Titel van de bachelorproef.

Optionele ondertitel.

Arthur Libberecht.

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

Promotor: Stijn Lievens
Co-promotor: Jana Van Damme
Academiejaar: 2022–2023
Eerste examenperiode

Departement IT en Digitale Innovatie .

HO
GENT

Woord vooraf

Deze bachelorproef is geschreven in het kader van mijn opleiding Bachelor in de Toegepaste Informatica aan de Hogeschool Gent. Dit onderzoek is toegewezen door docent Lena De Mol en sprak mij meteen aan vanwege het innovatieve idee van Zorglab 360°. De mogelijkheid om mee te helpen aan een nieuwe vorm van therapie leek mij dan ook zeer interessant.

Dit onderzoek had echter niet tot stand kunnen komen zonder de hulp van verschillende mensen, waarvoor ik enorm dankbaar ben.

Allereerst wil ik mijn promotor, Stijn Lievens, bedanken voor de waardevolle feedback die ik heb ontvangen tijdens het schrijven van deze bachelorproef. Hij heeft mij altijd goed advies gegeven en heeft ook bijgedragen aan de sturing van mijn onderzoek.

Daarnaast wil ik ook mijn co-promotor, Jana Van Damme, bedanken voor het introduceren van het onderzoek en haar inspanningen om het onderzoek te begrijpen. Ondanks dat zij misschien niet over de technische expertise beschikt, heeft zij altijd haar best gedaan om alles zo goed mogelijk te begrijpen en mij duidelijk uit te leggen wat er onderzocht moest worden.

Tot slot wil ik mijn ouders bedanken voor hun financiële steun bij het volgen van deze studierichting.

Ik wens u veel leesplezier toe en hoop dat dit onderzoek leidt tot iets innovatiefs.

Samenvatting

Inhoudsopgave

Lijst van figuren

1

Inleiding

Stotteren is een veelvoorkomende spraakstoornis die wereldwijd effect heeft op het dagelijks leven van mensen. Mensen die stotteren hebben vaak moeite met communiceren in sociale situaties, wat kan leiden tot gevoelens van angst, schaamte en isolatie. Helaas blijken traditionele behandelingen voor stotteren vaak beperkte doeltreffendheid te hebben en maken ze weinig tot geen gebruik van nieuwe technologieën.
Gelukkig zijn er organisaties zoals Zorglab 360° van HOGENT, die zich richten op het ontwikkelen van innovatieve oplossingen voor de zorgsector in Vlaanderen. Zij hebben een nieuwe vorm van stottertherapie ontwikkeld die gebruik maakt van virtual reality. Momenteel is deze behandeling nog in ontwikkeling. Het doel is om de therapie verder te automatiseren door gebruik te maken van Automatic Speech Recognition modellen, zoals bijvoorbeeld Kaldi of Wav2Vec. Dit kan een veilige en gecontroleerde omgeving bieden waarin de patiënten hun spraakvaardigheden kunnen verbeteren en hun zelfvertrouwen opbouwen.

1.1. Probleemstelling

Op dit moment werkt de stottertherapie van Zorglab 360° nog niet zoals bedoeld. Tijdens het afspelen van een video van een realistische situatie stopt de video op het moment dat er interactie nodig is, bijvoorbeeld wanneer de stotteraar antwoord moet geven op een gestelde vraag. Om dit probleem op te lossen is het noodzakelijk om gebruik te maken van Automatic Speech Recognition (ASR) modellen. Het vinden van het juiste ASR model met de juiste toevoegingen is echter een uitdaging. Omdat deze situatie zeer specifiek is, moet er onderzocht worden hoe dit het beste aangepakt kan worden om de therapie te optimaliseren.
Zo heb je dan het Kaldi speech recognition toolkit, dat gebruikt kan worden om de automatisering van de stottertherapie van Zoglab 360° verder te verbeteren. Kaldi

is een open-source toolkit dat veel wordt gebruikt in de spraakherkenning en natuurlijke taalverwerking gemeenschap. Het biedt een krachtig platform voor het ontwikkelen van automatische spraakherkenningsmodellen en kan worden aangepast aan de specifieke behoeften van de stottertherapie.
Naast Kaldi is er nog een ander ASR-model dat gebruikt kan worden om de stottertherapie te verbeteren, namelijk Wav2Vec. Wav2Vec is een deep learning-model dat in staat is om zeer nauwkeurig spraaksignalen om te zetten in tekst. Dit model kan met name van pas komen in situaties waarin spraaksignalen van slechte kwaliteit zijn, bijvoorbeeld bij spraakstoornissen zoals stotteren. Door het gebruik van Wav2Vec kan de therapie nog accurater worden, waardoor patiënten beter geholpen kunnen worden in hun dagelijks leven. Echter, het implementeren van Wav2Vec in de therapie vereist nog wel verder onderzoek en ontwikkeling om de effectiviteit en efficiëntie te waarborgen.

1.2. Onderzoeksvraag

Het doel van dit onderzoek is om te bestuderen of het mogelijk is om een ASR-model te gebruiken in combinatie met spraakherkenningstechnieken om een stotteraar te verstaan en om te bepalen of deze technologie geschikt is voor gebruik in echte stottertherapie. Om deze onderzoeksvragen te beantwoorden, zal er een literatuurstudie en experimenteel onderzoek worden uitgevoerd. Op volgende onderzoeksvragen zullen dan een antwoord worden gegeven:

- Welke aanpassingen zijn nodig in het ASR-model om het effectief te kunnen gebruiken in de context van stottertherapie?
- Hoe kunnen we de efficiëntie van het gebruik van ASR in stottertherapie maximaliseren?
- Welke combinatie van technieken en parameters resulteert in de beste prestaties van het ASR-model voor stottertherapie?

1.3. Onderzoeksdoelstelling

Het doel van dit onderzoek is om een proof-of-concept te ontwikkelen die antwoorden biedt op de onderzoeksvragen. De proof-of-concept zal worden ontwikkeld in Google Colaboratory en zal een duidelijke beschrijving bevatten van elke stap in het proces. Het primaire doel van de proof-of-concept is om de prestaties van verschillende ASR-modellen te evalueren en te vergelijken, met behulp van een dataset van stotterende spraak. Door middel van deze data kunnen we dan de modellen vergelijken op basis van een maateenheid. Na het uitvoeren van de experimenten en het evalueren van de resultaten, zal er een conclusie worden getrokken over welk ASR-model in combinatie met spraakherkenningstechnieken het meest geschikt is voor het verstaan van een stotteraar.

