SPC 2009: Luisterexperiment

David Weenink

20 mei 2009

Samenvatting

Dit experiment betreft het randgebied van “phonemic restauration”. In navolging van Shriberg [1992] gaan we een experiment uitvoeren met gefilterde klinkers. Dat betekent opnames maken, klinkermateriaal selecteren, bestandsnamen systematiseren, experient opzetten, experiment uitvoeren en gegevens verwerken, conclusies trekken en een verslag maken in artikelvorm waarin alle bovenstande aspecten worden behandeld.

1 Bestandsnamen

Geef de bestanden met de opnames van de zinnetjes “In sVs en in sVsse zit de V” en “In sYs en in sVze zit de V” de naam `Snn_sVs_tt.wav`, waarbij “tt” één uit het rijtje van tabel 2 is en “nn” een *tweecijferige* sprekercodering uit tabel 1.

De stimuli voor het luisterexperiment krijgen namen waarin behalve de spreker en de klinker ook het soort bewerking dat de stimulus heeft ondergaan is af te leiden: `Snn_Vtt_Ck.wav` . Deze bewerkingen staan in paragraaf 2.4.

Bijvoorbeeld voor klinker *ie* van spreker *02* in de conditie *LPN/HPF* is de bestandsnaam `S02_Vie_C4.wav`. Let op de kleine letters en de hoofdletters!

2 Het spraakmateriaal

De twaalf Nederlandse klinkers uit de voorgelezen zinnentjes ”In sVs en in sVsse zit de V” en “In sYs en in sVze zit de V” worden gebruikt om als basis te dienen voor het experiment.

2.1 Selectie van de klinkers

Markeer begin- en eindtijd van de klinkers in sVs, sVsse, sVze en in V op *nuldoorgangen*. Neem alleen stemhebbende delen van de klinker mee. Zet de eindgrens zo dat de intensiteit bij de eindgrens niet meer dan 15 dB onder de maximale intensiteit van de klinker ligt. Label de drie klinkers in elke zin op één intervaltier met de codes zoals weergegeven in tabel 2. Dit om de latere verwerking beter te kunnen standaardiseren.

2.2 Klinkerduren bepalen

Bepaal de klinkerduren van je 12 klinkers met een script. De resultaten komen in een tabel te staan met vijf kolommen (spreker, klinker, begintijd, eindtijd, duur). In paragraaf 8 staan scripts die je hiervoor kunt gebruiken.

2.3 |Formantfrequenties bepalen

Iedere student meet van zijn eigen klinkers op 40 ms vanaf het begin van elke klinker de eerste drie formantfrequenties samen met de toonhoogte en zet deze in een tabel. Je bepaalt ook klinkerduur. Je kunt de Table maken via `Create Table... Snn 12 speaker vo` waarbij in "Snn", de "nn" jouw sprekernummer is.

2.4 Bewerkingen

Vier verschillende stimuli. LP (low pass): Laagdoorlaat gefilterd, filteramplitude is 1 bij 700 Hz en 0 bij 1000 Hz. HP (high pass) is hoogdoorlaat (eigenlijk bandgefilterd), filteramplitude is 0 bij 1000 Hz en 1 vanaf 1300 Hz tot 10000 Hz, en 0 vanaf 10300 Hz.

1. C1: Ongemodificeerde klinkers 0-10 kHz, 100 ms duur.
2. C2: LPV. Klinkers laagdoorlaat gefilterd.
3. C3: LPV/HPN. Klinkers laagdoorlaat gefilterd in 900 ms ruis; de klinker begint bij 400 ms. De ruis is complementair en randomUniform(-0.5,0.5) hoogdoorlaatgefilterd.
4. C4: LPN/HPV. Laagdoorlaat gefilterde ruis met hoogdoorlaat gefilterde klinkers.
5. C5: HPV. Hoogdoorlaat gefilterde klinkers.

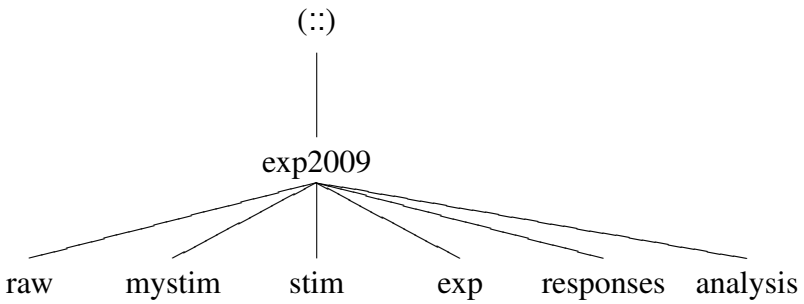
Duurnormalisatie. Alle klinkers moeten dezelfde duuur krijgen (100 ms) (`Sound:Extract part... 0 duur rectangular 1 no`)

Fade-in en fade-out gedurende 10 ms.

Amplitudenormalisatie door eerst de maximale amplitude van een klinker op 0.5 te zetten (`Sound: Scale peak... 0.5`)

Optellen van klinker en witte ruis (randomUniform (-0.5,0.5)).

3 Experiment opzetten



Figuur 1: De directoryhiërarchie voor het experiment.

Maak een map aan met de naam "exp2009". In deze map komt het script om het experiment te draaien samen met het programma Praat zodat je het hele experiment in principe vanuit deze map kunt draaien. Daarnaast bevat de map exp2009 nog de volgende submappen zoals figuur 1 laat zien:

raw Bevat de ruwe onbewerkte studio-opnames, de Snn_sVs_vv.wav bestanden en de TextGrids die de klinker annoteren. Uit deze map mag dan verder alleen gelezen worden. Alle stimuli moeten uit deze map via scripts geproduceerd kunnen worden.

mystim Bevat scripts om je eigen stimuli te maken uit het materiaal in raw. De eigen stimuli komen ook als wav-bestand in deze map.

stim Bevat alleen de stimuli die voor het experiment nodig zijn (je eigen 60 stimuli en de nx60 stimuli van de andere studenten).

exp De scripts om het experiment te verwerken.

responses Bevat alleen de bestanden die de antwoorden van de luisteraars bevatten.

analysis De werkmap met analysegegevens van de stimuli.

Iedere student zorgt dat zijn eigen 60 stimuli (5 condities 12 klinkers) goed zijn.

4 Experiment uitvoeren

Voor het uitvoeren van experiment moeten in het algemeen een aantal keuzes gemaakt worden.

4.1 Hoeveel stimuli

Het aantal stimuli dat we aan willen of aan kunnen bieden in een experiment hangt af een aantal factoren. Factoren zijn o.a. beschikbaarheid van voldoende goede opnames, beschikbaarheid van proefpersonen (pp), beschikbaarheid van experimenteerruimte etc. De statistische betrouwbaarheid die we willen stet ook voorwaardes aan het minimale aantalproefpersonen die we nodig hebben. We willen dit experiment kunnen doen gedurende één les van ongeveer vijfenveertig minuten dit zijn ongeveer 2700 secondes. Onze stimuli duren maximaal 0.9 s. Als we ongeveer vier of vijf secondes schatten voor de tijd die een pp nodig heeft om een stimulus te horen, deze te verwerken en een responsie te geven dan kunnen we uitrekenen dat we zo tussen de 540 (=2700/5) en 675 (=2700/4) stimuli kunnen aanbieden. We hebben dan nog geen rekening gehouden met noodzakelijke tussentijdse pauzes omdat pp over het algemeen geen vijfenveertig minuten aaneengesloten geconcentreerd kunnen blijven.

Gegeven deze grenzen aan het aantal stimuli kunnen we stellen dat het aantal spekers die we kunnen gebruiken maximaal elf is (elf sprekers leveren elf maal zestig is zeshonderzestig stimuli). Wanneer we dit aantal sprekers willen gebruiken dan kunnen we het materiaal van elke spreker maar één keer aanbieden aan de pp. Elke stimulus wordt dan precies één keer gebruikt in het experiment.

4.2 Hoe bieden we de stimuli aan

Om voor iedereen dezelfde luistercondities te kunnen garanderen moeten we de stimuli aanbieden via een koptelefoon.

