# De kwaliteit van fonetische transcripties

Begeleider:
Joost van Doremalen
j.vandoremalen@let.ru.nl

Begeleider:
Theo Schouten
t.schouten@cs.ru.nl

Auteur:
Robin Oostrum (0609803)
robinoostrum@student.ru.nl

27 juni 2011

# Inhoudsopgave

# 1 Inleiding

## 1.1 Introductie

Taalwetenschappers maken voor hun onderzoek gebruik van *corpora*: grote hoeveelheden gesproken tekst. Een dergelijk corpus bestaat meestal uit paren van audio- en annotatiebestanden. De annotatiebestanden, ook wel transcripties genoemd, geven aan wat er gezegd wordt in het bijbehorende audiobestand. Veel van zulke transcripties worden handmatig ingevoerd, en zijn niet altijd even betrouwbaar. Het is daarom wenselijk dat een bestaande transcriptie getest kan worden op een bepaald geluidsbestand, om te controleren in welke mate beide met elkaar overeenkomen.

Een dergelijke toepassing is de TQE-tool. TQE, voluit *Transcription Quality Evaluation*, is een project opgezet vanuit de faculteit der letteren aan de Radboud Universiteit Nijmegen, in samenwerking met Clarin-NL en het Max Planck Instituut ([1], [2], [3], [4], [5]). De TQE-tool maakt het mogelijk om paren van audio- en annotatiebestanden te uploaden, en behandelt deze als volgt: audiosignalen en hun fonetische transcripties worden uitgelijnd, segmentsgrenzen worden afgeleid per foneem, en voor elke segment-foneemcombinatie wordt bepaald hoe goed deze "bij elkaar passen". Dit wordt uitgedrukt in een TQE-maat, een getal van 0-100, die de kwaliteit van de fonetische transcriptie per foneem[1] weergeeft (hoe hoger het getal, hoe groter de overeenkomst tussen audiosignaal en transcriptie).

## 1.2 Opzet

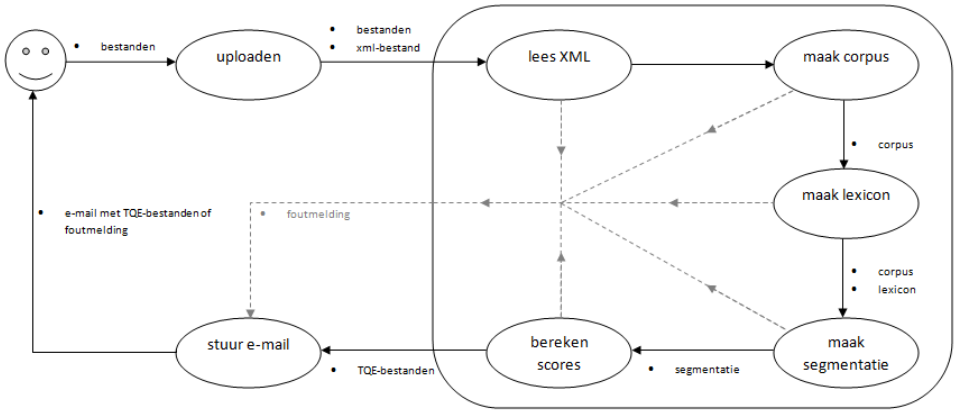
In deze scriptie onderzoek ik de volgende hoofdvraag: **hoe kan de kwaliteit van fonetische transcripties zo goed mogelijk geëvalueerd worden?** Het onderzoek kan globaal worden verdeeld over twee richtingen: de software-engineeringskant (het ontwikkelproces van de applicatie) enerzijds, en de spraaktechnologische kant (gebruikte formules, akoestische modellen etc.) anderzijds. De hoofdvraag kan verder worden opgedeeld in enkele deelvragen: eerst zal ik antwoord geven op de vraag wat de TQE-tool nu eigenlijk precies is (zie de introductie voor een korte uitleg), uit welke componenten deze is opgebouwd, en welke keuzes er gemaakt zijn tijdens de ontwikkeling ervan. Daarna komt de technische kant aan bod, en wordt antwoord gegeven op de vraag hoe per foneem de TQE-maat kan worden berekend: moet voor verschillende fonemen ook op een andere manier de score berekend worden,

[1] In taal- en spraakwetenschappen is een foneem (van het Griekse $\phi\omega\nu\eta\mu\alpha$) de kleinste begrensde eenheid van geluid, waarmee nog een zinvol onderscheid kan worden gemaakt tussen verschillende spraakuitingen.

wat voor scores krijgen “verkeerd” geannoteerde fonemen, en hoe betrouwbaar zijn de resultaten?
Tot slot worden de belangrijkste resultaten van het onderzoek besproken, en is er aandacht voor een conclusie en mogelijkheden voor verder toekomstig onderzoek.

# 2 TQE-tool

De TQE-tool is opgebouwd uit verschillende componenten. De onderverdeling begint met die in een online en een offline gedeelte, respectievelijk de web-applicatie en de tool zelf, die op zijn beurt weer uit componenten bestaat. Zie onderstaande flowchart voor een overzicht van de activiteiten en de dataflow:




Figuur 1: Dataflow binnen de tool

## 2.1 Uploaden

Het uploaden van audio- en annotatiebestanden gebeurt met behulp van een php-script. De gebruiker selecteert paarsgewijs een audio- en een annotatiebestand. Daarnaast geeft hij/zij een e-mailadres op waar de resultaten naar verstuurd kunnen worden, en geeft de gebruiker aan in welke “tier” van de annotatiebestanden de fonetische transcriptie op woordniveau te vinden is. Bij het uploaden wordt onmiddellijk gecontroleerd of er geldige wav-bestanden voor de audio, en geldige awd-bestanden voor de textgrids zijn geselecteerd. Vervolgens worden de bestanden in een nieuwe map op de server gezet, en wordt er een xml-bestand aangemaakt met daarin alle informatie over deze

upload: e-mailadres gebruiker, ip-adres gebruiker, geselecteerde tier en de paren van audio- en annotatiebestanden.

## 2.2 Inlezen xml-bestand

Een cronjob controleert elke 5 minuten of er nieuwe mappen zijn aangemaakt in de uploadfolder op de server, en runt het script op elke nieuwe map: hier begint het offline gedeelte van de tool. Het xml-bestand wordt ingelezen, en vijf variabelen worden opgeslagen voor het vervolg van het script: het e-mailadres van de gebruiker, het absolute pad van de geüploade bestanden, een array van de namen van de audiobestanden, een array van de namen van de annotatiebestanden en het nummer van de tier waarin de fonetische transcripties op woordniveau zijn te vinden. Hierbij geldt dat de lengte van beide arrays even lang is, en nummer $x$ uit de audio-array een paar vormt met nummer $x$ uit de annotatie-array.

## 2.3 Aanmaken corpus

Van alle paren van audio- en annotatiebestanden wordt vervolgens één corpusbestand gemaakt: een tekstbestand met daarin voor alle geluidsbestanden de begin- en eindtijd van elk (fonetisch geschreven) woord. Het script krijgt een array van audiobestanden, een array van daarmee corresponderende annotatiebestanden en het nummer van de te beschouwen tier, en levert een corpus op.

## 2.4 Aanmaken lexicon

Van het corpusbestand kan vervolgens eenvoudig een lexiconbestand worden gemaakt. Dit is, zoals de naam al doet vermoeden, een soort woordenboek met daarin alle woorden die voorkomen in het corpusbestand, alfabetisch gesorteerd. Het lexicon bevat, in tegenstelling tot het corpus, geen informatie over begin- en eindtijden of in welke geluidsbestanden een woord voorkomt. Tevens komt elk woord in het lexicon maar één keer voor. Het script dat een lexicon aanmaakt krijgt een corpusbestand mee, en levert een lexicon op.