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 5, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2

Stand van zaken

In dit hoofdstuk wordt de stand van zaken behandeld, ook wel *state of the art* genoemd. Het doel van dit hoofdstuk is om u een overzicht te geven van de meest recente kennis en ontwikkelingen binnen ASR-modellen en machine learning met betrekking tot spraakherkenning. In dit gedeelte wordt voornamelijk ingegaan op modellen en hun werking. Bovendien wordt de relevante terminologie verduidelijkt en uitgelegd. Dit state-of-the-art overzicht biedt de lezer een solide basis om de verdere ontwikkelingen en resultaten in dit onderzoek te begrijpen.

2.1. Automatic Speech Recognition

Automatic speech recognition (ASR) is een technologie die spraakaudio omzet in tekst. Zo bestaan er dan ASR-modellen die de audio analyseren en deze omzetten in een reeks woorden die overeenkomen met wat er werd gezegd. ASR-systemen worden al in verschillende toepassingen gebruikt, namelijk chatbots, automatische translaties in video's en nog veel meer.(**Microsoft2017**) Modellen worden getraind met behulp van grote datasets gesproken taal en kunnen daarna geoptimaliseerd worden voor specifieke talen en taken. Gaandeweg zijn er veel kern technologieën ontwikkeld, zoals Gaussiaanse Mixture Models (GMM's), verborgen Markov-modellen (HMM's), mel-frequentie cepstrale coëfficiënten (MFCC's) en hun afgeleiden, ngram-taalmodellen (LM's), discriminerende training. Dit zijn een hele boel technieken die de stand-van-zaken op gebied van ASR enorm verbeterd hebben.(**Mahesha2016**)

2.1.1. Word Error Rate

Word Error Rate (WER) is een belangrijke maateenheid die wordt gebruikt om de nauwkeurigheid van ASR-systemen te evalueren. WER is afgeleid van de *Levenshtein-afstand*, ook wel bekend als de *edit distance*, die de minimale afstand berekent die nodig is om een reeks woorden om te zetten in een andere reeks woorden door mid-

del van toevoegingen, verwijderingen en substituties van woorden. Bij het evalueren van de nauwkeurigheid van een ASR-systeem wordt de WER berekend als de Levenshtein-afstand tussen een gerefereerd woord en zijn automatische transcriptie, genormaliseerd door de lengte van de reeks referentiewoorden. Dit betekent dat de WER de fouten in het automatische transcriptieproces meet als een percentage van het totale aantal woorden in de referentie-uitvoer. Een lage WER duidt op een hoge nauwkeurigheid van het ASR-systeem, terwijl een hoge WER betekent dat er meer fouten zijn opgetreden tijdens het transcriptieproces (**mccowan2004use**). Door gebruik te maken van de WER als evaluatiemaatstaf kan worden bepaald hoe goed een ASR-systeem presteert en kunnen er verbeteringen worden aangebracht om de nauwkeurigheid van het systeem te verbeteren.

De definitie van WER gaat als volgt: N_r zijn de totale woorden in de referentie transcriptie, N_a zijn de totale woorden in de transcriptie, S is het aantal vervangen woorden in de automatische transcriptie, D als het aantal woorden uit de referentie verwijderd in de automatische transcriptie, I is het aantal woorden die in de automatische transcriptie zijn ingevoegd en niet in de referentie voorkomen en H als het aantal correct herkende woorden. Zo kunnen we dan de woord error rate definiëren als:

$$WER = \frac{S + D + I}{N_r}.$$

Dit wordt het meest gebruikt als error verhouding, je kunt aan de hand van WER ook de word recognition rate (WRR) bereken:

$$WRR = 1 - WER = \frac{H - I}{N_r}.$$

Soms wordt er ook gebruik gemaakt van de word correct rate (WCR), deze maateenheid maakt geen gebruik van inserties en errors en wordt gedefinieerd als volgt:

$$WCR = \frac{H}{N_r}.$$

(**mccowan2004use**) Verduidelijking aan de hand van een voorbeeld:

> Referentietekst: ”Het is een zonnige dag vandaag.”
> Herkende tekst: ”Het is een regenachtige dag vandaag.”

Nu wordt er berekend hoe veel invoegingen (I), verwijderingen (D) en substituties (S) er nodig zijn om da herkende tekst overeen te laten komen met de referentietekst:

> Invoegingen (I): Geen
> Verwijderingen (D): Geen
> Substituties (S): ‘regenachtig’ in plaats van ’zonnige’

Het totale aantal woorden (N_r) is 6, dus als we dan de formule volgen ziet dit er als volgt uit:

$$WER = \frac{S + D + I}{N_r} = \frac{1 + 0 + 0}{6}.$$

$$WER \approx 0.1667$$

In dit voorbeeld wordt er een word error rate van ongeveer 0.1667 berekend. Dit wil zeggen dan 16.67% van de woorden in de herkende tekst afwijkt van de referentie-tekst.

2.1.2. Self-supervised Learning

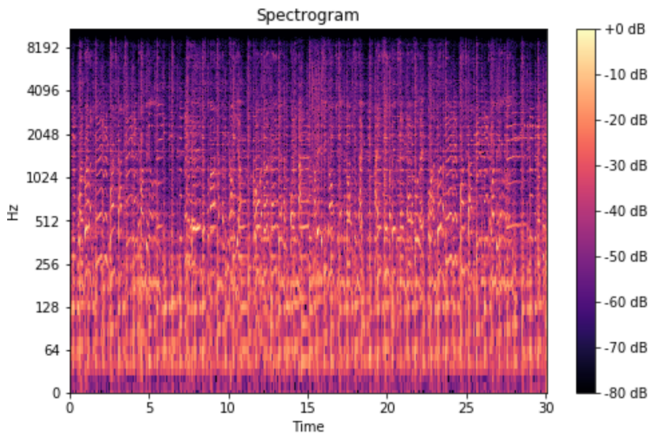
In het domein machine learning is self-supervised learning naar voren gekomen als een paradigma om algemene gegevens te leren. Normaal gesproken wordt bij supervised training gebruik gemaakt van data dat gelabeld is. Het probleem is dat ook al is er veel beschikbare data, je kunt pas gebruik maken van supervised learning als deze data gelabeld is. Bij self-supervised learning worden de labels zelf gegenereerd waardoor er geen door mensen geannoteerde labels nodig zijn (**Jaiswal2021**). ASR-systemen zoals Whisper en Wav2Vec gebruik maken van self-supervised learning. Waardoor het mogelijk wordt gebruik te maken van de grote hoeveelheid ongelabelde gegevens die beschikbaar zijn.