4.3 Wat voor type experiment

We willen dat een proefpersoon elke klank die ze gehoord heeft benoemd. In ons experiment willen we dat de proefpersoon kiezen uit een vast aantal voorgeschreven mogelijkheden. Zo'n experiment heet een *gedwongenkeuze-experiment*. Met behulp van Praat kunnen we zo'n luisterexperiment uitvoeren. We moeten dan een tekstbestand maken waarin de beschrijving van het experiment staat. Hoe dit bestand er precies uit moet zien kun je zien de de Help. ExperimentMFC . The experiment file. Uit de hoeveelheid regels die dit bestand beslaat kun je zien dat er nog al wat komt kijken bij zo'n luisterexperiment. In paragraaf 8.4 staat een script dat gegeven het aantal sprekers waarmee je het experiment wilt doen het complete experiment-bestand genereert.

4.4 Hoe moet de proefpersoon responsies geven

Omdat we het experiment via een computer met Praat uitvoeren liggen als responsies eigenlijk alleen ingetikte tekst of aanwijzen met een muis voor de hand. Andere responsies zoals via spraak of via invullen van een scoreformulier zijn dan minder geschikt. We hebben hier alleen keuzes uit een vaste set en daarvoor is aanklikken via een muis het meest geschikt.

We hebben de volgende layout voor de responsies op het scherm:

eu uu ee ie aa oo oe

u e ¡ a o

Deze layout is gekozen om zo duidelijk mogelijk de klinker te kunnen selecteren. Om verwarring, door het schriftbeeld, tussen lange klinkers en de korte klinkers te minimaliseren zijn de lange klinkers boven de korte geplaatst. Bovendien staat elke lange klinker zo dicht mogelijk bij de korte klinker waarmee hij op grond van zijn spelling mee verward zou kunnen worden.

4.5 Welke instructie geven we de proefpersoon

De pp moet een korte instructie krijgen over wat zij te horen krijgt en wat ze geacht wordt te doen tijdens het experiment. Leg uit welke klanken bedoeld worden met de opschriften. Meestal zijn er ook wat stimuli aan het begin om te oefenen. In

het script in paragraaf 8.4 hebben we ook rekening gehouden met oefenstimuli en lassen we een pause in als er een bepaalde hoeveelheid stimuli afgeluisterd zijn.

4.6 Wanneer voeren we het experiment uit

Via `Read from file...` lees je het experiment-bestand in. Er verschijnt een nieuw experimentMFC-object. Kies nu de optie “Run” en het experiment start met een leeg veld met in het midden de tekst “Klik om te beginnen.” Wees er op bedacht dat na de klik onmiddelijk de eerste stimulus volgt. Na de klik op één van de twaalf klinkers volgt snel hierna de volgende stimulus: : : Na tweehonderd stimuli is er de mogelijkheid om een (korte) pauze te nemen als de tekst “Korte pause. Klik om verder te gaan.”. Na de klik volgt onmiddelijk weer de volgende stimulus: : :

Als uiteindelijk de tekst "Einde van het experiment" verschijnt dan kun je het experimentvenster sluiten door File. Close te kiezen. Kies nu de optie Extract results en hierna verschijnt een nieuw object van het type ResultsMFC. Schrijf dit resultaat onmiddelijk weg onder de naam `Pnn.ResultsMFC`, in de directory `responses` waarbij `nn` je tweecijferige nummer is. Upload het naar BlackBoard.

5 Gegevens verwerken en uitwisselen

Uitwisseling van bestanden gaat via BlackBoard. Communication. Group Pages . exp2009 . File Exchange. Hier kan iedereen zijn geluiden en gegevens kwijt.

Om het overzichtelijk te houden moet iedereen zijn audiostimuli (60 stuks) via één containerbestand, een zogenaamde Collection, opsturen. Deze Collection moet je de naam `Snn_V60.Collection` geven, waarbij uiteraard nn staat voor je tweecijferige sprekercodering. De snelste manier om een Collectionbestand te maken is om precies alle 60 stimuli als sound in het objectvenster aanwezig te hebben, ze dan alle 60 samen te selecteren en dan Write. Write to binary file... te kiezen. Dit kan het snelste via een script waarbij je kunt controleren of inderdaad alle stimuli onder alle condities aanwezig zijn (zie paragraaf 8.4.5 en 8.4.6).

6 Verslag schrijven

Het verslag begint met de titel, auteur, datum. Dan een korte samenvatting, gevolgd door een inleiding die het experiment kort beschrijft. Dan een sectie over het

spraakmateriaal hoe het is verzameld en verwerkt. Dan beschrijving hoe het experiment is gedaan, welke proefpersonen en hoeveel, de luistercondities. Dan een stuk over de uitkomsten van het experiment, dat wordt gevolgd door een discussie/conclusie. Gebruikte literatuur. Een appendix met de scripts die je hebt gebruikt. Je kunt het verslag van het experiment zoals beschreven in Shriberg [1992] als leidraad nemen.

7 Coderingen

Tabel 1: De sprekercoderingen

Naam	Code
Bodelier, Jorine	01
Boersma, Folkert	02
Bootsma,Jael	03
Cruden, Kimberley	04
Frederiks, Gwendolyn	05
Grune, Cherish	06
Knauer, Kevin	07
Kobus, Sarah	08
Martins Lopes, Stephanie	09
Mestrom, Doreen	10
Nijon, Cassandra	11
Ooijevaar, Etske	12
Schuttenhelm, Lisan	13
Steinhauzer, Kathelijne	14
ter Hark, Anne	15
Tuijtel, Johanna	16
Verduijn, Lisanne	17
Waegemaekers, Eileen	18

Tabel 2: De klinkercoderingen met twee tekens.

IPA	tt
u	oe
oː	oo
aː	aa
eː	ee
i	ie
y	uu
øː	eu
ɔ	oh
ɑ	ah
ɛ	eh
ɪ	ih
ʏ	uh

8 Scripts

8.1 Selectie klinkersegmenten

Het volgende script komt in de exp2009 directory te staan en is voor de generatie van ruwe onbewerkte klinkersegmenten. Bij dit script en alle volgende is het van belang dat je het script *eerst bewaart in de exp2009 directory en hierna pas opent* in de scripteditor. Deze volgorde is nodig om er voor te zorgen dat verwijzingen naar de relative directories goed gaan.

```
1  # create_raw_vowel_stimuli.praat
2  # djmw 20080417, 20090325
3
4  basedir$ = "/home/david/tex/ba_spc/2009/exp2009/"
5  from_dir$ = basedir$ + "raw/"
6  to_dir$ = basedir$ + "mystim/"
7  myspeakerid$ = "S00"
8  vowels$ = "aa ah ee eh eu ie ih oe oh oo uh uu"
9  tier = 1
10 for ivowel to 12
11   vowel$ = mid$ (vowels$, (ivowel - 1) * 3 + 1, 2)
12   filenaam$ = "'from_dir$''myspeakerid$'_sVs_'vowel$'"
13   Read from file... 'filenaam$'.wav
14   s = selected ("Sound")
```

```
15    Read from file... 'filenaam$'.TextGrid
16    tg = selected ("TextGrid")
17    select s
18    plus tg
19    Extract intervals where... tier n "is equal to" 'vowel$'
20    n = numberOfSelected ("Sound")
21    if n = 3
22      v1 = selected ("Sound", 1)
23      v2 = selected ("Sound", 2)
24      v3 = selected ("Sound", 3)
25      select v1
26      Scale peak... 0.5
27      Write to WAV file... 'to_dir$''myspeakerid$'_V'vowel$'_C0.wav
28    else
29      exit Number of vowels 'vowel$' is not correct.
30    endif
31    select s
32    plus tg
33    plus v1
34    plus v2
35    plus v3
36    Remove
37  endfor
```

We gaan uit van de directorystructuur in figuur 1 en zoals beschreven in paragraaf 3. Onze spraakbestanden hebben namen van de vorm `Snn_sVs_Vtt.wav`, de corresponderende textgrids hebben dezelfde naam met alleen de extensie `TextGrid` in plaats van `wav`. Beide types bestanden bevinden zich in de subdirectory raw van `exp2009`.