## 2.5 Aanmaken segmentatie

Met als invoer een lexicon, corpus en akoestische modellen, wordt met behulp van een SPRAAK-script genaamd `vitprobs` een segmentatie gemaakt. Deze segmentatie is een oplijning van alle fonemen en hun begin- en eindtijden,

en bevat ook een aantal logaritmen waaruit later de TQE-maat kan worden afgeleid.

### 2.6 Berekenen TQE-scores

De segmentatie wordt tot slot opgesplitst: elk oorspronkelijk audio- en annotatiepaar krijgt één TQE-bestand als resultaat. Hiervoor wordt voor elk foneem de TQE-score berekend (zie later voor meer details). Voor elk paar wordt een textgrid gemaakt met drie tiers: een oplijning op woordniveau (gekopieerd uit het annotatiebestand), een oplijning op foneemniveau (uit de segmentatie gehaald) en de berekende TQE-score op een schaal van 0-100.

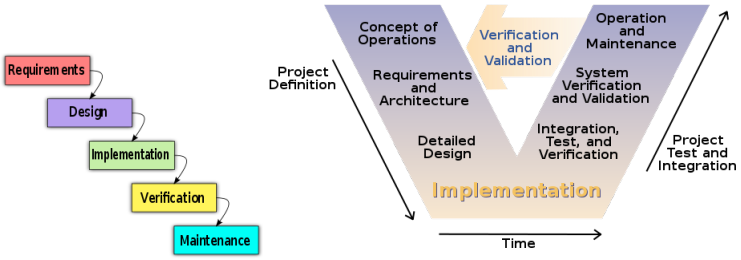
### 2.7 Versturen e-mail

De resultaten worden ge-e-maild naar het door de gebruiker, tijdens het uploaden opgegeven e-mailadres. Indien ergens tijdens het proces een fout is opgetreden, wordt dit script al eerder aangeroepen, en krijgt de gebruiker een foutmelding toegestuurd.

## 3 Ontwikkeling

### 3.1 Model

Het ontwikkelingsproces van de TQE-tool is te beschrijven aan de hand van een software-ontwikkelingsmodel. Hoewel er vanaf het begin van het project sprake was van duidelijke requirements ("deliverables" genoemd), en er duidelijk een ontwerp-, implementatie- en testfase hebben plaatsgevonden, kan het project niet simpelweg worden beschreven door middel van het Waterfall-model ([6], [7]): tijdens elke fase van het project is er continu terugkoppeling geweest van alle betrokkenen, en meerdere malen bleek er iets te moeten worden aangepast in een al afgeronde fase, terwijl het Waterfall-model er juist vanuit gaat dat je, als in een waterval, van boven naar beneden (van requirementsdocument naar werkende applicatie) iets ontwikkelt (zie figuur 2a).
Een variant van het Waterfall-model is het V-model [7], waarbij terugkoppeling plaatsvindt tussen de verschillende onderdelen van de definieerfase enerzijds, en de test- en integratiefase anderzijds (zie figuur 2b). Hoewel hier wel terugkoppeling in voorkomt, is ook het V-model duidelijk niet geschikt om het ontwikkelen van de TQE-tool te beschrijven.




(a) Waterfall-model (b) V-model

Figuur 2: Waterfall- en V-model

Een meer voor de hand liggend model is het spiraalmodel, zoals beschreven door Boehm [8]. Hoewel niet de modernste visie op softwareontwikkeling, is het nog altijd zeer toegankelijk: in tegenstelling tot eerder genoemde "klassieke" modellen, gaat het spiraalmodel meer uit van terugkoppeling, aanpassingen en bijgeschaafde requirements.
Een modernere, en meer op veranderende requirements anticiperende aanpak, is die van Agile Development, naar aanleiding van het Agile Manifesto uit 2001 [9]. In tegenstelling tot bijvoorbeeld de watervalmethode, levert een agile project elke week nieuwe onderdelen van het eindproduct op, waarna de klant weer reageert met eventuele aanpassingen aan de requirements. Het inspelen op veranderende wensen, het continu leveren van bruikbare software en de nauwe samenwerking tussen de ontwikkelaar en de klant, zijn dan ook zeker terug te vinden het TQE-project.

## 3.2 Requirements

Aan het begin van het project zijn er requirements opgesteld, deliverables genoemd. Het gaat hier dan ook enkel om op te leveren producten. De vereiste deliverables, met betrekking tot de tool zelf, zijn als volgt:

1. Software en metadata voor geselecteerde deelverzamelingen van de JASMIN- en CGN-corpora ([10], [11]).

2. Een demonstrator (software), die beschikbaar gesteld wordt via het CLARIN center.

3. Een gebruikershandleiding en een technische documentatie van de tool.

Zoals te zien is, zijn bovenstaande deliverables niet eenduidig of later eenvoudig te controleren: het zijn nog geen echte requirements. Hiervoor

was eerst een behoefte-onderzoek nodig in de vorm van een enquête. De enquête, ook user survey genoemd, is uitgevoerd per e-mail, onder mensen die gebruik maken van corpora en taal- en spraaktechnologie. Uiteindelijk werd de survey door 34 respondenten ingevuld, met de volgende voornaamste conclusies:

1. Het meestgebruikte bestandsformaat voor audiobestanden is WAV: 30 van de 34 respondenten gaf aan dit formaat te gebruiken.

2. Als kanaal wordt vooral mono gebruikt: 31 van de 34 gaven dit aan.

3. 24 respondenten maakten gebruik van bestanden met een opnameprecisie van 16-bit.

4. Voor fonetische transcripties worden zowel IPA als SAMPA veel gebruikt als formaat: resp. 25 en 23 zeiden één van beiden vaak te gebruiken (het geven van meerdere antwoorden was mogelijk).

5. Als software voor het combineren van audio- en annotatiebestanden wordt overwegend PRAAT gebruikt: maar liefst 32 van de 34 respondenten gaf aan dit te gebruiken.

6. 15 mensen gaven de voorkeur aan een automatische segmentatie, boven het zelf meeleveren ervan. Nog eens 6 respondenten wilden wel een automatische segmentatie, maar deze matchen met een meegeleverde segmentatie om fouten eruit te halen.

Als gevolg van deze user survey, was het mogelijk om specificaties en requirements op te stellen voor de TQE-tool:

1. De gebruiker kan audio- en annotatiebestanden, en eventuele segmentaties, uploaden naar de TQE-server:

   (a) Voor het uploaden kan de gebruiker een e-mailadres opgeven, een audio- en een annotatiebestand selecteren met voor de hand liggende knoppen als “bladeren”, en (eindeloos) paren toevoegen, waarna de gebruiker op een “upload”-knop kan drukken. Eventueel kan hij/zij bij elk paar een segmentatie-bestand in txt-formaat selecteren.

   (b) Na het klikken op de upload-knop, wordt direct gecontroleerd of er een e-mailadres is ingevuld, er binnen elk paar zowel een audio- als annotatiebestand geselecteerd is, en of beide bestanden het juiste formaat hebben. Voor audio zijn alleen 16-bits mono

WAV-bestanden toegestaan, terwijl voor de annotaties textgrids met txt- en textgrid-extensies geaccepteerd worden. De gebruiker krijgt te zien of de upload geslaagd is, of dat één of meer geselecteerde bestanden niet aan de gestelde eisen voldoen.

2. Als resultaat levert de TQE-tool per audio- en annotatiepaar één TQE-bestand op: dit is een textgridbestand dat te openen is met PRAAT, en per foneem een TQE-score weergeeft. Deze score is een (geheel) getal tussen 0 en 100, waarbij 0 een zeer slechte correlatie tussen het gesproken en het geannoteerde foneem weergeeft, en 100 een uitstekende. De gebruiker krijgt vervolgens op het door hem/haar opgegeven e-mailadres een bericht dat de resultaten gedownload kunnen worden van de server.