2.1.3. Whisper

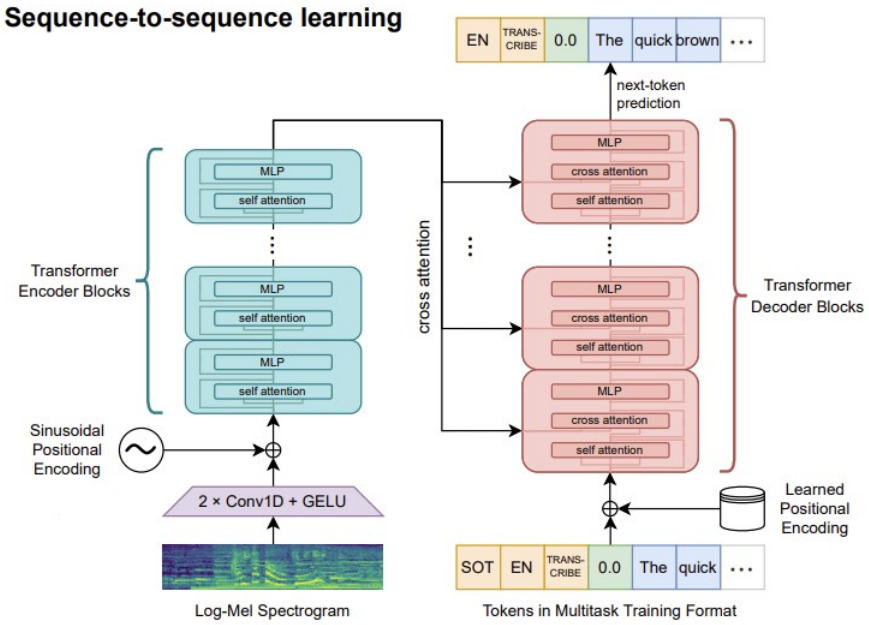
Whisper is een ASR-systeem dat in 2022 werd ontwikkeld door OpenAI. Het is getraind op 680.000 uur aan meertalige gegevens die via het internet zijn verzameld. Door gebruik te maken van grote en diverse data leidt tot verbeterde robuustheid voor accenten, achtergrondgeluiden en technisch taalgebruik(**OpenAI2022**). Om deze reden zou Whisper een geschikte ASR-systeem kunnen zijn voor dit onderzoek.

Whisper structuur

De architectuur van Whisper heeft een end-to-end benadering, geïmplementeerd als encoder-decoder transformer. De ingevoerde audio wordt opgesplitst in stukken van 30 seconden. Deze stukken worden dan omgezet in een log-Mel-spectogram wat vervolgens wordt doorgegeven aan een encoder, op figuur 2.1 ziet u een voorbeeld van een log-Mel spectogram. Veder gaat dan een getrainde decoder de corresponderende tekstonderschrift voorspellen (**OpenAI2022**). Op figuur 2.2 is een duidelijk overzicht weergegeven van de structuur van Wisper.



Figuur (2.1)
Voorbeeld van een log-mel spectogram, dit geeft een signaal weer op een frequentiespectrum. (**mel-spectrogram-medium**)



Figuur (2.2)
Overzicht van de structuur van het Whisper asr-systeem. (**Tjandra2020**)

Nederlandse taal

Whisper is in staat om meerdere talen te begrijpen, waaronder het Nederlands. Whisper heeft verschillende modellen van verschillende groottes, en de prestaties kunnen sterk variëren tussen deze modellen. Het Whisper large-v2-model presteert het beste voor de Nederlandse taal. Bij de Common Voice 9-dataset behaalt het een WER van 5,8% en bij de Librispeech-dataset een WER van 9,3% (**Tjandra2020**).

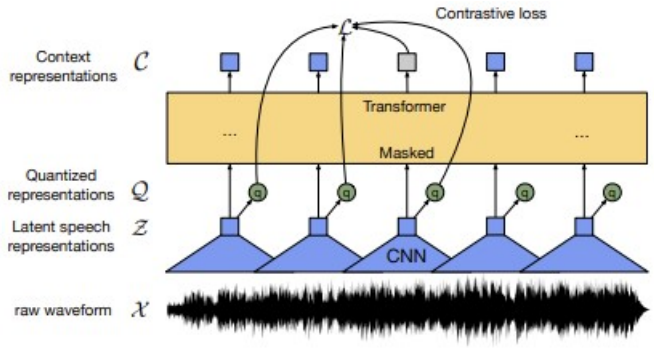
2.1.4. Wav2Vec 2.0

Wav2Vec 2.0 is een framework voor spraakherkenning dat is gemaakt door Facebook AI Research (FAIR). Het is heel populair geworden in de ASR-wereld omdat het heel

opmerkelijke prestaties levert. De reden waarom Wav2Vec 2.0 zo goed presteert is omdat het framework zich richt op de raw waveform van de spraak (**baevski2020wav2vec**). Dus in plaats van de audio om te zetten werken ze rechtstreeks met de audio van spraaksignalen.

Wav2Vec 2.0 structuur

Het model is samengesteld uit een Convolutional Neutral Network (CNN), ook wel de feature encoder genoemd, die de raw onbewerkte audio (X) neemt en het levert van latente spraakrepresentaties (z1, z2, ..., zT) voor T tijdsstappen. Vervolgens worden deze dan gevoerd aan een transformer. Deze transformer maakt dan representaties (c1, c2, ..., cT), bij deze stap wordt er ook informatie uit de gehele sequentie vastgelegd. De output van de feature encoder wordt dan gediscretiseerd met een quantization module om het representeren van een doel in het self-supervised learning (**baevski2020wav2vec**). In figuur 2.3 zie je een visueel beeld van de structuur van het Wav2Vec 2.0 model.



Figuur (2.3)
Visuele illustratie van de Wav2Vec 2.0 structuur (**baevski2020wav2vec**).

Performantie

Natuurlijk wanneer je een spraakherkenningsmodel wilt gebruiken moet je ook kijken naar de prestatie die het kan leveren. Wanneer Wav2Vec 2.0 getest wordt op de LibriSpeech dataset dan krijgen we goede resultaten weer. Bij de 10 minuten gelabelde data presteert het een Word Error Rate (WER) van 4.8% en bij ongelabelde is het 8.2%. Hiervoor moest het model natuurlijk eens getraind worden, dit werd gedaan door het 53 duizend uur ongelabelde data van de LibriVox dataset (**baevski2020wav2vec**).

Nederlands model

Wav2vec2-large-xlsr-53-dutch is gemaakt door Jonatas Grosman, een professor aan de katholieke universiteit van Rio de Janeiro. Zijn model is getest geweest met de Common Voice dataset en Collection of Single Speeker (CSS) dataset. Op deze datasets behaalde het een WER van 15.72% (**grosman2021xlsr53-large-dutch**).
Het maakt gebruik van XLSR, een taalmodel is dat gemaakt door Huggingface en Facebook (**2022**). Het doel hiervan is om modellen te trainen die meerdere talen

begrijpen en te kunnen verwerken. In tegenstelling tot traditionele taalmodellen, die meestal getraind zijn op één taal, kunnen XLSR-modellen meerdere talen bevatten en kunnen worden afgestemd op een specifieke talen om zo de prestatie van die talen te verbeteren (**2021**).