De directory waaruit gelezen wordt is een subdirectory van `exp2009` met de naam `raw`, de directory waar de klinkersegmenten naar toe geschreven worden heeft de naam `mystim`. De variabelen `from_dir$` en `to_dir$` worden gedefiniëerd om deze namen maar één keer te hoeven uitschrijven. In regel 7 krijgt `myspeakerid$` de fictieve waarde `S00`; deze waarde is voor iedereen anders en is voorgebakken in tabel 1. In regel 8 wordt aan de variabele `vowels$` een tekstregel toegekent met daarin de twaalf klinkercoderingen volgens tabel 2 gescheiden door spaties. Omdat een textgrid meer dan één tier kan hebben en we altijd van dezelfde tier willen lezen, definiëren we in regel 9: `tier=1` (wij gebruiken immers maar één tier).

Hierna volgt de hoofdlus van regel 10 tot en met regel 37. In de lus wordt voor elke klinker het corresponderende spraakbestand en zijn textgrid ingelezen om vervolgens de klinkersegmenten te isoleren. In regel 11 definiëren we de

hulpvariable vowel$ waaraan elke keer een andere klinkernaam wordt toegekend via de constructie vowel$=mid$(vowels$,(ivowel-1)*3+1,2). In deze regel wordt de functie mid$ gebruikt. Deze functie copiëert een deel uit een string en geeft dit gecopieerde deel dan terug (de functie eindigt ook op een $-teken, dit betekent dat hij een string terug geeft). Het eerste argument van deze functie is de string waaruit gecopiërd moet worden. Het tweede argument bepaalt de startpositie voor het copiëren en de het derde argument bepaalt hoeveel tekens er vanaf de startpositie gecopiëerd moeten worden. Het eerste teken in een string heeft positienummer 1. In de variabele vowels$ beginnen de klinkercoderingen op posities 1 (aa), 4 (ah), 7 (ee), 10 (eh), en zo voorts. Het tweede argument van de aanroep, (ivowel-1)*3+1, zorgt ervoor dat voor elke waarde van ivowel de goede startpositie wordt doorgegeven zodat de twee goede tekens gecopiëerd worden. De eerste keer dat de lus wordt uitgevoerd heeft ivowel de waarde 1, de startpositie is dan 1 en vowel$ krijgt dus de waarde aa.

In regel 12 wordt de naam van het te lezen bestand samengesteld uit de naam van de directory via de variabele from_dir$, de identificatie van de spreker, via myspeakerid$ en de klinker, via de variabele vowel$. Dus als vowel$ de waarde aa heeft krijgt filenaam$ in dit script de waarde raw/S00_sVs_aa (relatief t.o.v basedir$).

In regel 13 wordt dan het spraakbestand ingelezen nadat eerst nog de .wav extensie aan de filenaam wordt geplakt. Het unieke nummer van het ingelezen soundobject wordt doorgegeven aan de variabele s in regel 14. Via deze variabele s kunnen we via select s voortaan rechtstreeks deze sound selecteren.

In regel 15 wordt de textgrid ingelezen en in regel 16 wordt het unieke nummer van het nieuwe textgrid doorgegeven aan de variabele tg.

In regels 17 en 18 worden de sound en de texgrid samen geselecteerd. In regel 19 gebeurt tenslotte het belangrijkste in de lus: er worden drie stukjes geluid uit de sound gecopiëerd. We herhalen deze regel hieronder.

```
Extract intervals where... tier n "is equal to" 'vowel$'
```

Het praatcommando in deze regel werkt als volgt. Van elk interval in de textgrid dat tekst bevat die identiek is aan de tekst in de variabele vowel$ worden de begin- en de eindtijd van dit interval onthouden en uit de sound wordt het stukje tussen deze grenzen gecopiëerd en als een zelfstandige sound toegevoegd in de objectenlijst. Deze nieuwe geluidjes krijgen allemaal een naam die afgeleid is van de naam van de sound en de variabele vowel$. Als we onze klinkersegmenten goed hebben gelabeld dan zijn er telkens drie intervallen in de textgrid die aan de beschrijving beantwoorden.

De testen in regels 21-30 zijn dus een extra controle op eventuele fouten die we gemaakt kunnen hebben. De test in regel 21 controleert of er wel drie intervallen gevonden zijn met de gewenste tekst. Zo niet dan wordt het script afgebroken en een melding doorgegeven. In regels 22-24 maken we de nieuwe sounds selecteerbaar via hun nummers. We gebruiken alleen de eerste sound en in regel 26 wordt de maximale amplitude geschaald naar 0.5. In regel 27 wordt het nieuwe geluidje weggeschreven met de gecodeerde naam `Snn_Vtt_C0.wav`. We hebben hier voor conditienummer 0 gekozen omdat deze niet in het experiment voorkomt (zie paragraaf 2.4). We zijn nu eigenlijk klaar met deze klinker, de regels 31-36 verwijderen de net gemaakte objecten weer zodat de volgende iteratie van de lus weer met een lege objectenlijst kan beginnen.

8.2 Meten van duren

8.2.1 Het meten van je eigen klinkerduren

Voor het meten van de duren van de klinkersegmenten hebben we eigenlijk alleen de textgrids nodig. Deze bevatten immers de begin- en eindtijd van elke klinker. We gebruiken dezelfde structuur als in het vorige script. In de loop lezen we telkens een textgrid van de betre ene klinker en converteren deze naar een Table zodat we de waardes van de intervallen gemakkelijker kunnen uitlezen. Voor elke klinker lezen we dan een begin- en eindtijd uit deze tabel (tab), berekenen de duur en schrijven deze weg in een nieuwe tabel (result).

```
1  # get_my_vowel_durations.praat
2  # djmw 20090325
3
4  basedir$ = "/home/david/tex/ba_spc/2009/exp2009/"
5  from_dir$ = basedir$ + "raw/"
6  to_dir$ = basedir$ + "analysis/"
7  myspeakerid$ = "S00"
8  vowels$ = "aa ah ee eh eu ie ih oe oh oo uh uu"
9
10 result = Create Table with column names... 'myspeakerid$' 12
11  ... spreker klinker tmin1 tmax1 duur1 tmin2 tmax2 duur2 tmin3 tmax3 duur3
12
13 tier = 1
14 for ivowel to 12
15   vowel$ = mid$ (vowels$, (ivowel - 1) * 3 + 1, 2)
16   select result
17   Set string value... ivowel spreker 'myspeakerid$'
18   Set string value... ivowel klinker 'vowel$'
```

```
  filenaam$ = "'from_dir$''myspeakerid$'_sVs_'vowel$'"
  Read from file... 'filenaam$'.TextGrid
  tg = selected ("TextGrid")
  tab = Down to Table... n 6 n n
  for iv to 3
    select tab
    tmin = Get value... iv tmin
    tmax = Get value... iv tmax
    duur = tmax - tmin
    select result
    Set numeric value... ivowel tmin'iv' tmin
    Set numeric value... ivowel tmax'iv' tmax
    Set numeric value... ivowel duur'iv' duur
  endfor
  select tg
  plus tab
  Remove
endfor

select result
Write to short text file... 'to_dir$''myspeakerid$'_V12_durations.Table
```

8.2.2 Het meten van cumulatieve duren

```
# analysis/get_durations.praat
# djmw 20080605, 20090330

procedure read_all_tables
  .tablelist = Create Strings as file list...
    ... tablelist S*_V12_durations.Table
  .ns = Get number of strings
  .nrows = .ns * 12
  .summarytable = Create Table with column names...
    ... table .nrows spreker klinker duur1 duur2 duur3
  .irowtable = 1
  for .it to .ns
select .tablelist
    table$ = Get string... .it
    .tabit = Read from file... 'table$'
    # sommigen hebben hier Snn staan ipv nn !
    Formula (column range)... spreker spreker
      ... replace$ (self$, "S", "", 0)
    select .summarytable
    for .irow to 12
```

```
21        .spreker$ = Object_'.tabit'$[.irow, "spreker"]
22        Set string value... .irowtable spreker '.spreker$'
23        .klinker$ = Object_'.tabit'$[.irow, "klinker"]
24        Set string value... .irowtable klinker '.klinker$'
25        for .icol to 3
26          .label$ = "duur'.col'"
27          .duur = Object_'.tabit'[.irow, .label$]
28          Set numeric value... .irowtable '.label$' .duur
29        endfor
30        .irowtable += 1
31      endfor
32      select .tabit
33      Remove
34    endfor
35    select .tablelist
36    Remove
37    select .summarytable
38  endproc
39  call read_all_tables
```

Voor het verwerken van de duurinformatie van de andere sprekers moeten we de `Snn_V12_durations.Table` bestanden van BlackBoard halen en in de `analysis` directory plaatsen. Met dit script lezen we deze tabellen één voor één in en verzamelen de duurinformatie in één grote tabel met vijf kolommen getiteld `spreker`, `klinker` `duur1, duur2, duur3`.