3. De tool wordt gehost door het Max Planck Institute, en beschikbaar voor instanties die een link hebben met taalwetenschappen en/of één van de betrokken partijen binnen het project.

4. Als demonstratie zal de TQE-tool gerund worden op deelverzamelingen van de JASMIN- en CGN-corpora ([10], [11]).

5. Er moet een gebruikershandleiding komen, met daarin uitleg over hoe de tool gebruikt dient te worden, en wat de resultaten voor betekenis hebben.

6. Er moet een API-documentatie komen, met daarin (onder meer een technische) uitleg over de verschillende onderdelen van de tool. Deze documentatie dient gebruikt te worden bij toekomstige aanpassingen aan de tool, en kan geraadpleegd worden bij onverhoopte problemen.

NB: de requirements hebben enkel betrekking op de input en output van de tool, en niet op het proces daartussen. Dit proces is al globaal beschreven in de inleiding, en zal nader bekeken worden in een later hoofdstuk.

### 3.2.1 Aanpassingen

Tijdens het ontwikkelen van de tool, zijn er verschillende dingen toegevoegd en veranderd. Dit zijn grotendeels verdere, meer gedetailleerde toevoegingen aan de genoemde requirements, maar in sommige gevallen ook daadwerkelijke aanpassingen aan de eerder gestelde eisen. Deze toevoegingen en aanpassingen zal ik hier kort bespreken en motiveren:

1. De gebruiker krijgt niet de mogelijkheid een bestaande segmentatie aan te leveren: de TQE-tool zal dus zelf een automatisch gegenereerde

segmentatie gebruiken als uitlijning in de resulterende TQE-bestanden. Een reden hiervoor is dat er veel verschillende soorten segmentaties zijn, en het veel tijd kost deze te converteren (indien mogelijk) naar het gewenste type. Ook wordt bij het berekenen van de TQE-scores sowieso al een segmentatie gegenereerd door de `vitprobs`-module.

2. De bestanden worden niet bewaard op de server: de gebruiker krijgt dus geen e-mail met daarin een downloadlink, maar direct de resultaten toegestuurd. Dit heeft twee voordelen: het bespaart ruimte op de server, en tijd voor de gebruiker.

3. De deelverzamelingen van de JASMIN- en CGN-corpora ([10], [11]) zijn gekozen: van het JASMIN [10] wordt component Q gebruikt, en van het CGN [11] component O. Deze keuzes liggen redelijk voor de hand omdat beide componenten voorgelezen spraak bevatten, en daardoor naar verhouding goede transcripties zullen bevatten: ter demonstratie is het eenvoudig om kleine foutjes te introduceren en de lagere scores aan te wijzen ten opzichte van die van de originele data.

## 3.3 Ontwerp

Na het helder krijgen van de requirements kon een eerste ontwerp van de tool gemaakt worden. Dit ontwerp bevat al een helder beeld de verschillende onderdelen van de tool, en wat de input en output van de verschillende onderdelen moet zijn. Het ontwerp is als volgt te beschrijven, en het resultaat hiervan is weergegeven in figuur 3:

1. De applicatie krijgt paren van audio- en annotatiebestanden, en eventueel een meegeleverde segmentatie.

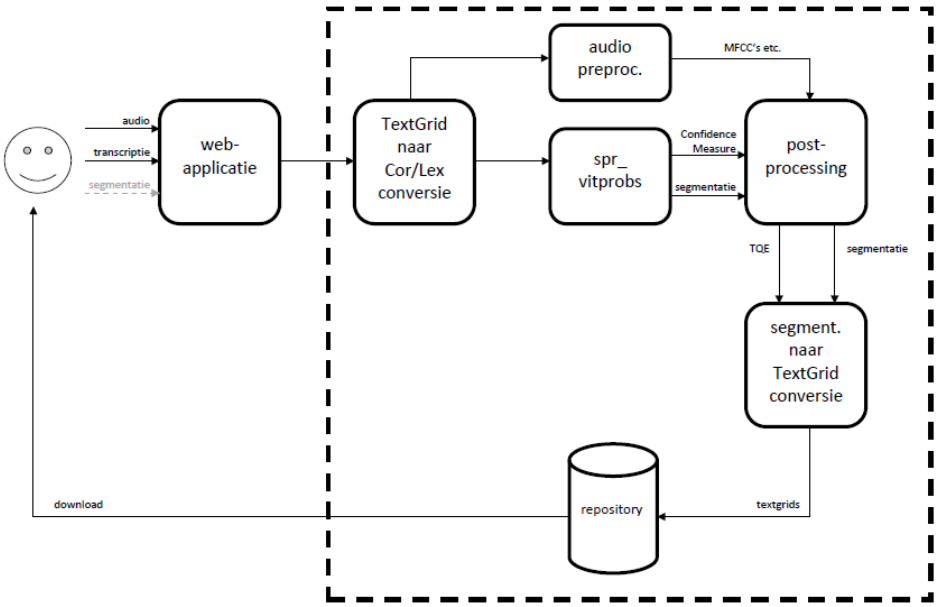
2. De geüploade annotatiebestanden worden geconverteerd naar één corpusbestand, waaruit vervolgens ook een lexiconbestand wordt gegenereerd.

3. Uit de audiobestanden worden **mel-frequency cepstral coefficients** gehaald, die de audiosignalen representeren als getallen.

4. Het corpus en lexicon worden gebruikt voor het berekenen van de confidence measures en het aanmaken van een segmentatie, door middel van `vitprobs`.

5. Met behulp van akoestische modellen kan de segmentatie nu vergeleken worden met de MFCC's, waaruit de TQE-scores bepaald worden.

6. De aangemaakte segmentatie wordt, aangevuld met de TQE-scores, geconverteerd naar het PRAAT textgrid-formaat, en de gebruiker kan de resultaten downloaden van de server.




Figuur 3: Oorspronkelijk ontwerp van de TQE-tool

### 3.3.1 Aanpassingen

Ook aan het ontwerp zijn elementen aangepast en toegevoegd, die ik hier kort zal toelichten en motiveren:

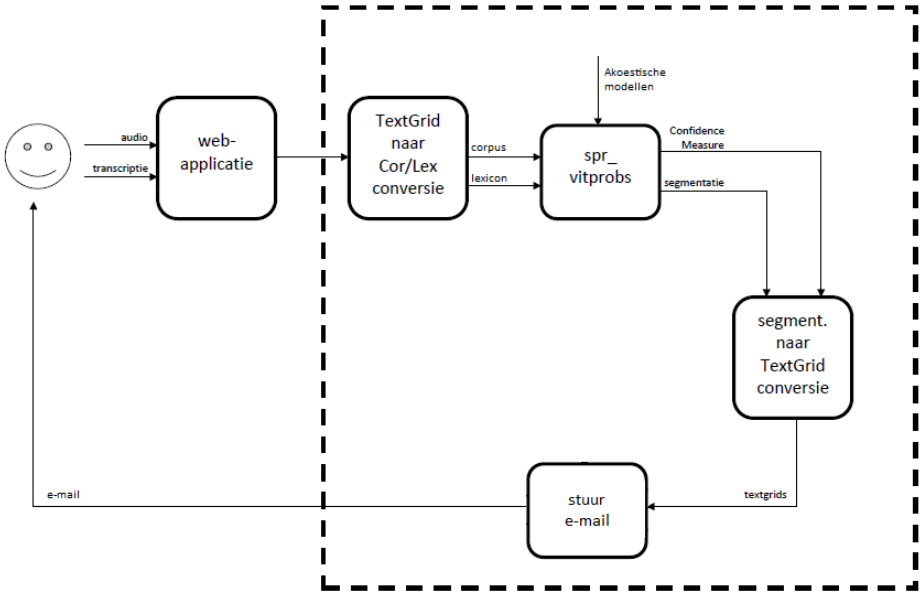
1. De applicatie krijgt geen meegeleverde segmentatie: dit is een gevolg van de eerder genoemde aanpassing aan de requirement dat de gebruiker eventueel een segmentatie zelf kon aanleveren.