2.2. Stotter Correctie

Wanneer iemand een ASR model gebruikt en onvloeiend spreekt dan heeft het model moeite met het correct te transcriberen. Gelukkig kan er gebruikt worden van stotter correctie. In 2020 hebben studenten van PES University in India een automatische correctie methode van onvloeibare spraak uitgewerkt. Er wordt gebruik gemaakt van Mel Frequency Cepstral Coefficients (MFCC) en Linear Predictive Coefficients (**KN2020**).

2.2.1. Mel Frequency Cepstral Coefficients

Mel Frequency Cepstral Coefficients is een concept voor het extraheren van kenmerken van een akoestisch signaal. Het is gebaseerd op de frequenties van meer dan 1KHz, dus wat het menselijk gehoor niet meer kan waarnemen. De berekeningen die door MFCC worden uitgevoerd zijn zeer afhankelijk van het proces waarbij de signalen veranderen van analoog naar digitaal. MFCC voert berekeningen uit variërend van de lengte van de golfhoogte, ruis en andere dingen zodat de woorden die werden uitgesproken door de gebruiker worden verkregen (**haq2020speech**).

2.2.2. Linear Predictive Coefficients

Linear Predictive Coding (LPC) is een techniek die wordt gebruikt in audioprocessing en spraakanalyse om de spectrale eigenschappen van een signaal te modelleren. De techniek is gebaseerd op het idee dat het spectrum van een signaal kan worden voorspeld door gebruik te maken van een lineair voorspellingsmodel. Bij LPC wordt het signaal opgedeeld in korte stukjes en wordt er voor elk stukje een set lineaire predictieve coëfficiënten berekend. Deze coëfficiënten worden gebruikt om een lineair voorspellingsmodel te bouwen, dat vervolgens kan worden gebruikt om het spectrum van het signaal te modelleren en te analyseren. De LPC-techniek wordt veel gebruikt in spraaktechnologie, bijvoorbeeld om spraak te comprimeren of om spraak te herkennen in een spraakherkenningssysteem. Het is ook een belangrijk onderdeel van codecs zoals MP3 en AAC, die worden gebruikt voor het comprimeren van audiobestanden. (**bradbury2000linear**).

2.2.3. Design

Het design bestaat uit een sequentie van drie algoritmen. Het eerste algoritme is voor het verwijderen van herhaling, dus worden alle gerepeteerde woorden geëlimineerd en behoudt alle niet-herhaalde woorden. Dit wordt behaald door gebruik

MFCC- en LPC-functies van 2 opeenvolgende woorden te extraheren. Als er dan een hoge correlatie is tussen de geëxtraheerde kenmerken van de 2 woorden duidt op een grote gelijkenis tussen woorden, waarna het herhaalde woord wordt geëlimineerd.

Er is ook een algoritme voor het verwijderen van lange pauzes in de audio. Dit wordt gedaan door het berekenen van de tijd tussen het einde van een woord en het begin van het opeenvolgende woord. Wanneer dit dan meer dan 0.5s is impliceert dit op een lange pauze.

Uiteindelijk is er dan ook een verlenging verwijdering aan de hand van een algoritme. Hierbij wordt het audio fragment opgedeeld in in frames van 50ms. Dan wordt er rekening gehouden met de MFCC- en LPC-kenmerken die overeenkomen met opeenvolgende frames en wordt er een correlatiefactor gevonden. Deze factor wordt achterhaald tussen kenmerken van opeenvolgende frames. Bij een langdurig optreden van hoge correlatie tussen frames impliceert dit dat er een verlenging is (bijvoorbeeld: stttttttop) (**KN2020**).

2.2.4. Performantie

De algortimen zijn effectief in staat om lange pauzes, verlengingen en herhalingen te herkennen en te corrigeren. Bij de 80 instanties van lange pauzes heeft het algoritme 78 keer de pauzes herkend, dat is een accuraatheid van 97.5%. Bij het testen op herkenning van verlengingen en herhalingen bevatte de audiobestanden in totaal 60 herhalings- en 70 verlengingsgebeurtenissen. Hierbij werden 2 verschillende functies gebruikt: MFFC en LPC. In figuur 2.4 kunt u zien dat MFCC accurater is dan LPC. MFCC behaalde een gemiddelde nauwkeurigheid van 92.8% en LPC een 89.9% (**KN2020**)

Features	Total repetition	Removed repetition	Retained repetition	Accuracy
MFCC	60	54	6	90%
LPC	60	52	8	86.7%
Features	**Total prolongation**	**Removed prolongation**	**Retained prolongation**	**Accuracy**
MFCC	70	67	3	95.7%
LPC	70	65	5	92.9%

Figuur (2.4)
Tabel dat overzicht geef over de accuraatheid van de functies (MFCC, LPC). (**KN2020**)

3

Methodologie

Dit hoofdstuk bespreekt hoe het onderzoek zal worden aangepakt. Er worden verschillende aspecten besproken om het uiteindelijke doel te bereiken: het onderzoeken van de theoretische mogelijkheid om stottertherapie in virtual reality te ontwikkelen. Een proof-of-concept zal worden opgesteld om te beoordelen of ASR-systemen stotteraars kunnen begrijpen met behulp van audio-bewerkingsfuncties

3.1. Dataset

De eerste fase in een onderzoek met machine learning is opzoek gaan naar geschikte data. De data moet natuurlijk relevant zijn voor het onderzoek, dus wat er nodig is zijn de audiofragmenten van mensen die stotteren. Daarnaast moet er ook nog voldoende data zijn om duidelijke conclusies te kunnen trekken. Mocht er te weinig data zijn dan is er een mogelijkheid dat de resultaten van het onderzoek afwijken van de realiteit. Ten slotte zou er ook genoeg variatie in de gegevens moeten zitten, want anders zijn de onderzoeken uitgevoerd op één bepaalde toepassing, wordt ook wel *overfitting* genoemd.

Voor het samenstellen van de dataset werd gezocht naar audio van personen die stotteren. Podcasts bleken hiervoor een goede bron te zijn en waren gemakkelijk te vinden op platforms zoals YouTube. Om voldoende variatie in de dataset te garanderen, bevat deze zowel audio van een man als van een vrouw. Een van de personen wiens audio in de dataset is opgenomen, is Charlotte Roggeman. Ze heeft haar hele leven last gehad van stotteren, wat relevant is voor dit onderzoek. De andere persoon in de dataset is Rowan Amatkario. Hij heeft sinds zijn jeugd last van stotteren en stottert nog steeds.