Regel 5 maakt een object met daarin de bestandsnamen die beginnen met een "S" en eindigen op "_V12_durations.Table". De volgende regel bepaalt het aantal elementen in het object en dit zal, als alles goed is, gelijk zijn aan het aantal tabelbestanden in de analysis directory. Regel 8 berekent dan het aantal rijen in de verzameltabel. In regel 9 wordt deze nieuwe tabel gemaakt. In regel 11 wordt de rijteller voor de verzameltabel op 1 gezet.

In de lus tussen regel 12 en 31 wordt telkens een tabel ingelezen en elke rij afzonderlijk verwerkt in de lus tussen regels 20 en 28. [1]

De gemiddelde duur van elke klinker kunnen we dan bepalen uit deze tabel. Dit kan met behulp van het commando `Collapse rows...`. Met dit commando kun je gemiddeldes en/of medianen op twee manieren berekenen: op de "gewone" manier tel je elementen op en daarna deel je door het aantal. (Op de logaritmische manier, die we niet gebruiken, tel je de logaritmes van de elementen op, dan middel je en uit deze gemiddelde logaritme kun je dan weer het "gemiddelde"

[1] In regel 17/18 wordt nog een correctie uitgevoerd en een eventueel toegevoegde "S" van het sprekernummer verwijderd.

berekenen.) Wij willen de gemiddelde duur van elke klinker apart. Wij hebben dan te maken met maar een factor `klinker`. Bij `Columns to average:` vullen we dan `duur` in. Ons commando wordt dan `Collapse rows... klinker "" "duur1 duur2 duur3" "" "" ""`. Het tweede, vierde, vijfde en zesde veld zijn dus leeg. De tabel die je nu krijgt bevat dan twaalf rijen en vier kollommen. De eerste kolom met de klinker en de ander kolommen met de gemiddelde duren.

8.3 Generatie van de stimuli

Het volgende script genereert, uitgaande van de gesegmenteerde klinkers (conditie 0), de vijf verschillende varianten hiervan. Dit script moet komen in de hoofddirectory exp2009 van het experiment. Het script bevat alle procedures die nodig zijn en kan na een kleine aanpassing door iedereen zo gebruikt worden. Let op dat je *eerst* het script bewaart in de exp2009 directory en dan pas opent met de scripteditor.

```
# generate_stimuli_from_C0.praat

myspeakerid$ = "S00"
from_dir$ = "mystim/"
to_dir$ = "mystim/"
vowels$ = "aa ah ee eh eu ie ih oe oh oo uh uu"

call create_lp_noise
lpn = selected ("Sound")
call create_hp_noise
hpn = selected ("Sound")
for ivowel to 12
  vowel$ = mid$ (vowels$, (ivowel - 1) * 3 + 1, 2)
  fname_common$ = "'myspeakerid$'_V'vowel$'"
  to_path_common$ = "'to_dir$''fname_common$'"

  Read from file... 'from_dir$''fname_common$'_C0.wav
  s = selected ("Sound")

  #1 100 ms klinkers
  call generate_c1 s
  s100 = selected ("Sound")
  Write to WAV file... 'to_path_common$'_C1.wav

  #2 Lowpass vowels
  call generate_c2 s100
  slpv = selected ("Sound")
  Write to WAV file... 'to_path_common$'_C2.wav
```

```
29
30    #3 lpv+hpn
31    call generate_sum slpv hpn
32    slpvhpn = selected ("Sound")
33    Write to WAV file... 'to_path_common$'_C3.wav
34
35    #5 Highpass vowel
36    call generate_c5 s100
37    shpv = selected ("Sound")
38    Write to WAV file... 'to_path_common$'_C4.wav
39
40    #4 lpn+hpv
41    call generate_sum shpv lpn
42    slpnhpv = selected ("Sound")
43    Write to WAV file... 'to_path_common$'_C5.wav
44
45    select s
46    plus s100
47    plus slpv
48    plus slpvhpn
49    plus shpv
50    plus slpnhpv
51    Remove
52 endfor
53 # De procedures volgen hierna...
```

In dit script gaan we er van uit dat er al twaalf bestanden `S00_Vtt_C0.wav` zijn waarin alleen de totale klinker zit met zijn volledige duur. Van elk van deze twaalf klinkers worden vijf varianten gemaakt volgens de specificaties in paragraaf 2.4.

In regel 3-6 worden een aantal globale variabelen gedefiniëerd. De variabele `myspeakerid$` krijgt de waarde uit tabel 1, hier heb ik de fictieve waarde “S00” gebruikt. De `from_dir$` is de relatieve directory waar de bestanden die eindigen op _C0.wav zich bevinden; hier wordt uit gelezen. De `to_dir$` is het relatieve pad naar de directory waar de 60 nieuwe bestanden naar toe geschreven worden. Deze directory namen zijn relatief t.o.v. de directory *waarin dit script zich bevindt*. Let op deze namen eindigen op een “/”-teken. Dit teken is de universele wijze waarop directories van elkaar gescheiden worden. Op computers met Windows mag je ook de speciale Windows-manier gebruiken met het “\”-teken als scheider, maar gebruik ze niet door elkaar heen!

In de regels 8-11 worden de twee verschillende ruisgeluiden gemaakt die we in dit experiment gebruiken, hun unieke nummer wordt aan variabelen toegekend die we later zullen gebruiken.

De hoofdlus van dit script begint in regel 13 en loopt door tot regel 53. De lus wordt twaalf keer doorlopen waarbij de lusvariabele `ivowel` de waardes 1 t/m 12 krijgt. In regel 14 krijgt de variabele `vowel$` telkens een andere waarde die gecopiëerd wordt uit de string `vowels$`. De eerste keer is de waarde van `vowel$` gelijk aan `aa`. In de twee volgende regels worden twee hulpvariabelen gedefiniëerd die het gemeenschappelijke deel van de bestandsnaam en het relatieve pad beschrijven. Als `ivowel=1` dan wordt: `vowel$ = "aa"`, `fname_common$ = "S00_Vaa"` en `to_path_common$ = "mystim/S00_Vaa"`. In regel 18 wordt dan het bestand gelezen waar de verdere bewerkingen op gedaan worden. In de rest van de lus worden dan telkens de vijf verschillende varianten gemaakt. Hiervoor wordt steeds een procedure aangeroepen die een nieuw geluid maakt dat in het objectvenster verschijnt. Elke procedure zorgt er voor dat het geluidje dat hij nieuw maakt ook geselecteerd is in het objectvenster. Alle nieuwe geluidjes worden dan onder de goede naam en met het goede conditienummer (C1-C5) weggeschreven.