2. Er wordt geen gebruik gemaakt van MFCC's: er bleken genoeg bestaande akoestische modellen te bestaan, getraind op Nederlandse spraak, en beschikbaar voor het berekenen van de confidence measures met `vitprobs`. Hierdoor vervallen binnen het ontwerp (zie figuur 3) de preprocessing- en postprocessingtoestand.

3. De gebruiker kan geen bestanden downloaden van de server, maar krijgt direct een mail met bestanden: ook dit is een gevolg van een veranderde requirement, en effectiever in de praktijk.

4. De grootste aanpassing aan het ontwerp is de plaats van handelen: in het oorspronkelijke ontwerp vindt alles sequentieel plaats, waarbij na het uploaden een textgrid-naar-corpus-conversie wordt aangeroepen, en uiteindelijk de bestanden in een repository terechtkomen. Nu worden de bestanden bij het uploaden op een server gezet, waar een crontab een soort "masterscript" aanroept per upload, dat op haar beurt sequentieel de genoemde scripts uitvoert. Essentieel verschil is dus dat alle bestanden al direct op de server staan, en het proces na het uploaden geheel offline plaatsvindt.

Deze aanpassingen zijn terug te zien in figuur 4. Hierin vindt, zoals in punt 4 hierboven genoemd, het offline gedeelte (tussen de stippellijnen) volledig plaats op de server.




Figuur 4: Aangepast ontwerp van de TQE-tool

## 3.4 Uitbreidingsmogelijkheden

Hoewel niet expliciet geformuleerd als requirement, dient de tool ook na oplevering makkelijk aanpasbaar en uitbreidbaar te zijn. Een mogelijke reden hiervoor kan zijn dat in de toekomst een ander fonetisch alfabet de standaard zal worden, of er verschillende soorten audiobestanden automatisch door de tool geconverteerd moeten kunnen worden naar het door `vitprobs` gewenste

formaat. Om dergelijke aanpassingen mogelijk te maken, moet de tool zo ontwikkeld zijn dat een variabele niet in bijvoorbeeld vijf verschillende scripts aangepast moet worden: er moeten zoveel mogelijk parameters globaal worden bijgehouden. Een eenvoudig voorbeeld van een dergelijke parameter is de locatie van de uploadfolder. Omdat tijdens het project op meerdere servers is getest, is deze locatie meerdere malen veranderd. Het is dan niet wenselijk om in de code deze locatie steeds aan te moeten passen, en daarom staat dit soort parameters opgeslagen in een apart configuratiebestand. Ook staan alle subprocessen, zoals genoemd in de inleiding, in aparte scripts, die worden aangeroepen vanuit één hoofdscript. Hierdoor is het eenvoudig een eventuele tussenstap toe te voegen, door een nieuw script te schrijven en dit op de gewenste plek aan te roepen in het masterscript.

## 3.5 Implementatie

### 3.5.1 Python

Als programmeertaal voor de implementatie is, op aanraden van het TQE-projectteam, gekozen voor Python [12]. Dit heeft een aantal voordelen:

- Er zijn talloze modules beschikbaar voor Python. Zo heb ik bijvoorbeeld bij het berekenen van de scores gebruikgemaakt van NumPy [13], waarmee eenvoudig histogrammen en dergelijke gemaakt kunnen worden.

- Zoals te zien is in de pseudocode, is het gebruik van exception handling erg eenvoudig: Python gebruikt zogenaamde *Easier to Ask Forgiveness than Permission*-procedures hiervoor, waardoor je geen checks hoeft uit te voeren of iets wel kan (bijvoorbeeld nuldelingen) met behulp van if-then-else-statements, maar simpelweg try-except-clauses eromheen hoeft te zetten.

- Python is een scripttaal, hetgeen een hoop voordelen met zich meebrengt: zo hoeven variabelen niet te worden gedeclareerd en zijn reguliere expressies toegestaan. Dit komt goed uit in de TQE-tool, waarin veel met (lange) tekstbestanden wordt gewerkt, zoals bijvoorbeeld bij het converteren van meerdere textgrids naar een corpusbestand.

- Python integreert makkelijk in een Linux-omgeving, waar ook de server op draait. Daarnaast worden Pythonscripts pas runtime gecompileerd. Dit maakt het makkelijk om een crontab te runnen die een Pythonscript aanroept.

- Python is intuïtiever dan bijvoorbeeld Java of C, en indien nodig is er een uitgebreide tutorial [2] online die alles perfect uitlegt.

### 3.5.2 Pseudocode

Het ontwikkelen van de code is grotendeels volgens het ontwerpschema in figuur 3 gegaan. Dat wil zeggen dat ik begonnen ben met het script dat textgrids converteert naar een corpusbestand (`txt2cor.py`), gevolgd door het script dat hieruit een lexiconbestand aanmaakt (`cor2lex.py`). Het aanroepen van `vitprobs` doe ik ook vanuit een apart script, zoals al vermeld in de inleiding. Het berekenen van de TQE-scores uit de segmentatie heb ik vervolgens geprogrammeerd in `seg2cm.py`. Op dit moment zag het totaalplaatje er nog warrig uit, met veel parameters gedefinieerd in de code zelf. Het `txt2cor`-script ging er nog standaard van uit dat de fonetische transcriptie in de tweede tier stond, dus veel variabelen werden nog niet doorgegeven. Bijvoorbeeld (pseudocode):

```
annotationfolder = '/uploads/annotations/'
audiofolder = '/uploads/audio/'
spraakpath = '/spraak/bin/'

corpus = txt2cor (annotationfolder)
lexicon = cor2lex (corpus)
segmentatie = vitprobs (spraakpath, corpus,
        lexicon, audiofolder)

tqe-scores = seg2cm (segmentatie)
```

Zoals eerder gezegd, werden sommige wensen pas later in het onwikkelproces duidelijk. Een voorbeeld hiervan is het gebruik van een XML-bestand bij elke upload, waarin alle gegevens over de bestanden te vinden zijn. De verwijzing naar het pad van SPRAAK kan natuurlijk beter buiten het script worden opgeslagen, ook met het oog op de genoemde wensen om eenvoudig elementen aan te kunnen passen. In het volgende voorbeeld wordt `spraakpath` meegegeven aan het script:

```
email, audiofolder, annotationfolder, tier = readXML
corpus = txt2cor (annotationfolder)
lexicon = cor2lex (corpus)
segmentatie = vitprobs (spraakpath, corpus,
        lexicon, audiofolder)
```

[2] The Python Tutorial - http://docs.python.org/tutorial/

```
tqe-scores = seg2cm (segmentatie)
sendmail (email, tqe-scores)
```