Naast de audiofragmenten moet er ook een transcriptie zijn per fragment, dit be-

staat uit een eenvoudig CSV-bestand. Dit bestand is opgedeeld in 2 kolommen. De eerste kolom is het pad naar de audio file en de tweede kolom is de transcriptie ervan. Transcriptie is een nood omdat dit een manier biedt om de nauwkeurigheid te meten van de ASR-modellen. In het tekstbestand komt dan de te verwachten transcripties van de modellen te staan. zo kunnen dan de word error rates worden berekend per model.

Er werd ook nog een dataset gemaakt, die bestond uit zelf ingesproken audio zonder het stotteren. Dit is gedaan om vergelijkingen te kunnen maken en een betere analyse te kunnen uitvoeren. De structuur en transcripties van deze dataset zijn hetzelfde als die van de andere dataset, maar dan zonder het stotteren.

Voor het gebruik van de datasets werd AudioFolder gebruikt, een dataset builder gemaakt door Hugging Face. AudioFolder is specifiek ontworpen om snel audio datasets in te laden zonder dat er enige code hoeft geschreven te worden. Ook werd de dataset online gezet, om er makkelijk aan te kunnen. Door het online te plaatsten op de Hugging Face website wordt er ook altijd automatisch een weergave van de dataset weergegeven.

3.1.1. Structuur

Om goed gebruik te kunnen maken van de dataset moet er natuurlijk gekeken worden naar de structuur. De dataset is een dictionary met één key, namelijk 'train'. Corresponderend met deze key staat nog een andere dictionary, deze bevat twee keys: 'audio' en 'transcription'.

```
DatasetDict({
    train: Dataset({
        features: ['audio', 'transcription'],
        num_rows: 32
    })
})
```

Listing 3.1.1: Dataset structuur die wordt weergegeven wanneer er *print(dataset)* wordt uitgevoerd.

Als de key 'audio' wordt aangesproken krijgt men een lijst terug met dictionaries. Deze dictionaries hebben drie keys: 'path', 'array' en 'sampling_rate'. De sleutel 'path' geeft een string terug die het pad is naar het audiobestand. Dan de key 'array' geeft een lijst weer van floats, deze lijst representeert het audio bestand. De laaste key 'sampling_rate' geeft de frequentie terug van de audio in Hertz.

```
{'path':
    '/content/drive/MyDrive/DatasienceNAI/Dataset/Data/Rowan_1.mp3',
    'array': array([-0.09932998, -0.13783528, -0.10971285, ... ,
    0.00368308,
    0.00385619,  0.0024402 ]), 'sampling_rate': 48000}
```

Listing 3.1.2: Structuur van dictionary in de lijst, dit wordt weergegeven wanneer *print(dataset['train']['audio'][0])* wordt uitgevoerd.

Moest de andere key 'transcription' in plaats worden aangesproken wordt er een string teruggegeven. Deze string is de transcriptie van het corresponderende audiobestand.

```
ik ben denk al 2 jaar ben ik een part-time softwaredeveloper hier
```

Listing 3.1.3: Wat er wordt weergegeven wanneer *print(dataset['train']['transcription'][0])* wordt uitgevoerd.

3.2. Omgeving

Nu dat de data is verzamelt en opgesteld moet er een omgeving worden opgezet waar de dataset kan worden toegepast. De gekozen omgeving is Google Colabratory. Dit is een cloud-based omgeving waar machine learning kan worden uitgevoerd. In Google Colab kan python-code geschreven en uitegvoerd worden zonder de noodzaak van lokale installatie of krachtige hardware.

3.2.1. Packages

Natuurlijk moeten er ook nog een paar packages geïnstalleerd worden om audio functies te kunnen gebruiken en om aan de ASR-modellen in te laden. Hieronder zie je een overzicht van packages waarvan gebruik is gemaakt dit onderzoek:

- jiwer: met deze package kun je de word error rate van de asr-modellen berekenen op een gemakkelijke manier
- librosa: package voor muziek en audio analyse, met behulp van deze package kun je bijvoorbeeld gebruik maken van MFCC
- huggingsound: voor het inladen van Wav2Vec 2.0 model en voor het inladen van de dataset
- openai-whisper: voor het inladen van het Whisper model
- numpy: wordt gebruikt om grafieken mee te maken en zo de modellen visueel te kunnen vergelijken

- scipy: wordt gebruikt om wiskundige berekeningen en wetenschappelijke computing te doen.
- datasets: package gemaakt door Hugging Face, zorgt er voor dat het werken met datasets simpeler verloopt

Om al deze packages te installeren werd er gebruik gemaakt van de package installer pip. Om pip te gebruiken moeten er uitroeptekens in de code staan, dit omdat je dan zo shell commando's kunt uitvoeren.

```
!pip install librosa
!pip install scipy
!pip install huggingsound
!pip install -U openai-whisper
!pip install jiwer
!pip install pydub
```

Listing 3.2.1: Shell commando's die de nodige packages installeert.

3.3. Proof of Concept

Om de haalbaarheid van het gebruik van ASR-systemen in combinatie met audio bewerkende functies te onderzoeken, is een proof-of-concept (PoC) opgesteld. Voor de PoC zijn de ASR-modellen Whisper en Wav2Vec 2.0 geütiliseerd. Deze modellen zijn vervolgens getest op de twee hiervoor besproken datasets. Hierbij is gekeken naar de prestaties van de modellen op de ruwe datasets en op de datasets waarbij gebruik wordt gemaakt van audio bewerkende functies.

Voor het toepassen van audio bewerkende functies zijn verschillende algoritmen gebruikt. De prestaties van de modellen zijn beoordeeld op basis van de Word Error Rate (WER). De WER geeft aan hoeveel woorden er onjuist zijn geïnterpreteerd door het ASR-systeem ten opzichte van het totale aantal woorden in de audiofragmenten.

De PoC heeft als doel om aan te tonen dat de implementatie van audio bewerkende functies inderdaad kan leiden tot betere prestaties van de ASR-modellen bij het verstaan van stotteraars. De resultaten van de PoC zullen worden besproken en geanalyseerd om zo de haalbaarheid van stottertherapie in virtual reality met behulp van ASR-systemen en audio bewerkende functies te onderzoeken.

4

Proof Of Concept

In dit hoofdstuk zal er dieper ingegaan worden op de ontwikkeling van het proof-of-concept. Het uiteindelijke doel van deze proof-of-concept is om te laten zien dat het theoretisch mogelijk is om ASR-systemen stotteraars te laten verstaan met behulp van audio bewerkende functies. Op basis van de bevindingen en resultaten van het PoC kunnen er conclusies worden getrokken over de haalbaarheid van het gebruik van audio bewerkende functies om ASR-systemen stotteraars te laten verstaan.