8.3.1 Conditie 1: 100 ms duur

Het volgende script beschrijft de procedure generate_c1. In regel 3 worden de eerste 100 ms van het geselecteerde geluid gecopiëerd in een nieuw sound.

```
1 procedure generate_c1 .sound
2   select .sound
3   .sp = Extract part... 0 0.1 rectangular 1 no
4   call fade_in_out .sp
5   .sp = selected ("Sound")
6   Scale peak... 0.5
7 endproc
```

De procedure fade-in-out is als volgt:

```
procedure fade_in_out .s
  select .s
  .xmax = Get total duration
  .x2 = .xmax - 0.01
  Formula...
  ... if x<=0.01 then self*(1-cos(2*pi*50*x))/2
  ... else
  ...   if x >= .x2 then self*(1+cos(2*pi*50*(x-.x2)))/2
  ...   else self
  ...   endif
  ... endif
endproc
```

8.3.2 Conditie 2: Laagdoorlaatgefilterde klinker

De procedure om een laagdoorlaatgefilterde versie van een geluidje te maken bestaat uit één aanroep van de procedure lpf.

```
procedure generate_c2 .sound
  call lpf .sound
endproc
```

De procedure voor de laagdoorlaatfiltering. We maken hier gebruik van de optie om een sound te filteren waarbij we een formule kunnen opgeven. Let op: deze formule werkt op het spectrum. De filtering gebeurt doordat Praat eerst van het geluid een spectrum maakt, dan de formule toepast op dit spectrum en daarna van het veranderde spectrum weer een nieuw geluid maakt. In de formule laten we de sterktes tussen 700 en 1000 Hz geleidelijk afnemen via een fade-out. Na afloop van `Filter (formula)...` hebben we een nieuwe sound in het objectvenster. De amplitude van deze sound wordt vervolgens geschaald tot maximaal 0.5.

```
procedure lpf .s
  select .s
  .f1 = 700
  .f2 = 1000
  Filter (formula)...
  ... if x < .f1 then self
  ... else
  ...   if x <= .f2 then self*(1+cos(pi*(x-.f1)/(.f2-.f1)))/2
  ...   else 0
  ...   endif
  ... endif
  Scale peak... 0.5
endproc
```

De procedure voor de hoogdoorlaat filtering is eigenlijk een bandfilter. De filterformule is hier de kombinatie van een hoogdoorlaat- en laagdoorlaatfilter. Alle frequentie lager dan 1000 en hoger dan 10300 Hz worden op nul gezet. Fade-in tussen 1000 en 1300 Hz, fade-out tussen 10000 en 10300 Hz.

```
procedure hpf .s
  .fl1 = 1000
  .fl2 = 1300
  .fh1 = 10000
  .fh2 = 10300
  select .s
  Filter (formula)...
  ... if x < .fl1 then 0
  ... else
```

```
  ...   if x <= .fl2 then self*(1-cos(pi*(x-.fl1)/(.fl2-.fl1)))/2
  ...   else
  ...     if x < .fh1 then self
  ...     else
  ...       if x <= .fh2 then self*(1+cos(pi*(x-.fh1)/(.fh2-.fh1)))/2
  ...       else 0
  ...       endif
  ...     endif
  ...   endif
  ... endif
  Scale peak... 0.5
endproc
```

8.3.3 Conditie 3: LPV met HPN

Deze procedure maakt van het laagdoorlaatgefilterde geluid en de hoogdoorlaatgefilterde ruis één sound door ze bij elkaar op te tellen op de goede manier. We maken een copie van de ruis omdat deze het langste in duur is en tellen dan daar het klinkertje bij op.We zorgen ervoor dat dit pas na 0.4 seconde gebeurt. In regel 6 vragen we het samplenummer op bij tijd 0.4 en dit gebruiken we in de formule. Omdat we hier met de naam van het soundobject moeten werken hebben we de `selected$`-functie in regel 3 nodig die de naam van het geselecteerde object teruggeeft. De `selected`-functie, zonder $-teken geeft immers het nummer van het geselecteerde object terug.

```
1 procedure generate_sum .sound .noise
2   select .sound
3   .sound$ = selected$ ("Sound")
4   select .noise
5   .vn = Copy... vn
6   .isample = Get sample number from time... 0.4
7   Formula... self+Sound_'.sound$'[col - .isample]
8 endproc
```

8.3.4 Conditie 4: LPN met HPV

De procedure is identiek aan de procedure die bij conditie 3 gebruikt is.

8.3.5 Conditie 5: Hoogdoorlaatgefilterde klinker

De gebruikte procedure bestaat simpelweg uit het aanroepen van het het hoogdoorlaatfilter.

```
procedure generate_c5 .sound
  call hpf .sound
endproc
```

8.3.6 Het maken van de LPN en de HPN

De ruis voor de LPN en de HPN condities kunnen we zelf maken. Het is gefilterde witte ruis. Witte ruis is een signaal waarin alle frequenties ongeveer even sterk voorkomen d.w.z. dat het amplitudespectrum ongeveer vlak is. De monsterwaardes (amplitudes) in een ruisgeluid kunnen willekeurig variëren. In Praat hebben we twee functies tot onze beschikking die willekeurige monsterwaardes kunnen genereren, `randomGauss(mu,sigma)` en `randomUniform(lower,upper)`. Zie ook de wiskundige inleiding over random numbers. Bij de `randomGauss` variëren de amplitudes om de waarde `mu` met een standaarddeviatie `sigma`. Alhoewel de kans op extreme amplitudewaardes klein is kunnen deze wel optreden wat de schaling moeilijker maakt. We gebruiken daarom de `randomUniform` functie. Hier kunnen we in ieder geval garanderen dat de amplitudes begrensd zijn. Het maken van de hoogdoorlaatgefilterde en laagdoorlaatgefilterde ruis bestaat uit twee stappen. In de eerste stap maken we een stereo ruissignaal dat alle frequentie bevat. In de tweede stap gaan we dit ruissignaal filteren. De volgende procedure beschrijft het maken van de laagdoorlaatgefilterde ruis. Eerst wordt ruis gemaakt waarin alle frequenties voorkomen en deze ruis wordt vervolgens laagdoorlaatgefilterd. Het oorspronkelijke ruisje kan vervolgens weggegooid worden en het nieuwe ruisje wordt geselecteerd.

```
procedure create_lp_noise
  .ns = Create Sound from formula... nu Stereo 0 0.9 44100
    ... randomUniform (-1,1)
  call lpf .ns
  .lpn = selected ("Sound")
  Rename... lpn
  select .ns
  Remove
  select .lpn
endproc
```

Het maken van de hoogdoorlaatgefilterde versie gaat analoog, er moet nu alleen hoogdoorlaatgefilterd worden i.p.v. laagdoorlaat.

```
procedure create_hp_noise
  .ns = Create Sound from formula... nu Stereo 0 0.9 44100
    ... randomUniform (-1,1)
```

```
  call hpf .ns
  .hpn = selected ("Sound")
  Rename... hpn
  select .ns
  Remove
  select .hpn
endproc
```

8.4 Het maken van het experiment

De bedoeling is om een bestand te maken dat aan het formaat beantwoordt zoals dat beschreven is bij “Help. ExperimentMFC. The first example. The experiment file” in Praat. In deze Help staat beschreven hoe het bestand er uit moet zien dat een luisterexperiment beschrijft. Het volgende script maakt automatisch dit experiment-bestand aan voor ons type experiment. Dit experiment-bestand kan dan via Read from file... ingelezen worden in Praat en het experiment kan dan gerund worden. Het is een beetje een lang script en dat komt omdat er nogal veel moet gebeuren. Hier kun je het halen. Het eerste stuk van dit script gaat als volgt:

```
1  # generate_experiment.praat
2  # djmw 20080422
3
4  expfile$ = "spc-2008-test.ExperimentMFC"
5  stimdir$ = "stim/"
6  stimuli$ = "'stimdir$'*.wav"
7  numberOfSpeakers = 1
8  numberOfConditions = 5
9  silenceBeforeNewStimulus = 1
10 numberOfVowels = 12
11 numberOfReplicationsPerStimulus = 1
12 breakAfterEvery = 200
13 numberOfDummies = 5
14
15 call exp_intro
16 call exp_get_stimuli
17 call exp_responses_general
18 call exp_generate_response_buttons
```

In regel 4 staat hoe het bestand gaat heten dat het experiment beschrijft. Deze naam kan vrij gekozen worden. De variabele `stimdir$` geeft aan in welke subdirectory de stimuli staan. De `stimuli$` variable geeft het patroon aan van de naam van de stimuli wav-bestanden. De `numberOfSpeakers` en `numberOfConditions`

geven aan van hoeveel sprekers er bestanden staan in de directory en hoeveel condities. Als er alleen nog maar materiaal van jezelf in de `stimdir$` staat dan kun je `numberOfSpeakers` op 1 laten staan en hiermee testen.[2] De variabele in regel 11 beschrijft hoe vaak elk bestand als stimulus gebruikt gaat worden en dat is precies één keer. Na elke 200 stimuli mag de proefpersoon even rusten (regel 12). De volgende regel geeft aan dat de eerste vijf stimuli om te oefenen zijn. De volgende vier procedure-aanroepen maken dan uiteindelijk het experiment-bestand.

