerken
- 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

TODO: Trek een duidelijke conclusie, in de vorm van een antwoord op de onderzoeksvra(a)g(en). Wat was jouw bijdrage aan het onderzoeksdomein en hoe biedt dit meerwaarde aan het vakgebied/doelgroep? Reflecteer kritisch over het resultaat. In Engelse teksten wordt deze sectie "Discussion" genoemd. Had je deze uitkomst verwacht? Zijn er zaken die nog niet duidelijk zijn? Heeft het onderzoek geleid tot nieuwe vragen die uitnodigen tot verder onderzoek?

- 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 om zo deze applicatie in gebruik te nemen voor zorgkundigen?

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

Bansal, H. (2019, juli 19). *How to get the Perfect start in AI ML as Newbie?? Learn the Art in just 5 mins!* Verkregen 18 maart 2022, van https://becominghuman.ai/how-to-get-the-perfect-start-in-ai-ml-as-newbie-learn-the-art-in-just-5-mins-cba28d2705e4

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.

Creeger, M. (2009). CTO Roundtable: Cloud Computing. *Communications of the ACM*, *52*(8), 50–56.

Europees Parlement. (2020, september 4). *Wat is artificiële intelligentie en hoe wordt het gebruikt? Twitter LinkedIn Whatsapp Artificiële of kunstmatige intelligentie (AI) staat op het punt om voorloper te worden als technologie van de toekomst. Maar wat is AI precies en hoe beïnvloedt het ons leven nu al?* Europees Parlement. Verkregen 18 maart 2022, van https://www.europarl.europa.eu/news/nl/headlines/society/20200827STO85804/wat-is-artificiele-intelligentie-en-hoe-wordt-het-gebruikt

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

Horan, C. (2020, januari 28). *Tokenizers: How machines read: We will cover often-overlooked concepts vital to NLP, such as Byte Pair Encoding, and discuss how understanding them leads to better models.* FloydHub. Verkregen 28 maart 2022, van https://blog.floydhub.com/tokenization-nlp/

IBM Cloud Education. (2021, juli 2). *Natural Language Processing (NLP)*. IBM. Verkregen 28 maart 2022, van https://www.ibm.com/cloud/learn/natural-language-processing

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

JavaTpoint. (2021, januari 1). *Regression vs. Classification in Machine Learning*. https://www.javatpoint.com/regression-vs-classification-in-machine-learning

Jung, D.-H., Kim, N. Y., Moon, S. H., Jhin, C., Kim, H.-J., Yang, J.-S., Kim, H. S., Lee, T. S., Lee, J. Y., & Park, S. H. (2021). Deep Learning-Based Cattle Vocal Classification Model and Real-Time Livestock Monitoring System with Noise Filtering. *Animals*, *11*(2), 357. https://doi.org/10.3390/ani11020357

Kavlakoglu, E. (2020, mei 27). *AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: Whats the Difference?* IBM. Verkregen 18 maart 2022, van https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks

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

Kleinings, H. (2022, februari 17). *How Natural Language Processing works in 2022*. Verkregen 28 maart 2022, van https://levity.ai/blog/how-natural-language-processing-works

Knuth, D. E. (1998). *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. Addison Wesley Longman Publishing Co., Inc.

Lievens, S. (2021, september 27). Lecture notes in Distributed Databases - Machine Learning. *HoGent*.

Oracle. (2014, april 24). *Wat is AI? Meer informatie over kunstmatige intelligentie*. https://www.oracle.com/nl/artificial-intelligence/what-is-ai/

Pollefliet, L. (2011). *Schrijven van verslag tot eindwerk: do's en don'ts*. Academia Press.

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