4.1. Stotter Correctie

De eerste stap in deze proof-of-concept is het implementeren van algoritmen die het audiobestand gaan bewerken. Er worden drie algoritmen gebruikt. Het eerste algoritme is bedoeld om de verlengingen van de stotteraars te verwijderen. Daarnaast is er een algoritme dat lange pauzes van de stotteraars detecteert en verwijdert. Ten slotte is er een algoritme dat repeterende woorden uit het audiobestand verwijdert. Het doel van deze algoritmen is om de ASR-systemen een beter begrip te geven van de audiobestanden.

4.1.1. Verlenging Verwijdering

4.1.2. Repetities Verwijderen

Het algoritme voor het verwijderen van repetities start met het converteren van het audiobestand naar 22.05kHz. Dit maakt het makkelijker repetities te detecteren in de audio. Dit werd gedaan met een eenvoudige methode genaamd *convert_sr_audio* die wordt weergegeven in figuur 4.1.1.

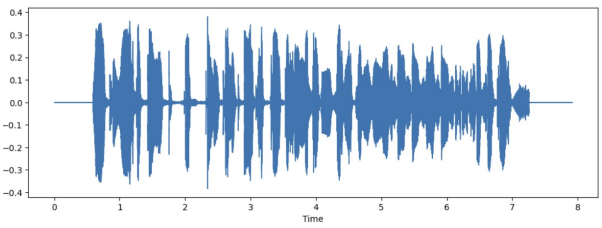
```
def convert_sr_audio(audio, orgSr):
sr = 22050
convertAudio = librosa.resample(audio, orig_sr=orgSr,
    target_sr=sr)
return convertAudio
```

Listing 4.1.1: Methode dat de audio omzet naar een frequentie van 22.05kHz.

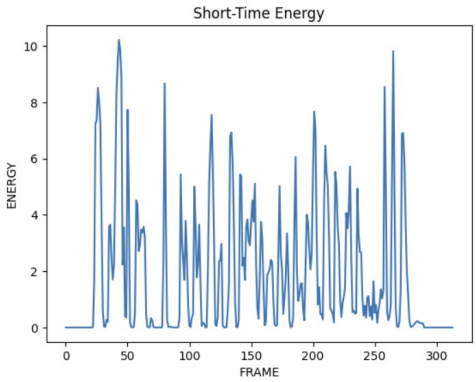
Het volgende dat moet gebeuren, is het berekenen van de short-time energy (STE) van de audio. Dit wordt gedaan aan de hand van de volgende vergelijking:

$$E_n = \sum_{m=-\infty}^{\infty} (x(m)w(n-m))^2$$

Bij deze vergelijking staat $E(n)$ voor de short-time energy op tijdstip n. Om het concept van STE beter uit te kunnen leggen, wordt er gewerkt met een voorbeeld. In figuur 4.1 is een visualisatie te zien van een audiofragment, waarvoor de STE is berekend. De STE van de audio wordt weergegeven in figuur 4.2. Zoals te zien is in de eerste figuur, is er geen geluid aan het begin en einde van de audio, wat overeenkomt met de grafiek. Het STE van een bepaald geluid komt dus overeen met de luidheid of stilte van het geluid. Als de audio luid is op een bepaald moment, zal dit overeenkomen met een hoge energie, en als het stil is op een bepaald moment, zal dit overeenkomen met een lage energie.



Figuur (4.1)
Visualisatie van een audiosignaal.



Figuur (4.2)
Short-time energy (STE) dat correspondeert met het audiosignaal dat links van deze grafiek staat.

Deze berekende energie kan dan gefilterd worden om hoge frequentie transities te verwijderen. Dit zorgt er voor dat er een dat het signaal vereffend wordt. Daarna wordt er een drempelwaarde toegepast op het vereffend signaal. De drempelwaarde die genomen werd is 0.08, dit volgt een paper waar wordt voorgesteld een design voor correctie van stotter .

4.2. Analyse

Nu dat alle nodige algoritmen in plaats zijn en klaar voor gebruik kan er aan analyse worden gedaan. Met behulp van een methode (4.2.1) worden de datasets overlopen en getranscribeerd door de twee ASR-modellen: Whisper large-v2 en Wav2Vec 2.0 (wav2vec-large-xlsr-53-dutch). Deze methode maakt ook gebruik van de jiwer package voor het berekenen van de word error rate per model.

In het begin van de methode worden twee lijsten aangemaakt. De bedoeling van deze lijsten is om ze te vullen met word error rates per audio bestand. Dit wordt gedaan aan de hand van een for-lus die de dictionary in de dataset overloopt.

In het begin van de for-lus wordt het pad van het audiobestand bijgehouden en de transcriptie, het pad wordt als 'path' bijgehouden en de transcriptie als 'transcription'. Vervolgens wordt met behulp van het path variabele de audio ingeladen en een mel spectogram gegenereerd. Dit spectogram wordt dan benut door het Whisper model en genereert een transcriptie. Deze gegenereerde transcriptie wordt samen met de audio transcriptie gebruikt om de Word Error Rate (WER) te berekenen, waarna deze in de juiste lijst wordt opgenomen.

Daarna is het de beurt aan het Wav2Vec-model om een transcriptie te genereren. Hiervoor moet het pad eerst in een lege lijst worden gestoken, die dan aan het model wordt meegegeven. Waarna dit een transcriptie oplevert die direct gebruikt kan worden om de rate te calculeren. Deze rate komt dan ook terecht in de correcte lijst.

Wanneer de for-lus volledig is doorlopen, wordt de gemiddelde WER per model berekend met behulp van een kleine methode genaamd *gemiddelde* (4.2.2). Daarnaast worden ook de twee lijsten met word error rates teruggegeven.

```
from jiwer import wer

def WERModellen(dataset):
    listWERWisper = []
    listWERWav2Vec = []
    for audio in dataset['train']:
        transcription = audio['transcription']
        path = audio['audio']['path']
        #Wisper
        audio = whisper.load_audio(path)
        audio = whisper.pad_or_trim(audio)
        mel = whis-
            per.log_mel_spectrogram(audio).to(modelWhisper.device)
        options = whisper.DecodingOptions(fp16 = False)
        result = whisper.decode(modelWhisper, mel, options)
        rate = wer(result.text, transcription)
        listWERWisper.append(rate)
        #Wav2Vec
        path = [path]
        modelTranscr = modelWav2Vec.transcribe(path)
        transW2V = modelTranscr[0]['transcription']
        rate2 = wer(transW2V,transcription)
        listWERWav2Vec.append(rate2)
    return gemiddelde(listWERWav2Vec),gemiddelde(listWERWisper),
        listWERWav2Vec,listWERWisper
```

Listing 4.2.1: Methode dat de word error rate (WER) van de ASR-modellen (Whisper, Wav2Vec 2.0) berekend op een bepaalde dataset. De dataset moet als argument worden meegegeven aan de methode.

```
def gemiddelde(listProp):
    return sum(listProp) / len(listProp)
```

Listing 4.2.2: Kleinde methode om het gemiddelde te berekenen. Wordt gebruikt in de WERModellen methode om het gemiddelde WER te berekenen.