De eerste procedure `exp_intro` schrijft algemene informatie over het experiment naar het experiment-bestand. De volgende procedure `exp_get_stimuli` leest de stimulibestanden en zet de namen in een speciale volgorde in het experiment-bestand. De derde procedure `exp_responses_general` maakt wat algemene teksten die op het experimentscherm komen en de laatste procedure `exp_generate_response_buttons` maakt de responsieknoppen en de teksten hierop. Je komt het snelst achter de werking van het script door er voor te zorgen dat jouw 60 stimulibestanden in de subdirectory stim staan en dit script in de hoofddirectory exp2009. Pas de sprekeridentificatie in het script aan en run vervolgens het script.

8.4.1 De intro

De algemene beschrijving van het experiment wordt als volgt automatisch gemaakt:

```
procedure exp_intro
  .intro$ = """ooTextFile""'newline$'"
  .intro$ = .intro$ + """ExperimentMFC 5""'newline$'"
  .intro$ = .intro$ + "stimuliAreSounds? <yes>'newline$'"
  .intro$ = .intro$ + "stimulusFileNameHead = ""'stimdir$'""'newline$'"
  .intro$ = .intro$ + "stimulusFileNameTail = "".wav""'newline$'"
  .intro$ = .intro$ + "stimulusCarrierBefore = """"'newline$'"
  .intro$ = .intro$ + "stimulusCarrierAfter = """"'newline$'"
  .intro$ = .intro$ +
  ... "stimulusInitialSilenceDuration = 'silenceBeforeNewStimulus' seconds'newline$'"
  .intro$ = .intro$ + "stimulusMedialSilenceDuration = 0.5'newline$'"
  .intro$ > 'expfile$'
endproc
```

In elke regel van deze procedure wordt aan de variabele `.intro$` een stuk tekst toegevoegd, de één na laatste regel van dit script schrijft de totale tekst dan weg naar het experiment-bestand. Omdat er in het experiment-bestand ook aanhalingstekens (") moeten staan en in Praat het aanhalingsteken gebruikt wordt om een

[2] In dit geval moeten er dan `numberOfCondtions` x `numberOfVowels` = 60 bestanden aanwezig zijn.

string te definiëren, moeten we iets speciaals doen om in een string een aanhalingsteken weer te geven: we verdubbelen ieder aanhalingsteken in de string. Dus om de variabele a$ de string a"b toe te kennen moeten we a$= "a""b" zeggen. Er staan daarom heel wat dubbele aanhalingstekens in deze procedure. Ook moeten we expliciet het nieuwe-regelteken toevoegen als we willen dat de volgende regel op een nieuwe regel begint.

In de laatste regel van de procedure wordt de tekst inde variabele .intro$ naar het experiment-bestand geschreven. Het ">"-teken betekent in dit geval dat het bestand gemaakt wordt en als het experiment-bestand al bestond dan wordt het gewoon overschreven.

8.4.2 De beschrijving van de stimuli

De volgende procedure beschrijft de stimuli.

```
1  procedure exp_get_stimuli
2    .s = Create Strings as file list... fl 'stimuli$'
3    .nds = Get number of strings
4    .nwanted = numberOfSpeakers * numberOfConditions * numberOfVowels
5    if .nds <> .nwanted
6       exit Het aantal stimuli ('.ns') klopt niet ('.nwanted')!!
7    endif
8    call randomize_stimuli .s
9    .rands = selected ("Strings")
10   .ns = Get number of strings
11   .text$ = "numberOfDifferentStimuli = '.ns''newline$'"
12   for .i to .ns
13     .file$ = Get string... .i
14     .stim$ = replace_regex$ (.file$, ".wav$", "", 1)
15     .text$ = .text$ + "  ""'.stim$'"" """"'newline$'"
16   endfor
17   .text$ >> 'expfile$'
18   select .s
19 #  plus .rands
20    Remove
21 endproc
```

In regel 2 gaan we in alle bestanden die overeenkomen met het patroon dat gegeven wordt door de waarde van variabele stimuli$, in een stringsobject zetten. Een stringsobject bestaat uit een rijtje strings. In ons geval beantwoorden alle bestanden in de directory stim die eindigen op .wav aan het patroon (het *-teken staat in dit geval voor een willekeurige verzameling tekens). We controleren of het aantal gevonden bestanden klopt met de beschrijving. De procedure randomize_stimuli

die hierna besproken wordt, zet de stimuli in de goede volgorde en maakt een aantal oefenstimuli erbij (`numberOfDummies`). Deze procedure levert een nieuw stringsobject af. Het aantal elementen in dit object wordt dan bewaard. In regels 12-16 wordt dan van elke string in de strings het laatste .wav deel gestript. Dit gebeurt met een functie die met reguliere expressies werkt.[3] De naam, zonder de extensie, wordt dan volgens het speciale format voor het experimentbestand geschikt gemaakt in regel 15.

In regel 17 wordt de text aan het reeds bestaande bestand geplakt. De >> betekent dat geappend wordt. De strings met de stimuli in de goede volgorde hebben we nu niet meer nodig maar wordt nog niet weggegooid. Als deze na afloop van het script in het objectvenster blijft dan kun je via de Editknop zien welke namen er in staan ter controle. Vandaar dat regel 19 een commentaarregel is.

Het in de goede volgorde zetten gaat via het volgende script. In het experiment worden eerst alle stimuli van conditie 1 in een willekeurige volgorde aangeboden aan de luisteraar en dan de stimuli van de andere condities allemaal door elkaar heen. De laatste `numberOfDummies` stimuli van conditie 1 worden tenslotte gecopiëerd en ook aan het begin gezet. Dit zijn de oefenstimuli en bij de resultaten kunnen dan de eerste `numberOfDummies` responsies weggelaten worden.

```
1  procedure randomize_stimuli .string
2    .vowels$ = "aa ah ee eh eu ie ih oe oh oo uh uu"
3    select .string
4    .ns = Get number of strings
5    .rands = Copy... rands
6    # eerst sorteren op conditie
7    .numberInC1 = numberOfSpeakers*numberOfVowels
8    .posc1 = 1
9    .poscr = .numberInC1 + 1
```

[3] Met behulp van reguliere expressies kun je heel algemene zoekpatronen formuleren. De reguliere expressie ".wav$" betekent dat gezocht wordt naar een willekerig teken dat gevolgd wordt door wav op het eind een string. Hier heeft het dollarteken dus weer een andere betekenis (einde string) en de punt ook (willekeurig teken). Zie “Help. Regular expressions” in Praat voor meer informatie over reguliere expressies.

Let op: een reguliere expressie is niet hetzelfde als een expressie waarmee je algemene patronen voor bestanden maakt. Deze laatste expressie heet een file-globber. Als je zoekt naar alle bestanden die eindigen op “.wav” in de bestandszoeker dan tik je in “*.wav”. De punt “.” betekent hier het letterlijke teken “.” en het“*”-teken betekent een willekeurige reeks tekens (in een reguliere expressie betekent de punt “.” één willekeurig teken en “*” betekent dat de expressie vóór dit teken nul of meer keer kan voorkomen.)

```
  for .i to .ns
     select .string
     .stim$ = Get string... .i
     .vow$ = mid$ (.stim$, 6, 2)
     .index = index (.vowels$, .vow$)
     .index = (.index + 2) / 3
     if .index < 1
        exit Foute klinker in '.stim$'
     endif
     select .rands
     .cond$ = mid$ (.stim$, 10, 1)
     if '.cond$' = 1
       .pos = .posc1
       .posc1 += 1
     else
        .pos = .poscr
        .poscr += 1
     endif
     Set string... .pos '.stim$'
  endfor
  # C1 en de rest apart randomiseren
  select .rands
  .p0 = To Permutation... n
  .p1 = Permute randomly... 1 .numberInC1
  .p2 = Permute randomly... .numberInC1+1 .ns
  plus .rands
  .randps = Permute strings
  # copy last numberOfDummies from C1 and paste at start
  .sdummies = Extract part... .numberInC1-numberOfDummies+1 .numberInC1
  select .randps
  .stmp = Copy... tmp
  plus .sdummies
  .splusdummies = Append
  select .rands
  plus .randps
  plus .sdummies
  plus .stmp
  plus .p0
  plus .p1
  plus .p2
  Remove
  select .splusdummies
  Rename... stimuli_'numberOfDummies'_dummies
endproc
```

Er wordt eerst een copie gemaakt en in deze copie worden dan in het eerste deel de namen die “C1” bevatten gezet en de rest wordt hierachter gezet (de namen zijn immers van de vorm `Snn_Vtt_Cn`). Het aantal stimuli in conditie 1 is gelijk aan het aantal sprekers maal het aantal klinkers per spreker (regel 7). De volgende twee variabelen definiëren de beginposities voor C1 en voor de rest. In de lus wordt het ene stringsobject sequentieel doorlopen en afhankelijk van het feit of de gevonden string van het type C1 is of niet op de goede positie in het andere stringsobject gecopieerd. In regels 13-18 wordt nog gecontroleerd of de klinkernaam goed is (misschien een beetje overbodig). In regel 20 wordt de conditie in een variabele gezet. Als conditie 1 is dan krijgt de variabele .pos de waarde die .posc1 heeft en hierna wordt .posc1 met één opgehoogd en de string op de goede positie gezet in het stringsobject. Als de lus helemaal doorlopen is dan staan alle C1-stimuli vooraan in het stringsobject.