Daarmee was het "hoofdscript" bijna af. Tijdens de daadwerkelijke uitvoering ervan is het handig om de bestanden tijdelijk op te slaan in een soort "progress"-folder, en na afloop te verplaatsen naar een "finished"-folder, om op de server eenvoudig te kunnen zien welke bestanden al verwerkt zijn. Daarnaast is besloten om een duidelijke log bij te houden van het runnen van de tool, en ervoor te zorgen dat het script er niet voortijdig mee ophoudt als er onverhoopt fouten optreden: zelfs voor de beste software is foutafhandeling een essentieel onderdeel van het geheel. Het "hoofdscript" ziet er uiteindelijk dan als volgt uit, in pseudocode:

```
try:
        log 'lees config file'
        email, audiofolder, annotationfolder,
                tier = readXML
except:
        log exception
        return

hernoem_folder ('_inprogress')

try:
        log 'maak corpus file'
        corpus = txt2cor (annotationfolder)
except:
        log exception
        sendmail(email, error)
        return

try:
        log 'maak lexicon file'
        lexicon = cor2lex (corpus)
except:
        log exception
        sendmail(email, error)
        return

try:
        log 'maak segmentatie file'
```

```
        segmentatie = vitprobs (spraakpath, corpus,
                lexicon, audiofolder)
except:
        log exception
        sendmail(email, error)
        return

try:
        log 'bereken TQE-scores'
        tqe-scores = seg2cm (segmentatie)
except:
        log exception
        sendmail(email, error)
        return

hernoem_folder ('_finished')

try:
        log 'stuur e-mail naar gebruiker'
        sendmail (email, tqe-scores)
except:
        log exception
        return

verwijder_oude_bestanden
```

# 4 Data

Voor het implementeren van de TQE-tool heb ik gebruik gemaakt van data van het Corpus Gesproken Nederlands, kortweg CGN [11]. Met een gekozen subset hiervan heb ik de verschillende transformaties per foneem bepaald (zie volgend hoofdstuk). Voor het gebruik van de SPRAAK-module `vitprobs` heb ik bepaalde akoestische modellen gebruikt.

## 4.1 CGN

Het Corpus Gesproken Nederlands [11] is een grote dataset van Nederlandse en Vlaamse spraak. Van de website:

> Het project Corpus Gesproken Nederlands was gericht op de aanleg van een databank van het hedendaags Nederlands zoals

dat wordt gesproken door volwassenen in Nederland en Vlaanderen. Bij de start van het project werd een corpus beoogd met een omvang van circa tien miljoen woorden, waarvan tweederde deel afkomstig zou zijn uit Nederland, en eenderde uit Vlaanderen. In totaal ging het daarbij om circa 1000 uur spraak. Het eindresultaat zoals beschikbaar in versie 1.0 omvat ongeveer 9 miljoen woorden: zo'n 3,3 miljoen woorden daarvan zijn afkomstig uit Vlaanderen, ruim 5,6 miljoen woorden werden opgenomen in Nederland.

Het Corpus Gesproken Nederlands wordt gevormd door een selectie van een groot aantal fragmenten van spraakopnames. Al het materiaal werd orthografisch getranscribeerd, terwijl er tevens een oplijning plaatsvond waarbij de orthografische transcriptie gekoppeld werd aan het spraaksignaal. De orthografische transcriptie vormde het uitgangspunt voor de lemmatisering en de verrijking van het materiaal met woordsoortinformatie. Verder werd er voor een selectie van n miljoen woorden een brede fonetische transcriptie vervaardigd, kwam er een geverifieerde oplijning op woordniveau beschikbaar en werd het materiaal door middel van een syntactische analyse verrijkt. Tenslotte werd een bescheiden deel van het corpus, circa 250.000 woorden, van een prosodische annotatie voorzien. [3]

### 4.1.1 Trainingsset

Het gehele CGN bevat zoals gezegd circa 1000 uur spraak, en is gedistribueerd over 33 dvd's. Voor de TQE-tool is gekozen om alleen gebruik te maken van component-O, die volledig bestaat uit voorgelezen teksten. Deze database bevat 903.043 woorden, waarvan 551.624 Nederlandse en 351.419 Vlaamse, en neemt $6,89$ gigabyte in aan audio-opnames. De TQE-tool hier één keer op laten draaien zou ongeveer 26 uur in beslag nemen. Voor het bepalen van de transformatie per foneem zou de tool dan anderhalve maand bezig zijn, wat nog steeds gekkenwerk is. Daarom heb ik een representatieve subset gemaakt van component-O, gebaseerd op geslacht, leeftijd en moedertaal van de spreker. Het voordeel van het gebruik van één component, is dat alle sprekers Algemeen Beschaafd Nederlands spreken, en alles voorgelezen spraak is (zelfde kanaal, signaalsterkte, verstaanbaarheid etc.). Uiteindelijk heb ik een dataset van 101 megabyte verdeeld over acht audio-opnames geselecteerd: vier mannelijke en vier vrouwelijke sprekers, beide onder te verdelen in vier

[3] bron: http://lands.let.kun.nl/cgn/doc_Dutch/topics/project/pro_info.htm

leeftijdscategorieën. Deze subset gebruik ik als trainingsset voor het bepalen van de transformaties, zoals besproken in een later hoofdstuk. Voor het testen maak ik daarnaast gewoon gebruik van willekeurige opnames uit component-O. Dit volgt op het principe om de trainings- en testset gescheiden te houden: cross-validatie [14].

## 4.2 JASMIN

Naast het Corpus Gesproken Nederlands, heb ik de tool ook getest op het JASMIN-corpus [10]. Het JASMIN (*J*ongeren, *A*nderstaligen, *S*enioren en *M*achine *I*nteractie voor het *N*ederlands) begon in 2005 als een project vanuit de taalwetenschapsafdelingen aan de universiteiten van Nijmegen en Leuven, en het bedrijf TalkingHome, dat zich vooral richt op de zorg. Omdat het CGN, mede door beperkte budgetten, zich slechts concentreert op volwassen autochtonen uit Nederland en Vlaanderen, was een uitbreiding gewenst. Het JASMIN vormt een dergelijke uitbreiding: het corpus bestaat uit gesproken Nederlands, door kinderen van verschillende leeftijden, anderstaligen met verschillende moedertalen, en ouderen uit Nederland en Vlaanderen. Daarnaast is er in het project oog geweest voor interactie tussen mens en machine.

## 4.3 Akoestische modellen

Voor het creëren van een segmentatie heeft SPRAAK [15] meer informatie nodig dan alleen de invoerbestanden: om `vitprobs` een succesvolle oplijning te laten genereren, zijn tevens akoestische modellen nodig. Voor spraakherkenning is het gebruik van Hidden Markov Models een voor de hand liggende keuze, zoals Rabiner in 1989 [16] al schreef. Een HMM bestaat uit toestanden, bezigheden, en waarschijnlijkheden voor elk van deze toestanden, de overgangen tussen de toestanden en het voorkomen van een bezigheid in een bepaalde toestand. Binnen SPRAAK [15] worden HMM's beschreven als twee verzamelingen: een verzameling toestandsonafhankelijke basisfuncties in de vorm van multivariate Gaussische verdelingen, en een verzameling gewichten die de basisfuncties aan de toestanden koppelt. Voor de TQE-tool zijn de standaard HMM's gebruikt.

# 5 Scores

Het berekenen van de scores gebeurt per foneem verschillend. Dit is nodig omdat de foutvariatie bij sommige fonemen veel groter is dan bij andere, terwijl het gewenst is dat een score van 10 voor een *a* dezelfde indicatie geeft

als een zelfde score voor een $b$. De scores worden berekend op basis van de segmentatie die gemaakt is door de `vitprobs`-module van SPRAAK. Deze segmentatie wordt hieronder eerst nader bekeken, waarna uitleg volgt over de gekozen optimalisatie.

## 5.1 Segmentatie

De segmentatie is een tekstbestand met regels van de volgende vorm ([15]):

```
<fname> <state> <f0> <nfr> <l_0> ... <l_(nfr-1)> <t0> <t1>
```

Met daarin:

**fname** de bestandsnaam van het audiobestand, of een '-' als het huidige segment van hetzelfde foneem is als het vorige.

**state** de HMM-state, bestaande uit het foneem gevolgd door een hekje en het toestandsnummer. Elk foneem heeft drie segmenten.

**f0** de index van het eerste frame van het huidige segment. Het eerste segment van het eerste foneem van elk geluidsbestand heeft altijd index 0.