4.2.1. Zonder Audiofuncties

Als eerste werd het gemiddelde WER op beide datasets berekend zonder enige bewerkingen te doen op de audiofragmenten. Dit wordt gedaan om te onderzoeken welk model stotterende spraak het best zou begrijpen. De hypothese bij dit onderzoek gaat als volgt:

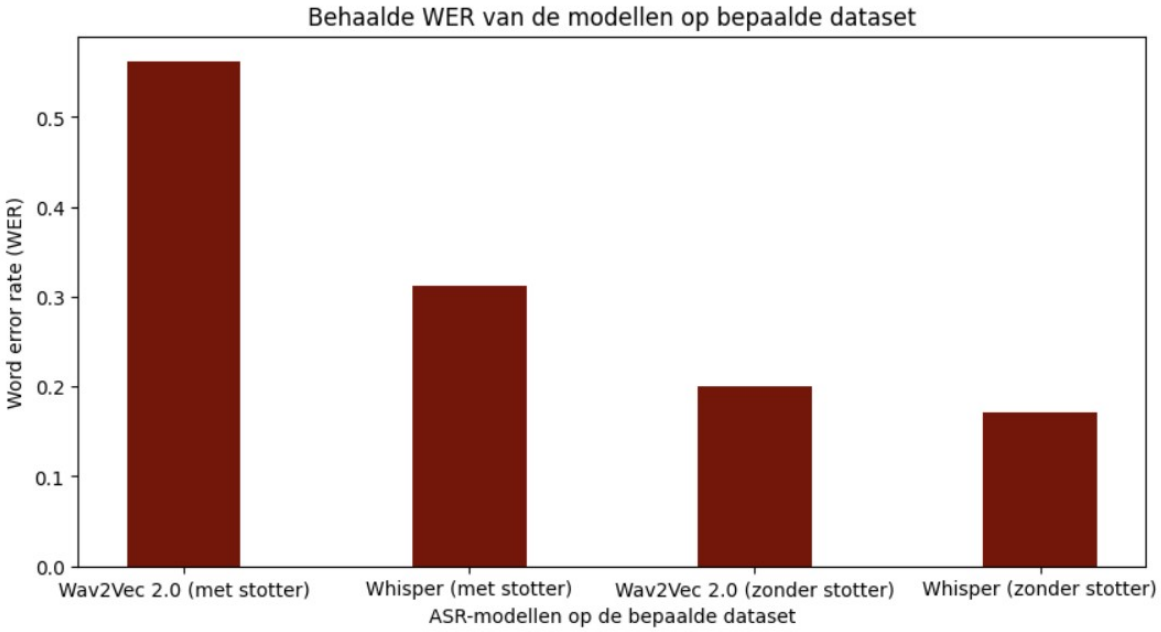
Hypothese 0: *Beide modellen scoren dezelfde word error rates op de dataset zonder het stotteren als de dataset met het stotteren.*

Hypothese 1: *De modellen scoren een lagere word error rates op de dataset zonder het stotteren dan op de dataset met het stotteren.*

Resultaten

Uit de gegevens blijkt dat zowel het ASR-model Wav2Vec 2.0 als het model Whisper verschillende prestaties vertonen op de datasets met en zonder stotteren. Voor de dataset met stotterende spraak had Wav2Vec 2.0 een word error rate van 0.5619%, terwijl Whisper een word error rate van 0.3119% behaalde. Dit suggereert dat Whisper beter in staat was om spraak van mensen met stotteren te verwerken en te transcriberen dan Wav2Vec 2.0 model.

Daarentegen, bij de dataset zonder stotteren presteerde beiden Wav2Vec 2.0 en Whisper beter. Wav2Vec haalde hierbij een goede WER, namelijk 0.1999%. Whisper doet het nog een beetje beter dan het Wav2Vec model, want het haalt immers een WER van 0.1705%.



Figuur (4.3)
Staafdiagram dat de gemiddelden per dataset van de modellen weergeeft.

Beiden modellen scoren dus veel slechter op de dataset met stotterende spraak, Wav2Vec 2.0 meer dan twee maal slechter zelfs. Toch presteert Whisper nog redelijk goed in vergelijking met wat het zou kunnen behalen. Dit duidt dus op de complexiteit van spraakherkenning bij mensen met stotteren en tonen aan dat verschillende ASR-modellen uiteenlopende prestaties kunnen hebben, afhankelijk

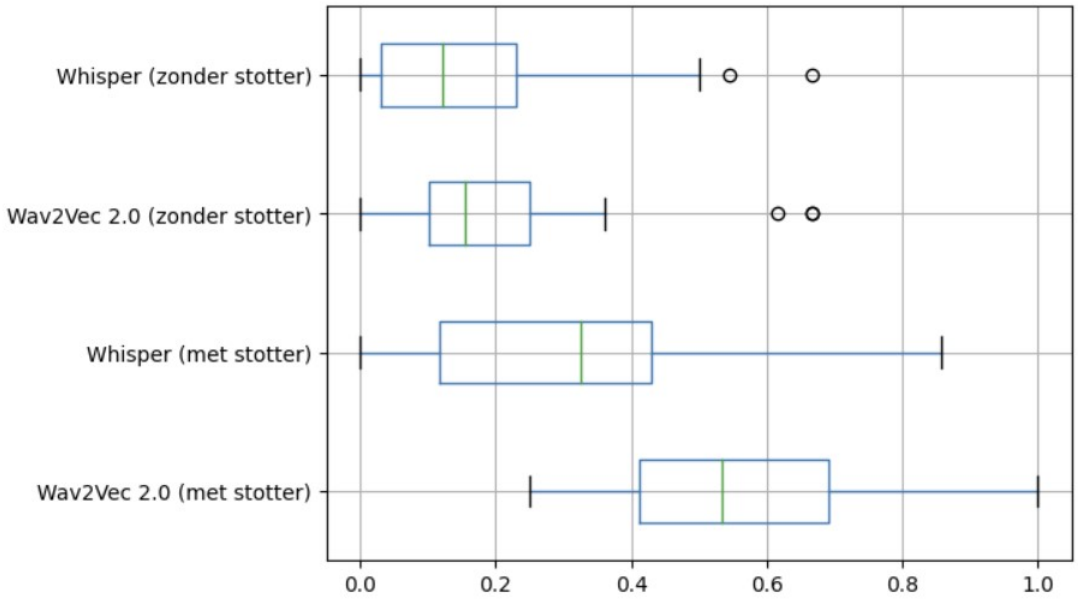
van de aard van de spraak.