In het volgende stuk van de procedure, in de regels 31-36, worden het deel waar de C1-stimuli staan en de rest afzonderlijk door elkaar gehusseld (gerandomiseerd). Hierbij wordt het permutatieobject te hulp geroepen.

In regel 38 worden de laatste `numberOfDummies` uit het C1-deel gecopiëerd en in de volgende regels uiteindelijk voor het begin geplaatst.

8.4.3 Het algemene responsiedeel

Vanwege de lange regels is het lettertype hier iets kleiner gekozen.

```
procedure exp_responses_general
 .text$ = "numberOfReplicationsPerStimulus = 'numberOfReplicationsPerStimulus''newline$'"
 .text$ = .text$ + "breakAfterEvery = 'breakAfterEvery''newline$'"
 .text$ = .text$ + "randomize = <CyclicNonRandom>'newline$'"
 .text$ = .text$ + "startText = ""Klik om te beginnen.""'newline$'"
 .text$ = .text$ + "runText = ""Kies de klinker die je hebt gehoord.""'newline$'"
 .text$ = .text$ + "pauseText = ""Korte pause. Klik om verder te gaan.""'newline$'"
 .text$ = .text$ + "endText = ""Einde van het experiment.""'newline$'"
 .text$ = .text$ + "maximumNumberOfReplays = 0'newline$'"
 .text$ = .text$ + "replayButton = 0 0 0 0 """" """"'newline$'"
 .text$ = .text$ + "okButton = 0 0 0 0 """" """"'newline$'"
 .text$ = .text$ + "oopsButton = 0 0 0 0 """" """"'newline$'"
 .text$ >> 'expfile$'
endproc
```

8.4.4 Het responsiescherm

Het responsiescherm heeft de layout zoals beschreven in paragraaf 4. De bovenste rij knoppen bevatten de zeven lange klinkers en de onderste rij de vijf korte. Ze

zijn zo gerangschikt dat elke korte klinker direkt onder de corresponderende lange klinker staat.

```
procedure exp_generate_response_buttons
  .text$ =  "responsesAreSounds? <no> """" """" """" """" 0 0'newline$'"
  .text$ = .text$ + "numberOfDifferentResponses = 12'newline$'"
  .nx = 7
  .xr1 = 0.2
  .xr2 = 0.8
  .dxr = (.xr2   .xr1) / 7
  .xsize = .dxr   3/4
  .yr1 = 0.3
  .yr2 = 0.6
  .dyr = (.yr2   .yr1) / 2
  .ysize = .dyr   3 / 4
  .buttons$= "eu uu ee ie aa oo oe"
  .responses$ = "eu uu ee ie aa oo oe"
  .y1 = .yr2   .dyr + 1/8   .dyr
  .y2 = .y1 + .ysize
  for .i to 7
    .button$ = mid$ (.buttons$, (.i  1) 3+1, 2)
    .resp$ = mid$ (.responses$, (.i  1) 3+1, 2)
    .x1 = .xr1 + (.i  1)   .dxr + .dxr /8
    .x2 = .x1 + .xsize
    .text$ =.text$ +
    ... "  '.x1:3' '.x2:3' '.y1:3' '.y2:3' ""'.button$'"" 20
    ... """" ""'.resp$'""'newline$'"
  endfor
  .buttons$= "u e i a o"
  .responses$ = "uh eh ih ah oh"
  .y1 = .yr1 + 1/8   .dyr
  .y2 = .y1 + .ysize
  for .i to 5
    .button$ = mid$ (.buttons$, (.i  1) 2+1, 2)
    .resp$ = mid$ (.responses$, (.i  1) 3+1, 2)
    .x1 = .xr1 + .i   .dxr + .dxr /8
    .x2 = .x1 + .xsize
    .text$ =.text$ +
    ... "  '.x1:3' '.x2:3' '.y1:3' '.y2:3' ""'.button$'"" 20
    ... """" ""'.resp$'""'newline$'"
  endfor
  .text$ =.text$ + "numberOfGoodnessCategories = 0'newline$'"
  .text$ >> 'expfile$'
endproc
```

He, he we zijn er.

8.4.5 Stimuli in één Collection zetten

Het volgende script selecteert alle stimuli voor spreker S00 en schrijft ze in een collection bestand weg. Een collection is een bestand waarin een willekeurig aantal objecten kunnen zitten. Bij het inlezen van een collection verschijnen alle objecten in het objectvenster. De collection wordt weggeschreven in de exp2009 directory.

```
1  # stimuli_to_collection.praat
2  # djmw 20080507
3
4  myspeakerid$ = "S00"
5  from_dir$ = "mystim/"
6  vowels$ = "aa ah ee eh eu ie ih oe oh oo uh uu"
7  select all
8  nocheck Remove
9
10 for ivowel to 12
11   vowel$ = mid$ (vowels$, (ivowel - 1) * 3 + 1, 2)
12   for ic to 5
13     Read from file... 'from_dir$''myspeakerid$'_V'vowel$'_C'ic'.wav
14   endfor
15 endfor
16 select all
17 Write to binary file... 'myspeakerid$'_V60.Collection
```

Het script leest eerst alle stimuli uit de goede directory en er komen dan zestig nieuwe sounds bij in het objectvenster. Deze zestig sounds worden op het einde van het script samen geselecteerd (regel 16) en om dan in één bestand weggeschreven te worden. Om er zeker van te zijn dat alleen deze sounds aanwezig zijn worden eerst alle mogelijk aanwezige objecten weggegooid in regels 7 en 8 (Remove geeft een foutmelding als er niks te removen valt en het script wordt afgebroken. De `nocheck` laat het script doorlopen.)

In regels 10-15 worden in een dubbele lus alle stimuli ingelezen, voor elke klinker zijn vijf variaties.

8.4.6 Stimuli uit een Collection halen

Het volgende script haalt de stimuli van de fictieve sprekers S00 en S30 uit een collection en schrijft voor elke spreker zestig sounds weg als wav-bestanden.

```
1  # stimuli_from_collection.praat
2  # djmw 20080509, 20090506
3
```

```
4  nspeakers = 2
5  speakers$ = "S00 S21"
6
7  to_dir$ = "stim/"
8
9  for ispeaker to nspeakers
10   speakerid$ = mid$ (speakers$, (ispeaker - 1) * 4 + 1, 3)
11   call get_one_speaker 'speakerid$'
12 endfor
13
14 procedure get_one_speaker .speakerid$
15   Read from file... '.speakerid$'_V60.Collection
16   .ns = numberOfSelected ("Sound")
17   if .ns <> 60
18     exit Aantal stimuli ('.ns') klopt niet!
19   endif
20   for .i to .ns
21     sound'.i' = selected ("Sound", .i)
22   endfor
23   for .i to .ns
24     select sound'.i'
25     .name$ = selected$ ("Sound")
26     Write to WAV file... 'to_dir$''.name$'.wav
27     Remove
28   endfor
29 endproc
```

Regel 3 definiëert het aantal sprekers die je in het experiment gebruikt. De sprekeridentificaties worden in de volgende regel gespecificeerd. In de lus tussen regels 9 en 12 wordt voor elk van de sprekers zijn collectie ingelezen en de wav-files gemaakt. Het eigenlijke werkt gebeurt in de procedure `get_one_speaker` die in regel 14 begint. In regel 15 wordt de collection ingelezen. Er verschijnen dan een groot aantal nieuwe sounds in het objectvenster. In regel 16-19 wordt gecontroleerd of er inderdaad zestig objecten geselecteerd zijn. Regels 21-23 lopen het rijtje geselecteerde sounds af en definiëren variabelen `sound1` t/m `sound60` om deze sounds later één voor één te kunnen selecteren. In de volgende lus wordt elk object geselecteerd, zijn naam opgevraagd, met deze naam als wav-bestand weggeschreven en daarna weer weggegooid. Als de procedure is uitgewerkt en er geen foutmeldingen zijn verschenen zijn er `nspeakers*60` nieuwe bestandjes gemaakt in de goede directory.