**nfr** het aantal frames waaruit het huidige segment bestaat.

**l_i** een "normalized frame likelihood" per frame waaruit het huidige segment bestaat. Deze worden berekend als $l_i = f(x| < u > \# < nr >)/f(x)$, met $f(x)$ geschat als de som van $f(x|s)$ en de onvoorwaardelijke toestandskans $P(s)$. Deze likelihoods zijn logaritmen log(10).

**t0, t1** de "transition probabilities" van respectievelijk het binnengaan en verlaten van het huidige segment.

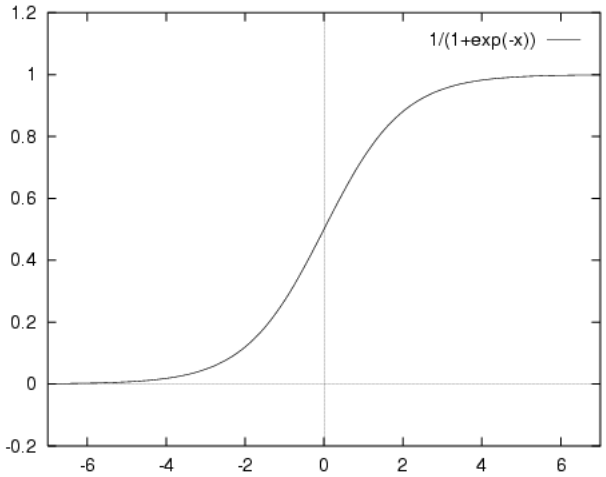
## 5.2 Transformatie

Door het gemiddelde te nemen van alle likelihoods per foneem, blijft er één (logaritmische) waarde over. Het is dan zaak om een transformatie te bepalen van dit logaritme (een waarde grofweg tussen $-8$ en 4) naar een score tussen 0 en 100. Deze transformatie kan op verschillende manieren gemaakt worden: bijvoorbeeld door middel van een exponentiële formule. Uiteindelijk heb ik gekozen voor een sigmoïde, en ik leg hieronder uit waarom.

### 5.2.1 Sigmoïde

Een sigmoïde is een formule van de vorm $y = \frac{1}{1+e^{a*x+b}}$. Een sigmoïde wordt gekenmerkt door een S-vormige curve (zie figuur 5), en gebruikt voor functies met een continue uitkomst in plaats van een ja/nee-uitkomst, een zogenaamde threshold-functie. Een sigmoïde kan een dergelijke threshold-functie benaderen door $a$ aan te passen: een *grotere* (absolute) waarde van $a$ zorgt voor een *korter* steil gebied op de grafiek, terwijl een *lagere* (absolute) waarde van $a$ juist tot een meer lineair-achtige grafiek leidt. Door $b$ aan te passen verander je de "threshold" van de sigmoïde (tussen aanhalingstekens omdat het bij een sigmoïde natuurlijk om een continue functie gaat, en er geen verwarring dient te onstaan met een threshold-functie die maar twee uitkomsten kan opleveren), en wordt de grafiek horizontaal verschoven.

Als je nogmaals naar de voorbeeldgrafiek kijkt (figuur 5), dan zie je meteen waarom dit een geschikte functie is voor de scoreverdeling: een getranscribeerd foneem kan vooral "goed" of "fout" zijn, maar een score van 0 of 100 is weer te zwart-wit. Terug naar de formule: in het geval van de TQE-maat is $y$ de op te leveren score, $x$ de rauwe logaritmische score, en zijn $a$ en $b$ foneemafhankelijke constanten. Voor het bepalen van de foneemafhankelijke transformaties, moeten dus per foneem $a$ en $b$ gevonden worden.



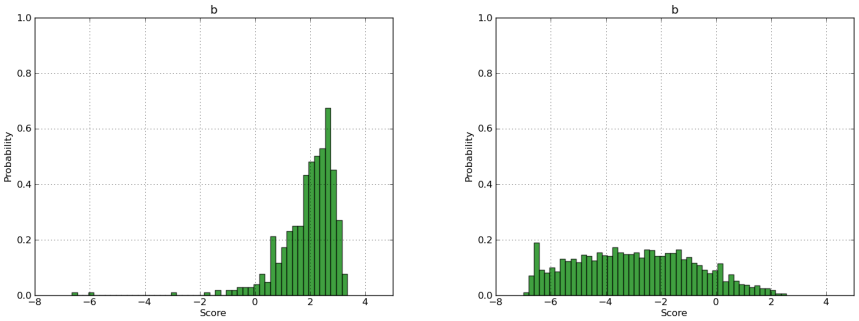

Figuur 5: Voorbeeld van een sigmoïde (bron: [17])

### 5.2.2 Buurklankmethode

Om per foneem een transformatie te vinden, heb ik kunstmatige fouten geïntroduceerd in de transcripties, om vervolgens de histogrammen van de "echte" scores te vergelijken met de "foute" scores, en hieruit de sigmoïde

af te leiden. Om het proces te kunnen uitleggen, neem ik als voorbeeld het foneem *b*. Aangezien het waarschijnlijker is dat een *b* ten onrechte als een *p* getranscribeerd zal worden dan als *z*, voer ik alleen fouten in bij fonemen uit dezelfde fonetische klasse als de *b*. In dit geval zijn dat de occlusieven: naast de *b* bestaat deze klasse uit de fonemen *p*, *k*, *g* (als in *goal*), *d* en *t*. Zie appendix A voor een volledig overzicht van de verschillende fonetische klassen. Van al deze fonemen, behalve de *b* zelf, heb ik de helft vervangen door een *b*, en daarnaast heb ik alle orginele voorkomens van *b* vervangen door zijn *naaste buurklank*, in dit geval de *p*. Dit doe ik om de oplijning zelf zo intact mogelijk te laten: als ik de *b* hier zou vervangen door een *o*, dan zou de oplijning sterk veranderen, en de scores onbetrouwbaarder worden. Tot slot run ik het hele script op deze "foute" transcriptie, en maak ik een histogram van de scoreverdeling van de *b*.

Dit proces heb ik voor alle fonemen herhaald, en zo heb ik voor elk foneem een "goed" en een "fout" histogram: hieruit kan worden afgeleid wat normale scores zijn, en wat dus richting 100 getransformeerd dient te worden, en wat slechte scores zijn, en dus lagere TQE-maten moet opleveren. Zie als voorbeeld beide histogrammen van de *b* in figuur 6.




(a) Originele scores (b) Na invoering fouten

Figuur 6: Scoreverdeling van de *b* zonder en met fouten

Voor het bepalen van de formule, heb ik vervolgens alle rauwe scores geclassificeerd: alle "goede" scores krijgen label 1, en alle "foute" scores label 0. Al deze scores en hun klasselabel vormen samen de trainingset voor de transformatie van de *b*. Een deel van deze trainingset ziet er als volgt uit:

```
2.462957 1
1.708550 1
2.649600 1
1.844933 1
```

```
2.675825 1
-5.185933 0
-1.391720 0
-4.011367 0
-5.083200 0
-5.500625 0
```