Verder moet worden aangetoond of de nulhypothese juist is of niet. Dit kan worden gedaan met behulp van een *t-test voor gepaarde steekproeven*. Deze test is gekozen omdat de gegevens in beide datasets dezelfde woorden bevatten, maar de eerste dataset stotterende spraak bevat en de tweede niet.

Op figuur 4.4 zie je een visualisatie van de gevonden data, het toont een boxplot per dataset die de asr-modellen hebben overlopen. Nu dat we een visualisatie van de data hebben kunnen we de p-waarde berekenen. Deze waarde toont dan aan of er een significant verschil is tussen WER's op de twee datasets.

P-waarde Whisper: 0.00294
P-waarde Wav2Vec 2.0: $0.3750 * 10^{-10}$

Beide p-waarden duiden op een verwerping van de nullhypothese. Dit wil dus zeggen dat er een significant verschil is tussen in de word error rates van de stotter dataset en de word error rates van de dataset met normale spraak. Dit wil dus ook zeggen dat hypothese 1 correct is, beide modellen halen een lagere WER's op de normale spraak dataset.



Figuur (4.4)
Boxplot dat de word error rate (WER) van de audiobestanden weergeeft.

5

Conclusie

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

Laatste jaren is de informatica wereld enorm vooruit gegaan. Zorglab 360°, een onderzoekscentrum in de zorg sector van HOGENT, maakt gebruik van deze vooruitgang. Ze werken namelijk met Virtual Reality (VR) voor een aantal domeinen. Dit wordt gedaan om de patiënt beter voor te bereiden en realistische oefenkansen te bieden. Dit realiseren ze aan de hand van interactieve video's. Stottertherapie is onder andere één van de domeinen waar ze dit toepassen. Het probleem is nu dat het interactieve gedeelte heel rudimentair is en nog veel uitgebreid kan worden. Één van de manieren waarop het kan worden uitgebreid is met behulp van artificiële intelligentie (AI) en Natural Language Processing (NLP).
Natural language processing verwijst naar een tak van de informatica, om meer specifiek te zijn de tak van artificiële intelligentie die zich bezig houdt met het vermogen om tekst en gesproken woorden te begrijpen. Het doel is dat deze computers de gesproken taal begrijpen op dezelfde manier als mensen dat doen (**Education**).
Natural language processing wordt al in veel gevallen gebruikt. Google Translate is een groot voorbeeld van algemene beschikbare NLP-technologie, zij maken gebruik van de AI Google Neural Machine Translation (GNMT). Naast GNMT zijn er ook nog andere AI's, namelijk wave2vec. Wave2vec is een open source NLP AI model dat audio gebruikt om automatic speech recognition (ASR) modellen te trainen. Het is gemaakt door Facebook en hun doel is dat deze AI werkt voor alle talen. In deze bachelorproef wil ik nagaan of het mogelijk is om wave2vec te integreren bij stottertherapie in VR en kijken hoe effectief dit is.

A.2. State-of-the-art

De nieuwste versie van wave2vec is wav2vec 2.0. Het maakt gebruik van niet-gelabelde training data om spraakherkenning voor meer talen, dialecten en domeinen mogelijk te maken. Met maar slechts één uur aan gelabelde data presteert wave2vec 2.0 beter dan de vorige versie (**Baevski2020**).
LibriSpeech is de meest gebruikte data die als maatstaf wordt gebruikt om te bepalen hoe goed een natural language prosessing AI is. Het bestaat uit een collectie van ongeveer 1000 uur aan audioboeken. De meeste van de audioboeken komen van Project Gutenberg, een digitale bibliotheek is met meer dan 60,000 e-books (**7178964**).
Met behulp van LibriSpeech kunnen we dan bepalen welke natural language processing AI-model het beste is voor de taal Nederlands. Uit verder onderzoek bleek dat wav2vec2-large-xlsr-53-dutch gemaakt door Jonatas Grosman de beste AI is die werkt met het Nederlands (**Grosman2021**). Deze AI is de beste omdat het een Word Error Rate (WER) van 15.72% heeft .Dit is een redelijk groot verschil met de tweede beste, wav2vec2-large-xls-r-300m-nl, die een WER heeft van 17.17%.
Volgens het doctoraal onderzoek van R. Boey in 2008 heeft stottertherapie een matige doeltreffendheid. Namelijk dat 68% van de patiënten niet meer stottert na therapie. De effecten zijn ook het meest uitgesproken voor de behandeling bij heel jonge kinderen waarvan 82% niet meer stottert **Boey2008**. Er is dus zeker ruimte voor verbetering en artificiële intelligentie kan daar een rol bij spelen.

A.3. Methodologie

De eerste fase van het onderzoek is het verzamelen van natural language AI's voor de taal Nederlands. Deze AI's zouden dan gebruikt worden om de correcte fragmenten af te spelen bij een interactief moment.
Wanneer de AI's verzamelt zijn dan zullen deze vergeleken worden. Dit zal op bepaalde criteria gedaan worden, namelijk:

- Kan deze AI stotteren correct detecteren?
- Hoe presteert de AI op de hardware? Hoeveel maakt het model gebruik van de processor, RAM-geheugen, ... ?
- Is de AI open source? Wat zijn de bijkomende kosten bij het model?
- Op welke manier gaat de AI moeten worden geïntegreerd in VR?

Op deze criteria gaat er dan een model gekozen worden. Deze fase van de bachelorproef gaat data en grafieken genereren over de maatstaven die hierboven zijn uitgelegd.
Op basis van de analyse van deze gegevens kan er een weloverwogen keuze worden gemaakt voor de toepassing van artificiële intelligentie. Na de keuze van het

model kan dit worden geïntegreerd in de virtual reality toepassing van Zorglab 360°. Hierdoor kan het uiteindelijk worden gebruikt in stottertherapie. Het succesvol integreren van AI in de therapie kan bijdragen aan een nieuwe evolutie in stottertherapie en het onderzoek zal daarmee een belangrijke bijdrage leveren aan deze ontwikkeling.

A.4. Verwacht resultaat, conclusie

Wanneer de AI volledig is opgesteld en kan gebruikt worden in virtual reality zal deze data kunnen tonen na enige testen. Deze grafieken kunnen dan trends of patronen aanduiden die er voor niet herkenbaar waren. De data kan dan weergeven bij welke woorden of lettergrepen de gebruiker het meeste moeite mee heeft.
Hieruit kunnen we concluderen dat dit logopedisten zou helpen om de individuele behoeften van elke gebruiker beter te begrijpen en hun therapie daarop afstemmen. Dit wil dus zeggen dat natrual language processing AI het potentieel hebben stottertherapie te verbeteren. Hopelijk komt er uit dit onderzoek een PoC die in praktijk kan worden gebruikt en voor evolutie zorgt in de stottertherapie.