8.5 Het verwerken van het experiment

8.5.1 Verzamelen in een tabel

Het volgende script verzamelt alle resultaten van de proefpersoonbestanden uit de directory responses en zet deze allemaal in één grote tabel.

```
# results_to_table.praat
# djmw 20080509,20090506

numberOfDummies = 5
from_dir$ = "responses/"
to_dir$ = "responses/"

resultfiles = Create Strings as file list... results 'from_dir$'P*.ResultsMFC
npp = Get number of strings
numberOfTrials = npp * 60
table = Create Table with column names... table numberOfTrials subject speaker condi

trial = 1
for ip to npp
  select resultfiles
  result$ = Get string... ip
  subject$ = mid$ (result$, 2, 2)
  rid = Read from file... 'from_dir$''result$'
     nt = Get number of trials
  for it from numberOfDummies+1 to nt
    select rid
    stim$ = Get stimulus... it
    resp$ = Get response... it
    speaker$ = mid$ (stim$, 2, 2)
    vowel$ = mid$ (stim$, 6, 2)
    condition$ = mid$ (stim$, 10, 1)
    select table
    Set string value... trial subject 'subject$'
    Set string value... trial speaker 'speaker$'
    Set string value... trial condition 'condition$'
    Set string value... trial stim 'vowel$'
    Set string value... trial resp 'resp$'
    trial += 1
  endfor
  select rid
  Remove
endfor

select table Write to short text file... 'to_dir$'results.Table
```

8.5.2 Verwarringsmatrices

Het volgende script maakt voor elke conditie een verwarringsmatrix. Bewaar dit script in de `responses` directory! In Praat is een verwarringsmatrix van het type "Confusion". Een verwarringsmatrix is een tabel waarin de rijen gelabeld zijn met de stimuluscategoriën en de kolommen met de responscategoriën. In dit experiment zijn dus zowel de rij- als de kolomlabels de tweeletterige klinkersymbolen. De tabel laat zien hoe vaak en welke responses gegeven zijn op elke stimuluscategorie.

De procedure `get_confusion` berekent met de responstabel uit paragraaf 8.5.1 een verwarringsmatrix. Alleen als een regel uit de responstabel aan een een bepaalde voorwaarde voldoet wordt hij meegeteld. Dit wordt getest in regel 51 van het script. Deze `.selection$` wordt als een stringparameter meegegeven aan de procedure. Zie regel 21 voor de aanroep van deze procedure. Deze manier biedt uitbreidingsmogelijkheden. Enkele van deze zijn in regels 4 en 5 als commentaar gegeven. Als je bijvoorbeeld wilt testen hoe subject 20 gescoord heeft dan kun je de conditie in regel 21 veranderen in "`condition='ic' and subject=20`". Door het subject nummer in te geven kun je dus ook je eigen score bekijken!

```
1  # confusions_from_table.praat
2  # djmw 20080521, 20090519
3
4  #selection$ = "subject=21 and condition<>0"
5  #selection$ = "condition=1"
6
7  condense = 0
8  tr = Read from file... results.Table
9
10 Erase all
11
12 if condense = 0
13   Font size... 6
14   h = 2.4
15 else
16   h = 1.5
17   Font size... 10
18 endif
19
20 for ic to 5
21  call get_confusion tr condition='ic'
22  cf = selected ("Confusion")
23  Rename... cm'ic'
24  fc = Get fraction correct
```

```
 printline Condition 'ic': Percentage correct: 'fc:1%'
 if condense <> 0
   call condense cf
   cm'ic'= selected ("Confusion")
   Rename... cm'ic'c
 endif
 Select outer viewport... 0 6 (ic-1)*h ic*h
 Draw as numbers... y free 5
endfor

procedure get_confusion .tr .selection$
  .vowels$ = "aa ah ee eh eu ie ih oe oh oo uh uu"
  select .tr
  .nr = Get number of rows
  .tor = Create TableOfReal... conf 12 12
  for .i to 12
    .vow$ =  mid$ (.vowels$, (.i - 1) * 3 + 1, 2)
    Set row label (index)... .i '.vow$'
    Set column label (index)... .i '.vow$'
  endfor
  for .ir to .nr
    subject = Object_'.tr' [.ir, 1]
    speaker = Object_'.tr' [.ir, 2]
    condition = Object_'.tr' [.ir, 3]
    stim$ = Object_'tr'$ [.ir, 4]
    resp$ = Object_'tr'$ [.ir, 5]
    if '.selection$'
      .irow = (index (.vowels$, stim$) + 2) / 3
      .icol = (index (.vowels$, resp$) + 2) / 3
      .val = Object_'.tor' [.irow, .icol]
      .val += 1
      Set value... .irow .icol .val
     endif
  endfor
  .cf = To Confusion
  select .tor
  Remove
  select .cf
endproc

procedure condense .conf
  select .conf
  .cfF = Condense... ie|uu|ee|ih|uh|eu|eh|aa F 0 Regular Expressions
  .cfB = Condense... oe|oo|oh|ah B 0 Regular Expressions
  select .cfF
```

```
70    Remove
71    select .cfB
72  endproc
```

Als je de `condense` variabele in regel 7 een andere waarde dan nul geeft, dan krijg je de verwarringsmatrices gecondenseerd tot de twee categoriën F(ront) en B(ack). Door de procedure `condense` aan te passen kun je andere e ecten zichtbaar maken. De `Condense`[4] werkt als volgt bij een Confusion. De eerste Condense vervangt de labels “ie”, “uu”, “ee”, “ih”, “uh”, “eu”, “eh” en “aa” in de rij èn in de kolom door het nieuwe label “F”. [5] Vervolgens worden alle rijen met dit zelfde label “F” bij elkaar opgeteld zodat een nieuw rijtje ontstaat. Voor alle kolommen met label “F” doen we hetzelfde. in de nieuwe verwarringsmatrix die dan ontstaat hebben we de betre ende labels dus gegroepeerd onder één categorie. Deze verwarringsmatrix heeft dus zeven rijen en zeven kolommen minder dan de oorspronkelijke matrix.

Iets analoogs gebeurt voor de achterklinkers oe|oo|oh|ah die in de groep “B” terecht komen. We houden dan een Confusion over die twee rijen en twee kolommen heeft.

8.5.3 Statistiek

Met statistiek kunnen we ruwweg gegevens beschrijven/samenvatten (*description*) en gevolgtrekkingen maken (*inference*). Beschrijven betekent een hoeveelheid gegevens samenvatten op een informatieve manier, meestal met een paar kerngetallen (bijvoorbeeld gemiddelde en spreiding). Gevolgtrekkingen maken betekent metingen generaliseren naar een populatie (testen van hypotheses). Beschrijvende statistiek (descriptive statistics) is meer bezig met de verzamelde gegevens.

Met een t-toets kunnen we bepalen in hoeverre de verschillen tussen twee populaties significant van elkaar verschillen. De vooronderstelling is dat de getallen in elke kolom *normaal* verdeeld zijn.

[4] Let op: We hebben in ons script een procedure die condense heet én we hebben een methode/knop Condense die alleen werkt als een object van het type Confusion geselecteerd is!

[5] We gebruiken hier een reguliere expressie waarbij het pijpsymbool “|” gebruikt wordt om aan te kunnen geven dat we zoeken naar een label dat óf “ie” óf “uu” óf : : : is.

Referenties

Elisabeth E. Shriberg. Perceptual restoration of filtered vowels with noise. *Language and Speech*, 35:127–136, 1992.