### 5.2.3 Logistische regressie

Voor het bepalen van de formule, maak ik gebruik van logistische regressie, hetgeen een sigmoïde oplevert. Zie voor de motivatie van het gebruik van sigmoïdes paragraaf 5.2.1. Met logistische regressie wordt de *uitkomstvariabele*, in dit geval de TQE-score, bepaald aan de hand van de volgende vergelijking:

$$\ln(y) = a + b_1X_1 + b_2X_2 + ... + b_kX_k$$

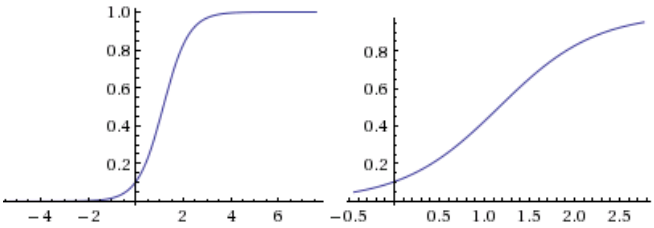
De vergelijking levert dus niet de uitkomstvariabele zelf, maar het natuurlijk logaritme hiervan op. In het geval van de TQE-score is er maar één variabele $X$, namelijk de rauwe score: alleen constante $a$ en logistisch regressiecoëfficient $b$ moeten gevonden worden. Hiervoor maak ik gebruik van de WEKA-toolkit [18]. In het geval van de $b$ zijn deze parameters $-1.8661$ en 2.1678, waardoor de transformatie van de rauwe score $r$ naar de TQE-maat $TQE_b$ als volgt gaat:

$$TQE_b = 100 * \frac{1}{1+e^{-1.8661*r+2.1678}}$$

Als we deze transformatie nader bekijken in een grafiek, is goed het idee achter het gebruik van een sigmoïde zichtbaar (zie vorige paragraaf). Figuur 7a laat de volledige grafiek zien, en figuur 7b is ingezoomd op het "steile" deel (NB: de vermenigvuldiging met 100 ten behoeve van een schaal van 0-100 is nog niet verwerkt in de grafiek. Voor het genereren van de grafieken heb ik gebruikgemaakt van Wolfram|Alpha [19]).

# 6 Resultaten

Het gebruik van een aparte transformatie voor elk foneem werpt duidelijk zijn vruchten af. Voor fonemen die van zichzelf lastiger te herkennen zijn, en vaak lagere scores krijgen, is hier en daar zichtbaar de score omhooggegaan. De belangrijkste voorbeelden hiervan zijn de $w$, $h$, $r$ en $j$: fonemen die vaak enkel

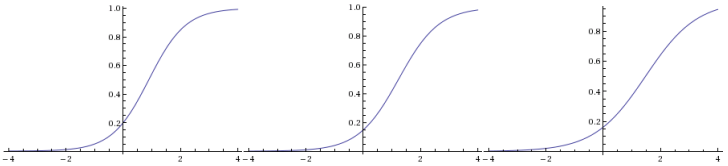



(a) Sigmoïde van de transformatie van de $b$ (b) Deel van de sigmoïde

Figuur 7: Transformatie van de rauwe score naar een TQE-maat van de $b$

kort "geademd" worden, en afzonderlijk bijna niet te herkennen zijn als het betreffende foneem. Deze fonemen scoren met de Buurklankmethode vaker richting de 100 op plaatsen waar de rauwe score rond de 50 ligt. Fonemen die slechter scoren zijn vooral klinkers: waar de rauwe score nabij 100 ligt, houdt de Buurklankmethode het vaak bij 80, en een rauwe score van 50 wordt al snel naar een hele lage TQE-maat getransformeerd. Dit heeft te maken met de steilere sigmoïde voor klinkers, omdat deze vaak eenvoudiger te herkennen zijn: klinkers worden vaak beklemtoond, en bovendien langer uitgesproken dan medeklinkers.

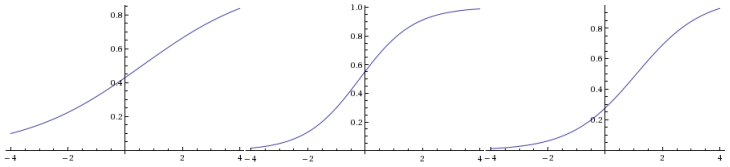
Deze bevindingen kunnen het best aangetoond worden aan de hand van enkele grafieken. Eerst het verschil tussen de scoreverdelingen van klinkers enerzijds, en medeklinkers anderzijds. In figuur 8 staan de transformaties van rauwe score naar TQE-maat (schaal 0.0 tot 1.0) van de fonemen $A$, $E$ en $I$ [4]. Als we deze verdelingen met drie willekeurige medeklinkers ($d$, $s$ en $m$) vergelijken (zie figuur 9), dan valt meteen op dat de medeklinkers een *langer* overgangsgedeelte hebben. Dit bevestigt het genoemde vermoeden dat het voor medeklinkers moeilijker te bepalen is of hij "goed" of "fout" is getranscribeerd.



(a) $A$ (b) $E$ (c) $I$

Figuur 8: Transformaties van klinkers van rauwe score naar TQE-maat

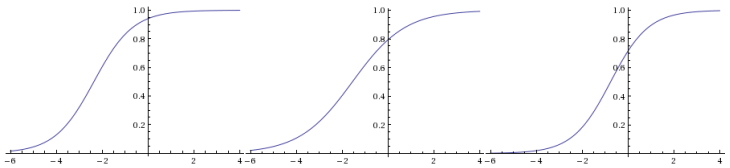
[4]gecapitaliseerde fonemen geven een korte vocaal aan: de $A$ wordt uitgesproken als in *bak*, en de $a$ als in *zaal*



(a) $d$ (b) $s$ (c) $m$

Figuur 9: Transformaties van medeklinkers van rauwe score naar TQE-maat

Het tweede punt dat ik al eerder noemde, was de verandering in de scores van de $w$, $r$ en $j$. Tijd om van deze fonemen de sigmoïden nader te bekijken, zie figuur 10. Onmiddellijk valt de positie van de sigmoïde op: in vergelijking met de eerdere sigmoïdes, staan die van de $w$, $r$ en $j$ verder naar links (merk op dat ik dit keer gekozen heb voor een plot van $r = -6.0$ tot $r = 4.0$). Dit betekent dat een lage rauwe score relatief sneller een hoge TQE-maat oplevert, zoals ik al vermoedde.



(a) $w$ (b) $r$ (c) $j$

Figuur 10: Transformaties van $w$, $r$ en $j$ van rauwe score naar TQE-maat

# 7 Conclusie

De hoofdvraag van mijn onderzoek luidde hoe de kwaliteit van fonetische transcripties zo goed mogelijk geëvalueerd zou kunnen worden. Om transcripties zo goed mogelijk te kunnen evalueren heb ik de *Transcription Quality Evaluation*-tool, kortweg TQE-tool, ontwikkeld. Het ontwikkelproces paste in het gedachtegoed van *agile development*, waarin requirements tijdens het proces veranderen naar aanleiding van nieuwe eisen of punten die tijdens de ontwikkeling aan het licht komen. Zo werd tijdens de ontwikkeling onder meer besloten om de gebruiker niet de optie te geven een segmentatie mee te geven bij het uploaden, en de gebruiker een e-mail te sturen na afloop in plaats van de bestanden ter beschikking te stellen via een downloadlink.

Na de ontwikkeling van de tool kon ik beginnen met het optimaliseren van de scores. Voor de foneemafhankelijke transformaties verving ik fonemen uit bepaalde fonetische klassen steeds door één specifiek foneem. Het gekozen foneem zelf verving ik dan door een buurklank om de oplijning goed te houden, waardoor ik deze methode de naam *buurklankmethode* heb gegeven. Voor elk van deze fonemen heb ik vervolgens de TQE-tool gerund tot en met de segmentatie: dit leverde rauwe (niet getransformeerde) scores op, die ik gelabeld heb met "goed" of "fout" (als 1 en 0). Door per foneem deze scores en hun bijbehorende labels te analyseren, door middel van logistische regressie, kon ik voor elk foneem een sigmoïde bepalen die de rauwe score naar een TQE-maat transformeert.

Vergelijking van de scoreverdelingen van verschillende fonemen, in de vorm van sigmoïdes, liet een interessant verschil zien tussen sommige fonemen. Het gebruik van foneemafhankelijke transformatie leverde bij sommige fonemen een zichtbare verbetering op: de scores van bijvoorbeeld de $w$, $r$ en $j$ zijn beduidend betrouwbaarder geworden. Daarnaast is het nut van foneemafhankelijke transformaties bewezen aan de hand van de verschillende scoreverdelingen van klinkers en medeklinkers.

Kortom, fonetische transcripties kunnen met behulp van de juiste akoestische modellen, goed gekalibreerde foneemafhankelijke transformaties, een SPRAAK-module en een aantal Pythonscripts goed geëvalueerd worden. Er is daarnaast nog voldoende ruimte voor verbetering en toekomstig onderzoek, zoals ik in de volgende sectie beschrijf.

## 7.1 Discussie en toekomstig onderzoek

Voor het optimaliseren van de transformaties heb ik twee keuzes gemaakt die een invloed op de uitkomsten gehad kunnen hebben. Ten eerste koos ik ervoor om slechts een kleine subset van het CGN te gebruiken, om zo

niet het script anderhalve maand te hoeven laten runnen, nog los van de capaciteitsproblemen die waarschijnlijk zouden ontstaan op de server. Door een subset te nemen is het aantal voorkomens van een aantal fonemen echter wel iets aan de lage kant: sommige wat zeldzamere fonemen, zoals de *nj* in *oranje*, zijn hierdoor op een beperkt aantal scores getraind, waardoor de transformatie mogelijk niet optimaal is. Een toekomstig project zou het proces van de Buurklankmethode kunnen herhalen voor álle bestanden uit het component-O van het CGN, om dit verder te optimaliseren.

Een tweede punt is, dat ik in de Buurklankmethode nadrukkelijk ervoor gekozen heb om alleen fonemen uit dezelfde fonetische klassen onderling te verwisselen. Een idee voor de toekomst zou kunnen zijn om alle fonemen onderling te vervangen: voor elk annotatiebestand uit de dataset wordt dan voor elk foneem $x$, voor alle fonemen een nieuw bestand gemaakt waarin deze fonemen vervangen zijn door foneem $x$. Als de foneemset uit $n$ fonemen bestaat, levert dat dus $n \times (n-1) = n^2 - n$ annotaties per oorspronkelijke annotatie op, een gigantische explosie van data. De door mij gebruikte foneemset bestaat uit 39 verschillende fonemen, dus dat zou leiden tot 1482 annotaties met kunstmatige fouten, per oorspronkelijke annotatie, en bij gebruik van de door mij samengestelde subset tot 11856 annotaties in totaal. Dit betekent dat de TQE-tool 1482 keer zou moeten runnen, een idiote gedachte voor de simpele server die gebruikt is voor de ontwikkeling van de TQE-tool, maar niet onmogelijk op bijvoorbeeld een MOSIX-cluster [20]. Met de rauwe scores die dit hele proces oplevert, kunnen nauwkeurige voorspellingen gedaan worden: een $e$ die een $l$ vervangt kan bijvoorbeeld nog steeds geclassificeerd worden als 0, maar een kunstmatige $e$ op de plek waar eigenlijk een $a$ gezegd wordt is "minder fout" en kan daarom geclassificeerd worden als 0.5 (of een andere onderbouwde classificatie). Op deze manier kan de transformatie nauwkeuriger worden afgestemd, en kan er meer rekening gehouden worden met de aard van de transcriptiefout.

# Referenties


[1] C. Cucchiarini, A. Neri, and H. Strik. Oral proficiency training in Dutch L2: The contribution of ASR-based corrective feedback. *Speech Communication*, 51(10):853–863, 2009.

[2] H. Strik, K. Truong, F. de Wet, and C. Cucchiarini. Comparing different approaches for automatic pronunciation error detection. *Speech Communication*, 51(10):845–852, 2009.

[3] J. van Doremalen, C. Cucchiarini, and H. Strik. Automatic detection of vowel pronunciation errors using multiple information sources. In *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*, pages 580–585. IEEE, 2009.

[4] M. Gubian, B. Schuppler, J. van Doremalen, E. Sanders, and L. Boves. Novelty Detection as a Tool for Automatic Detection of Orthographic Transcription Errors. In *Proc. of 13-th International Conference on Speech and Computer (SPECOM'2009)*, 2009.

[5] C. Van Bael, L. Boves, H. van den Heuvel, and H. Strik. Automatic phonetic transcription of large speech corpora. *Computer Speech & Language*, 21(4):652–668, 2007.

[6] W.W. Royce. Managing the development of large software systems. In *proceedings of IEEE WESCON*, volume 26. Los Angeles, 1970.

[7] R.S. Pressman and D. Ince. *Software engineering: a practitioner's approach*. McGraw-Hill New York, NY, 1982.

[8] B.W. Boehm. A spiral model of software development and enhancement. *Computer*, 21(5):61–72, 1988.

[9] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, et al. Manifesto for agile software development. *The Agile Alliance*, pages 2002–04, 2001.

[10] C. Cucchiarini, H. Van Hamme, O. Van Herwijnen, and F. Smits. Jasmin-cgn: Extension of the spoken dutch corpus with speech of elderly people, children and nonnatives in the human-machine interaction modality. In *In LREC*. Citeseer, 2006.

[11] J. Zavrel and W. Daelemans. Evaluatie van part-ofspeech taggers voor het Corpus Gesproken Nederlands. Technical report, Technical report, CGN-Corpusannotatie. Working Paper, 1999.

[12] G. Van Rossum and Centrum voor Wiskunde en Informatica. *Python reference manual*. Centrum voor Wiskunde en Informatica, 1995.

[13] T. Oliphant et al. Numpy, a python library for numerical computations, 2009.

[14] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International joint Conference on artificial intelligence*, volume 14, pages 1137–1145. Citeseer, 1995.

[15] K. Demuynck, J. Roelens, D.V. Compernolle, and P. Wambacq. SPRAAK: An Open Source Speech Recognition and Automatic Annotation Kit. In *Ninth Annual Conference of the International Speech Communication Association*, 2008.

[16] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[17] Mark Humphrys. Continuous output - the sigmoid function. http://www.computing.dcu.ie/~humphrys/Notes/Neural/sigmoid.html.

[18] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[19] Wolfram Alpha LLC. Wolfram|alpha. http://www.wolframalpha.com.

[20] A.B. Barak and A. Shapir. Unix with satellite processors. *Software: Practice and Experience*, 10(5):383–392, 1980.

# A Tabellen

Onderstaande tabellen zijn gebaseerd op het document “Fonemische Transcriptie - Versie 2 (4 april 2000)”

| vb | YAPA |
|---|---|
| lip | I |
| leg | E |
| lat | A |
| bom | O |
| put | Y |
| liep | i |
| buut | u |
| leeg | e |
| deuk | & |
| laat | a |
| boom | o |
| boek | u |

(a) Vocalen

| vb | YAPA |
|---|---|
| fop | f |
| vod | v |
| sap | s |
| zak | z |
| sjaal | S |
| ravage | Z |
| licht | x |
| geen | G |
| heel | h |

(b) Fricatieven

| vb | YAPA |
|---|---|
| scène | E: |
| freule | @: |
| zone | O: |
| vaccin | E˜ |
| congé | O˜ |
| croissant | A˜ |

(c) Leenvocalen

| vb | YAPA |
|---|---|
| put | p |
| bad | b |
| tak | t |
| dak | d |
| kat | k |
| goal | g |

(d) Occlusieven

| vb | YAPA |
|---|---|
| wijs | Eˆ |
| huis | @ˆ |
| koud | Oˆ |

(e) Diftongen

| vb | YAPA |
|---|---|
| maan | m |
| nam | n |
| lang | N |
| oranje | Jj |

(f) Nasalen

| vb | YAPA |
|---|---|
| loop | l |
| rook | r |

(g) Liquidae

| vb | YAPA |
|---|---|
| ja | j |
| weer | w |

(h) Halfvocalen

Tabel 1: Overzicht fonetische klassen